NWI-ISOFSE - Software Security

Project 2, 2018 January 11th

Alan Ribeiro Andrade	s1002082	a.ribeiroandrade@student.ru.nl
Brigel Pineti	s1005549	b.pineti@student.ru.nl
Christoph Schmidl	s4226887	c.schmidl@student.ru.nl
Thomas Churchman	s4206606	thomas.churchman@student.ru.nl

1. Organisation

Division of Work

The Level 1 security requirements were divided equally in a round-robin style. All Level 1 requirements across all categories were pooled, to ensure team members would contribute to almost all categories of requirements. The exceptions are *Cryptography at Rest* and *Error Handling and Logging*, as these categories have relatively few Level 1 requirements.

Level 2 requirements were not initially divided, as it was not clear if the requirements could be verified in appropriate time.

Quality Control

Considering not all the members in the team have a solid security or PHP background in some of the requirements, the most experienced members helped verify all requirements.

Static Analysis Tools

Static code analysis using RATS and RIPS provided an overview of what was to be expected when analyzing the code in depth, such as SQL injection and XSS vulnerabilities. In addition, many members of the team used simple tools such as *grep* to find pieces of code or key-words related to the security requirement under investigation.

Running the Application

Christoph Schmidl made the application available for all group members here. The application is running on an Ubuntu environment with the Apache web server, including SSL certificates. Additionally, Thomas Churchman ran the application locally and experimented with the source code to observe its impact on the application’s behavior. The main advantage in running the application is the ability to gain a better understanding of how the system works, making it easier to understand the code.

2. Verdict

Authentication

V2.1 Fail

Alan Andrade

Among other pages and resources, the file "test.sql" file is available without requiring any authentication, revealing the database structure, which can be very useful for attackers.

V2.2 Pass

Brigel Pineti

In order to verify that forms containing credentials are not prefilled, the linux command grep was used to filter line by line through files containing sensitive input information like username, password and email. Consequently, no occurrences of prefilling by the application were found.

V2.4 Pass

Christoph Schmidl

All authentication controls found in ./system/classes/users.php and ./system/admin/theme/users/login.php files are comparing the user's input against the given database. No client-side checks have been found which means there is no hard coded string on the client side which could be exploited to get access. Therefore I assume that this requirement is passed.

V2.6 Pass

René den Hertog

The only authentication control found using the script below does only succeed if the correct username and password combination is supplied. If one of these fields is empty or incorrect, the procedure fails. It is reasonable to conclude this behaviour prevents attackers from logging in. Some confusion arises from 'fail securely'. What makes an authentication attempt fail "securely"? We understood it as simply rejecting to log in & returning to the login page with an appropriate error message. A script using find and grep for "login|authenticate|password", helped focus the search through the source code.

V2.7 Pass

Thomas Churchman

The password entry fields do not place any limitations on the length of the passwords.

- ./system/admin/theme/users/add.php : 66
- ./system/admin/theme/users/edit.php : 66
- ./system/admin/theme/users/login.php : 17
- ./system/admin/theme/users/reset.php : 14

V2.8 Fail

Thomas Churchman

The password reset mechanism authenticates requests using an MD5 hash of the concatenated strings of the user's ID, email address and hash of the current (lost) password. If this MD5 digest string is correct for any user in the database, the request is authenticated to change the password of that user. As this MD5 hash is used on its own (i.e., no other information such as the username is required), an attacker need only try many different MD5s to access an account, rather than guess both usernames and passwords. There are 2^128 possible MD5 digests. This is many orders of magnitudes easier than guessing both usernames and passwords, for strong passwords.

- ./system/classes/users.php : 113
- ./system/admin/controllers/users.php : 39

V2.9 Fail

Brigel Pineti

The requirement is not met at ./system/classes/users.php:129-150. The website does not required the user to enter the old password or to retype the new password for confirmation. The functionality to change the password will only require the user to enter the new password. It is crucial for the application to require reauthentication. Why? Well if attacker could hijack another user session, reauthentication will prevent the attacker from changing credentials.

V2.12 Fail

Thomas Churchman

No authentication audit / log is implemented.

V2.13 Fail

Thomas Churchman

No salt is used in hashing passwords. The password hash work factor is not clearly set in the application.

