Software Security '15 - '16

OWASP Open Souce Review Project

Bor de Kock / 4666798
b.b.d.kock@student.tue.nl

Pieter Kokx / 4671953
p.a.kokx@student.tue.nl

Ylona Meeuwenberg / 4671066
y.w.a.meeuwenberg@student.tue.nl

December 2015

0 Organization

The evaluation was carried out as follows: firstly, the group gathered in a few sessions to install the TestCMS on a server, run some basic tests, analyze the code with the Fortify tool and set up the basis for both the written report and the ASVS Checklist. A division of the requirements was then made: this was in general done by category based on the expertise of the group members.

Our group was mixed in experience and background: one of us got a Bachelor's in mathematics, the others in computer science. One of us has a lot of experience in web development, the others don't. This was both an advantage — different types of requirements could be split over people with different skills — as well as a disadvantage — as it is harder to get a second opinion or a double-check on some of the work. What we did to tackle this issue was walking though the spreadsheet page-by-page, explaining our decisions and judging the criteria together. This real-life-variant on rubber duck debugging helped us identify issues with the reasoning for some of the items, but also made the group members learn more about other parts of the project and the knowledge behind it.

For a significant part of the requirements, we used the live version of the application we ran on a local web server. Take a look at a requirement like V2.9:

Verify that the changing password functionality includes the old password, the new password, and a password confirmation.

Looking at a working version of the application is a fast and easy way of checking the requirement: looking through the code will only make it harder to get the same result.

For the rest of the requirements, we used a combination of Fortify and manual code searching. Fortify was used to produce a list of problems with the code. For each problem, we checked whether the problem actually exists and how it maps to a requirement. Those requirements

were filled in first. For the rest of the requirements, we either looked through the code to locate defects and problems, or we used grep to search through all files with a keyword.

1 Verdict

For all verification requirements, a verdict and motivation is given in the accompanying spreadsheet file. In this report, we will give a short, general remark per chapter in the OWASP ASVS.

V2 Authentication Verification Requirements

The problems with authentication are large. There are no restrictions on passwords strength, no guidelines are provided. Users can pick any password for any period of time, whether or not they used it earlier. It is even possible for users to change the passwords of other users. All of these problems result in failing a lot of requirements.

V3 Session Management Verification Requirements

Session management is included in PHP, so most of the requirements are a trivial pass due to the PHP standards. The functions that are managed by the application are not that strong. There are no additional functionalities, compared to PHP.

V4 Access Control Verification Requirements

The access control is handled poorly. It is possible to change your own status to administrator. When your are an administrator, you can delete other accounts, including other administrators. You can put an account on inactive, but this doesn't mean that this account can't do anything. If the users of the inactive account, want to make it active, he just have to push a button.

V5 Malicious input handling verification requirements

Fortify did give a lot of SQL injection errors. Most of these were false positives, but Fortify was triggered because the application had a custom database class.

V6 Output encoding / escaping

V6 is trivially satisfied: *This section was incorporated into V5 in Application Security Verification Standard 2.0. ASVS requirement 5.16 addresses contextual output encoding to help*

prevent Cross Site Scripting. [1]

V7 Cryptography at rest verification requirements

Only the functions `crypt()` and `hash()` are used. These functions are proven to be unsafe. This is summarized well in the fortify error: *"Antiquated encryption algorithms such as DES no longer provide sufficient protection for use with sensitive data. Encryption algorithms rely on key size as one of the primary mechanisms to ensure cryptographic strength. Cryptographic strength is often measured by the time and computational power needed to generate a valid key. Advances in computing power have made it possible to obtain small encryption keys in a reasonable amount of time."*

V8 Error handling and logging verification requirements

There are only a few things that are logged, and only basic error messages are logged. More information, like the IP address where the request originated, are not stored anywhere. Another issue is that logs are publicly visible if you used the installation instructions. This can be changed in the server configuration.

V9 Data protection verification requirements

TestCMS doesn't keep track of a lot of sensitive data. The login form however, may leak passwords.

V10 Communications security verification requirements

The V10 requirements all apply to the communication over TLS. This means that the verdict depends on the configuration of TLS in the system. The administrator of the server should take care of these requirements.

2 Reflection

Getting from *code* to *code review*, a lot of steps were needed.