See also verification requirement 7.7.

- ./install/installer.php : 122
- ./system/classes/users.php : 73, 142, 190, 247,

V2.16 NA - check is beyond scope of code

Christoph Schmidl

Although we have SSL/TLS activated in our test system, this requirement is not implemented in the code base but has to rely on ssl/tls activation in the given webserver. Our apache server configuration forbids any redirects from https to http and therefore the channel would be encrypted and secure. Nevertheless, this requirement check is beyond the scope of the code base.

V2.17 Pass

René den Hertog

In '[...]/index.php/admin/users/amnesia' users can enter an email address. If the given email address is valid and equal to the one registered with an user, an email is sent to this address with a link which contains this user's hash (MD5 of this user's identifier, email & current password) necessary to access '[...]/index.php/admin/users/reset/[user's hash]' where the new password for this user can be entered.

In './system/admin/theme/users/reset.php : 14' the password input value attribute contains the 'password' value of POST. Currently this can not return the entered new password in plain text, because the form is only returned to if the password field is empty. Nonetheless, if the behaviour of the user class reset_password function changes, this might occur.

A script using find and grep for "password|amnesia|recover_password|reset|reset_password", helped focus the search through the source code.

V2.18 Fail

Thomas Churchman

On the 'Recover Password' page, an attacker can harvest email addresses. An email address not registered will generate an error, while a known email generates a success message. This allows an attacker to first guess an email address, and then guess a password for the given user. This is many orders of magnitude easier than having to guess both at the same time. (Luckily the application only allows logging using the **username**, slightly reducing the security risk.)

More prudent would be to say "an email with password reset instructions will be sent if an account has been registered with that email address."

Some other parts of the system are fine. For example when logging in, an erroneous username or erroneous password will yield the same error message.

- ./system/classes/users.php : 74, 77, 104

V2.19 Pass

Alan Andrade

The system generates a random 8 digit long password including numbers, lowercase and uppercase letters.

V2.20 Fail

Brigel Pineti

The requirement is not met at ./system/classes/testcms.php:63. No automated tools like CAPTCHA or TARPIT methods were used to limit the number of tries. However, limitations on the number of tries can be used as a denial-of-service attack to prevent the admin from logging in.

V2.21 Trivial Pass

Thomas Churchman

The system has no services that are external to the application.

V2.22 Pass

Christoph Schmidl

There is no offline recovery mechanism in the recover_password function in the ./system/classes/users.php file on lines 96 - 127. The user receives a link via Email after she entered her corresponding email address to change her password. The link contains a random hash which makes it a weak solution. The requirement says that a random value in an email should be a last resort but would still be sufficient to pass this test.

V2.23 NA - check is beyond scope of system

Thomas Churchman

The system has no account lock-out implementation.

V2.24 Trivial Pass

René den Hertog

A script using find and grep for "question|secret|answer", did not reveal any source code which contains "secret questions" functionality. Hence, we reason that the system does not use "secret questions". Ergo, the requirement does not apply to the system and is trivially passed.

V2.25 Fail

Thomas Churchman

The system does not implement such functionality. Almost no checks on passwords are performed: the only requirement checked by the application is that passwords must be of length 1 or more.

V2.26 Fail

Thomas Churchman

User management might be a high-value transaction in scope of this system. No re-authentication is required for this action.

V2.27 Fail

Thomas Churchman

No measures are in place. In fact, the system even allows passwords of length 1.

- ./system/classes/users.php : 129
- ./system/classes/users.php : 160
- ./system/classes/users.php : 217

V2.30 Pass

Alan Andrade

Thomas Churchman

The application requires a combination of username and password for authentication. This is a standard authentication mechanism that is secure if handled correctly. However, see e.g. verification requirement 2.8, indicating the application's implementation of this verification mechanism is flawed and insecure.

- ./system/classes/users.php : 53-90

V2.31 Fail

Thomas Churchman

No such 2FA scheme is in place.

- ./system/classes/users.php : 53

V2.32 Fail

Brigel Pineti

Thomas Churchman

The requirement is not met at ./system/admin/controllers/users.php : 62-77. The administrative interface (e.g. user management) is available to all authenticated users, regardless of their role (user/editor/administrator).