As stated in the section on organization, we had some situations where not all group members had the expertise to provide a clear answer. This was solved mostly by dividing up the work, and evaluating with the entire group in a thorough way.

As for the tools used, it was sometimes hard to decide on the proper way to asses a certain requirement. We answered quite a few of them based on the results of running the code, which is a very effective strategy but not truly what a code review is about. Fortify helped a

lot by providing clearly stated problems and analyzing a lot of issues, but a lot of the work still had to be done by hand.

A problem with Fortify was that quite some false positives appeared, mostly these were alleged possible SQL Injections. We disproved all of those by looking through the source code ourselves, and to be sure we also ran the `SQLmap` suite on the running version of the CMS.

In the end, the ASVS provided us with a very clear way of reviewing the code and "filling in the blanks" about the requirements that were given. That is, however, also the weakness of this approach: it is very tempting to focus on giving the right verdicts, where the focus should be on the code in general. This is something the ASVS could focus on to a larger extent.

Working as a group was not an issue, but we preferred dedicating a few larger timeslots to the course over planning a lot of afternoon sessions. This greatly reduced the overhead and made the work more efficient in general.

In terms of lessons learned for developing web applications, not all of us were aware of the OWASP ASVS Checklist. Those that were, will perform security assessments and development earlier on in the process to avoid having to fix issues later on. Those of us who where not aware yet, now need what issues to avoid in general while building web applications and developing other software.

References

[1] OWASP, *Application Security Verification Standard 3.0*, October 2015

Verdict	Explanation
Pass	The system fullfills the requirement
Fail	The system fails the requirement
Trivial Pass (N/A)	The requirement does not apply, so it is trivially satisfied. (For example, for V7.4, if no cryptographic algorithms are used, then all these algorithms have been FIPS 140-2 validated)
Don't know - security requirement unclear	You do not know what the verdict should be, since you do not sufficiently understand the requirement
Don't know - not clear how to check this	You do not know what the verdict should be, since you do not know how you would check whether this requirement holds
Don't know - no time to look into this	You do not know what the verdict should be, since you do not have the time to check this
Don't know - other	You do not know what the verdict should be for some other reason, specify in comments.
NA - check is beyond scope of code	You believe the requirement is not relevant for this system, since the relevant code is not part of the code you were asked to review.
Don't know - other	You believe the requirement is not relevant for this system, since the check is beyond the scope of the system, such as server configuration.
NA - other	You believe the requirement is not relevant for this system for some other reason, specify in the comments.

Regarding tools: if no tools are used feel free to leave the relevant column open. If you use an unlisted tool please chose "other" and specify which in the comments.

#	ASVS Level	Verification Requirement	Verdict	Source Code Referen	Comment	Tool Use
V2.1	1	Verify all pages and resources require authentication except those specifically intended to be public (Principle of complete mediation).	Pass		The pages other than in the /admin/ directory are locked.	
V2.2	1	Verify all password fields do not echo the user's password when it is entered.	Fail		Echos to the source code if reset by admin	Fortify
V2.4	1	Verify all authentication controls are enforced on the server side.	Pass			
V2.6	1	Verify all authentication controls fail securely to ensure attackers cannot log in.	Pass			
V2.7	1	Verify password entry fields allow, or encourage, the use of passphrases, and do not prevent long passphrases/highly complex passwords being entered.	Pass		There are no limits	
V2.8	1	Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism.	Pass			
V2.9	1	Verify that the changing password functionality includes the old password, the new password, and a password confirmation.	Fail		There is only one field: new password.	Other
V2.12	2	Verify that all suspicious authentication decisions are logged. This should include requests with relevant metadata needed for security investigations.	Fail		There are hardly any log files. Specifically, none related to the authentication.	
V2.13	2	Verify that account passwords make use of a sufficient strength encryption routine and that it withstands brute force attack against the encryption routine.	Fail		Crypt is used. It is proven to be unsafe if used with default parameters.	
V2.16	1	Verify that credentials are transported using a suitable encrypted link and that all pages/functions that require a user to enter credentials are done so using an encrypted link.	NA - check is beyond scope of system		If encryption is to be used, this should be configured	
V2.17	1	Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent in clear text to the user.	Fail		Password is sent in clear text.	
V2.18	1	Verify that information enumeration is not possible via login, password reset, or forgot account functionality.	Pass			
V2.19	1	Verify that there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").	Pass		There is an 8-char random password upon installation.	
V2.20	1	Verify that request throttling is in place to prevent automated attacks against common authentication attacks such as brute force attacks or denial of service attacks.	Fail		There are no limits on the amount of passwords you can try. There is no time-out	
V2.21	2	Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location.				
V2.22	1	Verify that forgotten password and other recovery paths use a soft token, mobile push, or an offline recovery mechanism.	Fail			
V2.23	2	Verify that account lockout is divided into soft and hard lock status, and these are not mutually exclusive. If an account is temporarily soft locked out due to a brute force attack, this should not reset the hard lock status.	Fail		It is possible to manually lock an account. This is the only locking option.	
V2.24	1	Verify that if knowledge based questions (also known as "secret questions") are required, the questions should be strong enough to protect the application.	Fail		There are no security questions.	
V2.25	2	Verify that the system can be configured to disallow the use of a configurable number of previous passwords.	Fail		This is not possible.	
V2.26	2	Verify re-authentication, step up or adaptive authentication, two factor authentication, or transaction signing is required before any application-specific sensitive operations are permitted as per the risk profile of the application.	Fail		There is only one authentication factor, a password	
V2.27	1	Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases.	Fail		There are no requirements on passwords.	
V2.30	1	Verify that if an application allows users to authenticate, they use a proven secure authentication mechanism.	Pass			
V2.31	2	Verify that if an application allows users to authenticate, they can authenticate using two-factor authentication or other strong authentication, or any similar scheme that provides protection against username + password disclosure.	Fail			
V2.32	1	Verify that administrative interfaces are not accessible to untrusted parties	Fail		All users can access all administrative panes, no matter their roles. They can even change administrator's passwords.	