- ./system/admin/controllers/users.php : 62-77

V2.33 Don't know - not clear how to check this

Christoph Schmidl

Although the autocomplete feature is not explicitly turned off, I do not know how to test the risk based policy. I would assume that this requirement passes but I'm not sure. Following files have been tested for the autocomplete flag:

- ./system/admin/theme/users/add.php
- ./system/admin/theme/users/edit.php
- ./system/admin/theme/users/login.php
- ./system/admin/theme/users/reset.php

Session Management

V3.1 Fail

René den Hertog
Thomas Churchman

The application defines a Session class, which currently uses default session management functions of PHP. However, if in later versions of the system the session class becomes custom, then this custom session management should be verified to be 'resistant against common session management attacks'. In any case, the application currently fails this requirement. One issue is Session ID Name Fingerprinting: the session is exposed as PHPSESSID, giving the attacker knowledge of the underlying session system. A more serious issue is that of session fixation: the application does not generate a new session ID when logged in. If an attacker manages to induce a specific session ID on a client (i.e. a session the attacker started), and the user logs in, the attacker now shares that a valid session. A script using find and grep for "session", helped focus the search through the source code.

find [gnu.org/software/findutils]
grep [gnu.org/software/grep]

- ./system/classes/session.php
- ./system/classes/curl.php

V3.2 Pass

Thomas Churchman

If a user logs out, the session is not forgotten on the client side -- instead, the user is removed from the session on the server side (i.e. the session is practically invalidated regarding authentication).

- ./system/classes/users.php : 93
- ./system/classes/session.php : 30

V3.3 Pass

Alan Andrade

By default the sessions timeout after 60 minutes.

V3.5 Pass

Brigel Pineti

The grep command was used to filter through pages that have authentication. No missing logout links were found.

V3.6 Pass

Christoph Schmidl

The system does not log by default. URL rewriting of PHP's default session cookies is disabled by default on current versions of PHP. The sessionid is transported/saved by cookie with the PHPSESSID flag and not over query strings in URLs. See ./config.default.php on line 43.

V3.7 Fail

René den Hertog
Thomas Churchman

In './system/classes/users.php' @ the login function (: 87), the authenticated user is set to the current session. No new session is started after successful login. At the update function (: 207-210), the updated user is set to the current session if the current session contains an user & its identifier matches the identifier of the updated user. No new session is started after the successful update of users' information. One can argue that technically speaking no re-authentication takes place during the update of user information. However, since it is possible to change the users password via this method, re-authentication should take place & a new session should be started. To fulfill this requirement, session_regenerate_id() should be called each request where the session contains an authenticated user. A script using find and grep for "login|authenticate|session", helped focus the search through the source code.

find [gnu.org/software/findutils]
grep [gnu.org/software/grep]

- ./system/classes/users.php : 53-90, 87, 160-215, 207-210

V3.10 Pass

Thomas Churchman

The application uses PHP's sessions. Only session IDs known by the server carry data, and they are all generated by the application (as long as there are no other applications running on this PHP instance).

V3.11 Pass

Thomas Churchman

PHP's session manager handles this correctly by default.

V3.12 Fail

Alan Andrade
Thomas Churchman

"HttpOnly" and "secure" keys were not used for cookies. "HttpOnly" for PHP sessions depends on configuration key session.cookie_httponly, but is off by default.

- /system/classes/cookie.php : 19

V3.16 Fail

Brigel Pineti

The application is not monitoring the number of active concurrent sessions. We could not spot any actions that store the users IP address, session id and user id.

V3.17 Fail

Christoph Schmidl

The application is not providing any functionality similar to active session listing displayed in the account profile or similar of each user.

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.

V3.18 Fail

René den Hertog

No option to terminate all other active sessions after a successful password change is available. (The application implements an authentication session as simply placing the user directly in PHP's session object.) A script using find and grep for "password|amnesia|recover_password|reset|reset_password|session|update|edit", helped focus the search through the source code.

find [gnu.org/software/findutils]
grep [gnu.org/software/grep]

- ./system/admin/controllers/users.php : 28-51, 62-77
- ./system/admin/theme/users/edit.php(: 64-69)
- ./system/admin/theme/users/reset.php
- ./system/admin/theme/users/amnesia.php
- ./system/classes/users.php : 96-150, 160-215(, 188-194)

Access Control

V4.1 Fail

Thomas Churchman

The application implements a role system with administrators, editors and users (good!), but it does not actually use the role system for authorization.

E.g. one would expect a user to have read authorization for posts, an editor to have write authorization for posts, and an administrator to have additional authorization for user management. However, even a normal user can manage users (and create administrator users). Very bad!

- ./install/test.sql : 78
- ./system/classes/users.php : 218

V4.4 Fail

Alan Andrade

Thomas Churchman

The system uses different roles such as admin, user and editor, but all these roles have unrestricted access to all information available in the system.

See also verification requirement 4.1.

V4.5 NA - check is beyond scope of code

Brigel Pineti

Thomas Churchman

This is a matter of webserver configuration.

V4.8 Pass

Christoph Schmidl

The login, logout, amnesia and reset functions in the ./system/admin/controllers/users.php file are using the response class and its redirect function to redirect the user to the login page if the request fails. I would assume that this means that access control "fails" securely.

V4.9 Pass

Thomas Churchman

The only form of access control in the system (based on whether the user is logged in) is implemented on the server side.

V4.10 Fail

Thomas Churchman

Roles are implemented, but not used for authorization. Any authenticated user can change everything in the application -- regardless of their intended access level.

See also verification requirement 4.1.

V4.12 Fail

Thomas Churchman

The application implements no audit log.

V4.13 Fail

Thomas Churchman

No CSRF tokens are used for post requests anywhere in the application.
For example:

- ./install/installer.php : 21
- ./system/admin/controllers/user.php : 15
- ./system/classes/posts.php : 266

V4.14 NA - check is beyond scope of system

Thomas Churchman

No such system is in place, but the impact on this application is likely minimal.

V4.15 NA - check is beyond scope of system

Thomas Churchman

No such authorization scheme is in place. It is arguably out of scope for this system.

V4.16 Fail

Alan Andrade

Thomas Churchman

Though authorisation is implied through roles, no authorisation checks are performed (and thus no context-sensitive authorisation checks are performed).

See also verification requirement 4.1.

Malicious Input Control

V5.1 Pass

Thomas Churchman

The application is implemented in PHP, which is not susceptible to buffer overflows.

V5.3 Fail

Christoph Schmidl

Although input validation failures result in a redirect and an error message, the system does not log and therefore this requirement fails. Also see the config file on line 43: config.default.php.

V5.5 Fail

René den Hertog

Reading the report generated by the analysis of RIPS, in section 3.2.1 issue 486078 an actual SQL injection vulnerability of the system under test is described. Validation of user input used in SQL commands is also a server side responsibility, as is described in this verification requirement. In the case just described, this responsibility is failed to be taken. Hence, the system under test does not always enforce input validation at the server side. The rest of the system is not tested further for input validation on the server side. Sadly, we do not know a method to effectively verify this requirement more extensively. Nonetheless, the one flaw has already been found as described above, which is enough to verdict this verification requirement as failed.

V5.10 Pass

Thomas Churchman

The system uses prepared statements everywhere except for the installer. The installer is vulnerable to SQL injection, but as long as the installer is removed (as is suggested by the install page) this is not a security vulnerability but a matter of configuration.

- ./system/classes/db.php : 78, 124
- ./install/installer.php : 121 - 123

V5.11 Trivial Pass (N/A)

Alan Andrade

The application does not use LDAP to store any data.

V5.12 Pass

Brigel Pineti

Thomas Churchman

The application does not call OS commands with user data at any point.

V5.13 Pass

Christoph Schmidl

RFI and LFI are mostly only feasible when the require, require_once, include, include_once or file_get_contents PHP functions are used in conjunction with a filename or file path which can be altered by the users input. After performing a full-text search over the whole codebase and searching for this combination, no possible combination which would be accessible to a user has been found.

The only way that RFI or LFI would work is if one would activate SSI (Server Side Includes) and work with include directives while writing comments to a blog post. Such include directives would go unescaped and therefore would work to perform such an attack but the application itself is not vulnerable if SSI is inactive.

V5.14 Pass

René den Hertog

Thomas Churchman

The system does not use XML as input, but it outputs an RSS feed of the posts. The dynamic output is suitably cleansed through PHP's **DOMDocument** to ensure no common vulnerabilities in RSS readers of the site's feed can be exploited.