#	ASVS Level	Verification Requirement	Verdict	Source Code Referen	Comment	Tool Use
V3.1	1	Verify that there is no custom session manager, or that the custom session manager is resistant against all common session management attacks.	Trivial Pass (N/A)			
V3.2	1	Verify that sessions are invalidated when the user logs out.	Trivial Pass (N/A)			
V3.3	1	Verify that sessions timeout after a specified period of inactivity.	Pass			
V3.5	1	Verify that all pages that require authentication have easy and visible access to logout functionality.	Pass			
V3.6	1	Verify that the session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.	Fail	\system\classes\session.php		
V3.7	1	Verify that all successful authentication and re-authentication generates a new session and session id.	Trivial Pass (N/A)			
V3.10	2	Verify that only session ids generated by the application framework are recognized as active by the application.	Trivial Pass (N/A)			
V3.11	1	Verify that session ids are sufficiently long, random and unique across the correct active session base.	Trivial Pass (N/A)			
V3.12	1	Verify that session ids stored in cookies have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the “HttpOnly” and “secure” attributes	Fail		This is not done by PHP by default	
V3.16	1	Verify that the application limits the number of active concurrent sessions.	Fail	\system\classes\session.php	There is no limit.	
V3.17	1	Verify that an active session list is displayed in the account profile or similar of each user. The user should be able to terminate any active session.	Fail		There is no list of current active sessions.	
V3.18	1	Verify the user is prompted with the option to terminate all other active sessions after a successful change password process.	Fail		Not such an option.	

#	ASVS Level	Verification Requirement	Verdict	Source Code Referen	Comment	Tool Use
V4.1	1	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege.	Fail		Users can escalate their own privileges to administrator level.	
V4.4	1	Verify that access to sensitive records is protected, such that only authorized objects or data is accessible to each user (for example, protect against users tampering with a parameter to see or alter another user's account).	Fail		Users can escalate their own privileges to administrator level.	
V4.5	1	Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.	NA - check is beyond scope of system		This is in the web server configuration.	
V4.8	1	Verify that access controls fail securely.	Pass			
V4.9	1	Verify that the same access control rules implied by the presentation layer are enforced on the server side.	Pass			
V4.10	2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	Fail	.\system\functions\users.php	Users can escalate their own privileges to administrator level.	
V4.12	2	Verify that all access control decisions can be logged and all failed decisions are logged.	Pass			
V4.13	1	Verify that the application or framework uses strong random anti-CSRF tokens or has another transaction protection mechanism.	Fail		There is no anti-CSRF protection.	
V4.14	2	Verify the system can protect against aggregate or continuous access of secured functions, resources, or data. For example, consider the use of a resource governor to limit the number of edits per hour or to prevent the entire database from being scraped by an individual user.	Fail	.\system\functions\posts.php	There are no limits.	
V4.15	2	Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.	Fail		Users can escalate their own privileges to administrator level.	
V4.16	1	Verify that the application correctly enforces context-sensitive authorisation so as to not allow unauthorised manipulation by means of parameter tampering.	Fail		Users can escalate their own privileges to administrator level.	