V5.15 Fail

Thomas Churchman

The blog allows any user to place a comment. On display, the html in comments is not escaped and is sent to the browser unfiltered. As such, anyone can easily perform XSS.

There are other places where user input is sent unfiltered, such as the site description and title, but at the moment all this input is under control of authenticated users.

- ./system/functions/comments.php : 81-87

V5.16 Trivial Pass (N/A)

Thomas Churchman

No such mass parameter assignment is available.

V5.17 Pass

Thomas Churchman

The application only uses POST parameters to mutate data.

V5.18 Fail

Thomas Churchman

The application cleans up "post slugs" (URLs) on the client side, but the server side does not enforce the same clean-up. By removing the client side clean-up or constructing a post request manually, the slug can be set to any value desired.

- ./system/admin/theme/posts/add.php : 126 - 136
- ./system/classes/posts.php : 266 - 324

V5.19 Fail

Thomas Churchman

The system performs poor input validation.

See also verification requirement 5.18.

V5.20 NA - check is beyond scope of system

Thomas Churchman

This check is arguably out of scope of the system. Minor validation is performed of e-mail address formats.

See also verification requirements 5.18 and 5.19.

V5.21 Fail

Thomas Churchman

The system performs no data sanitization. For example, it is possible to directly influence the webpage DOM by placing a comment on a post, for which no authentication is required. See also verification requirement 5.15.

V5.22 Fail

Alan Andrade

Multiple sections do not use HTML sanitizer and allow malicious code to be executed.

V5.23 Trivial Pass (N/A)

Thomas Churchman

The application does not use template technology.

V5.24 Pass

Thomas Churchman

The application uses jQuery for DOM manipulation, which uses secure methods under-the-hood.

- ./install/assets/js/app.js : 84, 87
- ./themes/default/js/main.js : 29

V5.25 Pass

Thomas Churchman

The application utilizes jQuery to perform JSON serialization / deserialization safely.

- ./install/assets/js/app.js : 22-29, 45-51
- ./system/admin/theme/assets/js/comments.js : 10-22, 81-118, 126-136

V5.26 Trivial Pass (N/A)

Thomas Churchman

The application does not store authenticated data in client-side storage.

Cryptography at Rest

V7.2 Fail

René den Hertog

Analyzing the source code using the script below, 5 cryptographic functions are used throughout the PHP source code.

> '**random**' : A custom function for generating random passwords. The documentation is not rich enough to determine if failures/errors lead to issues.
> '**mt_rand**' : According to the documentation [php.net/manual/en/function.mt-rand.php], the uses of this function should not lead to failures/errors.
> '**md5**' : According to the documentation [php.net/manual/en/function.md5.php], the uses of this function should not lead to failures/errors.
> '**crypt**' : According to the documentation [php.net/manual/en/function.crypt.php], using this function without a salt not only results in the raising of an 'E_NOTICE' error (a potential leakage of information), but also could lead to unexpected results. Many uses of 'crypt' do not supply a salt, meaning that it is possible that the calls to this function lead to unsecure failures/errors.
> '**hash**' : According to the documentation [php.net/manual/en/function.hash.php], the uses of this function should not lead to failures/errors.

Regarding the verification of oracle padding resistance, the documentation of the cryptographic functions found being used did not contain any information in relation with oracle padding. Either this is not considered, or the functions are implemented in such a way that they do not enable it possible (or its effect does not apply to them). A script using find and grep for "crypt|hash|random|sodium|password|openssl|crack"*, helped focus the search through the source code. From this analysis the usage of the above 5 cryptographic functions were found. The regular expression pattern "random\(|mt_rand\(|md5\(|crypt\(|hash\(" leads to a more focused search.

find [gnu.org/software/findutils]
grep [gnu.org/software/grep]

V7.6 Fail

Thomas Churchman

Random password generator uses 'random', which is not a cryptographically secure random number generator.

- ./install/installer.php : 119

V7.7 Fail

Thomas Churchman

The application uses crypt, which does not generate cryptographically secure password hashes by default. At the very least when using crypt a salt should be set manually. The application does not do this. An example attack this opens up is the possibility of using rainbow tables if the hashes are leaked. Furthermore, crypt does not always generate passwords with a suitable cost (hash iterations), making reversing password hashes more feasible.

It is recommended to use password_hash (and password_verify), which generates cryptographically secure password hashes by default.

- ./install/installer.php : 122
- ./system/classes/users.php : 73, 142, 190, 247,

V7.9 Trivial Pass (N/A)

Thomas Churchman

The application does not manage cryptograhic keys.

V7.12 Trivial Pass (N/A)

Thomas Churchman

Arguably the application does not store sensitive personal identifiable information; only usernames and email addresses are stored.

V7.13 Fail

Thomas Churchman

The system does not do this (and is incapable of doing this, as it uses PHP).

See also verification requirement 9.11.

V7.14 Pass

Thomas Churchman

The password for the administrator account is generated at installation time. The authentication details for the database are configured at installation time. No other keys/passwords are used in the application.

- ./install/installer.php : 82-93, 119

Error Handling and Logging

V8.1 Fail

Brigel Pineti
Thomas Churchman

The application does not set PHP's logging level explicitly. As such, if the application encounters problems such as SQL errors, they will be sent to the client. This can be overcome by using a "production ready" PHP configuration, which by default does not show warnings and errors to the client.

V8.2 Pass

Thomas Churchman

When logging in, any exception raised ensures the user is not logged in, as the exception goes unhandled in the login functionality. Similarly, any non-exception error recorded ensures the user is not logged in: as the system requires there to be no recorded errors before an authenticated session is set.

- ./system/classes/users.php : 53-90

V8.3 Fail

Thomas Churchman

The system has no audit / log of authentication or other security control events.

V8.4 Trivial Pass (N/A)

Thomas Churchman

The system does not log.

V8.6 Trivial Pass (N/A)

Thomas Churchman

The system does not log.

V8.7 Trivial Pass (N/A)

Thomas Churchman

The system does not log.

V8.10 Fail

Thomas Churchman

The system does not log sensitive transactions.

V8.13 Fail

Christoph Schmidl

The system does not log in its default state but there is also no sight of the usage of the Network Time Protocol (NTP), a dedicated network time server or any other mechanism to synchronize time sources. It seems that all date and time specific functions rely on their underlying native PHP functions. Also see: ./system/classes/log.php, line 12.

Data Protection

V9.1 Fail

Alan Andrade

Autocomplete feature is NOT explicitly turned off in all input forms. Critical forms are ignored, such as the login input fields. Besides that, cache-control is not explicitly defined.

V9.3 Pass

Brigel Pineti
Thomas Churchman

All sensitive data is transferred via POST requests. No GET-based forms are used for the submission of data. Doing that will cause the data to be encoded in the RequestURI, being easily visible to third parties.

V9.4 Fail

Christoph Schmidl

The response contains the following subset of headers which are required for the verification of this requirement:

Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

Based on the description in the section "Web Content Caching" in https://www.owasp.org/index.php/Session_Management_Cheat_Sheet, these headers should be sufficient to pass this requirement. After performing a fulltext search over the codebase, I could not find any explicit declaration of these headers though. Therefore I assume that these headers are default values set by our Apache Webserver and not set by the application. Therefore this requirement fails.

V9.5 Trivial Pass (N/A)

Thomas Churchman

The system does not cache or store temporary copies of sensitive data.

V9.7 Don't know - not clear how to check this

Thomas Churchman

This requirement is difficult to verify. In my opinion the system does not use an excessive number of parameters throughout the codebase. E.g., in general there are no hidden form fields (though a hidden csrf field would be a very welcome addition).

V9.9 Pass

René den Hertog

Thomas Churchman

The only sensitive data used by the application is stored in the session. Only the session identifier is stored in client-side storage.

V9.10 Trivial Pass (N/A)

Thomas Churchman

The system does not collect data for which data protection directives apply.

V9.11 Fail

Thomas Churchman

The system does not do this (and is incapable of doing this, as it uses PHP).

See also verification requirement 7.13.