#	ASVS Level	Verification Requirement	Verdict	Source Code Reference	Comment	Tool Use
V5.1	1	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.	Trivial Pass (N/A)		PHP has built-in protection against buffer overflows.	
V5.3	1	Verify that server side input validation failures result in request rejection and are logged.	Fail		Validation errors aren't logged.	
V5.5	1	Verify that input validation routines are enforced on the server side.	Pass			
V5.10	1	Verify that all SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected by the use of prepared statements or query parameterization, and thus not susceptible to SQL injection.	Don't know - other		Some parts of queries are fed with parameters of unclear origin.	
V5.11	1	Verify that the application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection.	Trivial Pass (N/A)		No LDAP.	
V5.12	1	Verify that the application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.	Pass		Looking for results by using grep for exec, shell_exec and ` (backtick operator) do not give results (t	Other
V5.13	1	Verify that the application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.	Pass		Should be OK. There is the possibility that there is one LFI and RFI in the autoloader. (grep)	Other
V5.14	1	Verify that the application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.	Trivial Pass (N/A)		No usage of XML (grep)	Other
V5.15	1	Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilize templates that automatically encode contextually to ensure the application is not susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks.	Fail	login.php	The data is included in dynamic content that is sent to a web user without being validated.	Foritfy
V5.16	2	If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the inbound request to a model, verify that security sensitive fields such as “accountBalance”, “role”or “password”are protected from malicious automatic binding.	Don't know - no time to look into this		Unclear at the moment. See Template::render. This uses extract, which comes from a parameter. (grep)	
V5.17	2	Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, environment, etc.)	Fail	/system/admin/theme/error_php.ph	Outputs the User-Agent header directly without checking. (grep)	Other
V5.18	2	Verify that client side validation is used as a second line of defense, in addition to server side validation.	Don't know - no time to look into this			
V5.19	2	Verify that all input data is validated, not only HTML form fields but all sources of input such as REST calls, query parameters, HTTP headers, cookies, batch files, RSS feeds, etc; using positive validation (whitelisting), then lesser forms of validation such as greylisting (eliminating known bad strings), or rejecting bad inputs (blacklisting).	Fail	/system/admin/theme/error_php.ph	Outputs the User-Agent header directly without checking. (grep)	Other
V5.20	2	Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as validating suburbs and zip or post codes match).	Fail	\system\admin\controllers\metadata	Can input text in number of posts per page without an exeption.	
V5.21	2	Verify that unstructured data is sanitized to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped (e.g. natural names with Unicode or apostrophes, such as ねこ or O'Hara)	Fail	\system\admin\controllers\metadata	Can input special characters.	
V5.22	1	Make sure untrusted HTML from WYSIWYG editors or similar are properly sanitized with an HTML sanitizer and handle it appropriately according to the input validation task and encoding task.	Fail		Admins, editors and other users can put HTML in text forms, including XSS.	
V5.23	2	For auto-escaping template technology, if UI escaping is disabled, ensure that HTML sanitization is enabled instead.	Don't know - not clear how to check this			
V5.24	2	Verify that data transferred from one DOM context to another, uses safe JavaScript methods, such as using innerText and .val.	Pass			
V5.25	2	Verify when parsing JSON in browsers, that JSON.parse is used to parse JSON on the client. Do not use eval() to parse JSON on the client.	Pass		With the assumption that jQuery and MooTools do this correctly. Because they have fallbacks to eva	Other
V5.26	2	Verify that authenticated data is cleared from client storage, such as the browser DOM, after the session is terminated.	Pass			

#	ASVS Level	Verification Requirement	Verdict	Source Code Reference	Comment	Tool Use
V7.2	1	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding.	Trivial Pass (N/A)			
V7.6	2	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be not guessable by an attacker.	Fail	installer.php	mt_rand() is not cryptographically secure.	Foritfy
V7.7	1	Verify that cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard.	Fail	users.php	crypt() is used. This is proven to be unsafe	Foritfy
V7.9	2	Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired). Verify that this key lifecycle is properly enforced.	Trivial Pass (N/A)			
V7.12	2	Personally Identifiable Information should be stored encrypted at rest and ensure that communication goes via protected channels.	Trivial Pass (N/A)			
V7.13	2	Verify that where possible, keys and secrets are zeroed when destroyed.	Trivial Pass (N/A)			
V7.14	2	Verify that all keys and passwords are replaceable, and are generated or replaced at installation time.	Trivial Pass (N/A)			

#	ASVS Level	Verification Requirement	Verdict	Source Code Reference	Comment	Tool Used
V8.1	1	Verify that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id, software/framework versions and personal information	Fail	/system/admin/theme/error_php.ph	Shows stack traces for admin exceptions.	
V8.2	2	Verify that error handling logic in security controls denies access by default.	Pass			
V8.3	2	Verify security logging controls provide the ability to log success and particularly failure events that are identified as security-relevant.	Pass			
V8.4	2	Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens.	Fail		Only a basic message is logged.	
V8.6	2	Verify that security logs are protected from unauthorized access and modification.	Fail	/system/classes/log.php	By default, logs are world-readable, may be configured to not be world-readable.	
V8.7	2	Verify that the application does not log sensitive data as defined under local privacy laws or regulations, organizational sensitive data as defined by a risk assessment, or sensitive authentication data that could assist an attacker, including user's session identifiers, passwords, hashes, or API tokens.	Pass			
V8.10	2	Verify that an audit log or similar allows for non-repudiation of key transactions.	Fail		No audit log to be found.	

#	ASVS Level	Verification Requirement	Verdict	Source Code Referen	Comment	Tool Used
V9.1	1	Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.	Fail	login.php	One could use the computer after the initial user to see information previously submitted.	Foritfy
V9.3	1	Verify that all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data).	Pass			
V9.4	1	Verify that the application sets appropriate anti-caching headers as per the risk of the application, such as the following: Expires: Tue, 03 Jul 2001 06:00:00 GMT Last-Modified: {now} GMT Cache-Control: no-store, no-cache, must-revalidate, max-age=0 Cache-Control: post-check=0, pre-check=0 Pragma: no-cache	Pass			
V9.5	2	Verify that on the server, all cached or temporary copies of sensitive data stored are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.	Trivial Pass (N/A)		There is no caching.	
V9.7	2	Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.	Pass			
V9.9	1	Verify that data stored in client side storage - such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies - does not contain sensitive or PII).	Pass			
V9.10	2	Verify accessing sensitive data is logged, if the data is collected under relevant data protection directives or where logging of accesses is required.	Trivial Pass (N/A)		There is no sensitive data, except for passwords, which cannot be accessed (because are hashed).	
V9.11	2	Verify that sensitive data is rapidly sanitized from memory as soon as it is no longer needed and handled in accordance to functions and techniques supported by the framework/library/operating system.	Trivial Pass (N/A)		There is no sensitive data, except for passwords, which are hashed as soon as possible.	

#	ASVS Level	Verification Requirement	Verdict	Source Code Referen	Commen	Tool Used
V10.1	1	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid.	NA - check is beyond scope of system			
V10.3	1	Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions, and does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm.	NA - check is beyond scope of system			
V10.6	2	Verify that all connections to external systems that involve sensitive information or functions are authenticated.	NA - check is beyond scope of system			
V10.11	1	Verify that HTTP Strict Transport Security headers are included on all requests and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains	NA - check is beyond scope of system			
V10.13	1	Ensure forward secrecy ciphers are in use to mitigate passive attackers recording traffic.	NA - check is beyond scope of system			
V10.14	1	Verify that proper certification revocation, such as Online Certificate Status Protocol (OSCP) Stapling, is enabled and configured.	NA - check is beyond scope of system			
V10.15	1	Verify that only strong algorithms, ciphers, and protocols are used, through all the certificate hierarchy, including root and intermediary certificates of your selected certifying authority.	NA - check is beyond scope of system			
V10.16	1	Verify that the TLS settings are in line with current leading practice, particularly as common configurations, ciphers, and algorithms become insecure.	NA - check is beyond scope of system			