HTTP Security Configuration

V11.1 Fail

Thomas Churchman

Testing with Chrome extension Postman, at least the following HTTP verbs unecessary for this application are accepted:
- PUT
- PATCH
- DELETE
- COPY
- OPTIONS
- LINK
- UNLINK
- PURGE
- LOCK
- UNLOCK
- PROPFIND
- VIEW

Only the following HTTP verb is disallowed:
- HEAD

Luckily it appears the behaviour of the application is not undefined in such cases: it simply acts as if the request is GET, as it only branches on **$_POST**.

V11.2 Pass

Alan Andrade

Content-Type is defined as 'text/html; charset=UTF-8'

V11.3 Trivial Pass (N/A)

Thomas Churchman

The system does not use such schemes for any security sensitive access.

V11.4 Fail

Thomas Churchman

The system does not send X-FRAME-OPTIONS anywhere, so also not on authentication pages.

V11.5 NA - check is beyond scope of code

Brigel Pineti

Thomas Churchman

In a default Apache and PHP configuration the system leaks information about the Apache and PHP versions (see Requirement 11.8). However, this is a configuration issue not specific to this application's code.

V11.6 Fail

Christoph Schmidl

The given system does not really contain an API, the only class which is sending *'Content-Type: application/json'* is system/admin/controllers/comments.php. Because all requests and responses are encapsulated in the corresponding classes (system/classes/request.php, system/classes/response.php) and none of them is containing the mentioned fields like *X-Content-Type-Options: nosniff* and *Content-Disposition: attachment; filename="api.json"*, this requirement fails after a fulltext-search. All available routes have been visited using curl and the actual response from the server has also been inspected without any sight of the given fields. Most of the time the right content-type has been used though.

V11.7 Fail

René den Hertog

A script using find and grep for "content|security|policy"*, did not reveal any source code which contains Content Security Policy functionality. Hence, we reason that the system does not have a Content Security Policy in place. Ergo, the requirement does not pass.

V11.8 Fail

Thomas Churchman

The application only sets the Content-Type header to **text/html; charset=UTF-8**.

Testing with the Chrome extension Postman, the application (on a standard Apache 2.4 + PHP 7 installation) sends the following headers:

cache-control: no-store, no-cache, must-revalidate
connection: Keep-Alive
content-length: 2514
content-type: text/html; charset=UTF-8
date: Wed, 20 Dec 2017 02:07:48 GMT
expires: Thu, 19 Nov 1981 08:52:00 GMT
keep-alive: timeout=5, max=100
pragma: no-cache
server: Apache/2.4.27 (Win64) OpenSSL/1.0.2l PHP/7.1.9
x-powered-by: PHP/7.1.9

- ./system/classes/response.php : 81 - 101

3. Reflection

Application Security Verification Standard

The ASVS is a very useful guideline for performing security audits, especially for people who do not have a strong software security background. It makes people aware of requirements and edge cases that they would likely not think of on their own, but which they are able to check once aware. Especially useful were OWASP's *Testing for…* articles on owasp.org, providing guidance and practical information to get a better understanding on how a specific requirement can be verified.

However, some requirements were not clearly defined -- to the point were discussion was necessary to ascertain the actual requirement expectation -- or defined in a broad manner -- to the point where it becomes difficult to state with certainty a requirement is satisfied. The standard could be improved by including more details in some of the requirements (such as Requirement 2.30) to give requirement verifiers a better idea on how to check the requirement, or by splitting up requirements (such as Requirement 7.2).

Code Analysis Tools

Code analysis tools provided a basic overview on what to expect from the code. The tools gave an initial indication of where to look at and how to check for certain security vulnerabilities. Although the tools are a guide into certain directions and reveal some vulnerabilities, in general the tool only found a very small set of vulnerabilities, of which many false positives. As such, the verifier has to invest intensive research into the underlying concept of each vulnerability and verification requirement, and broaden their search based on these new insights.

Most likely these code analysis tool issues were caused by the simplicity of the tool. A tool better aware of the language of the code it is verifying is more likely to find vulnerabilities, e.g. through symbolic execution or overlaying a more complex type system such as for taint propagation.

Bottlenecks

Although OWASP provided some articles and step-by-step instructions on how to check the different requirements, not all topics were covered. Verifiers have to invest time in researching verification requirements and their ramifications on their own. Even with additional research into a specific requirement topic it sometimes was hard to tell what the requirement was asking for specifically. These obscurities caused confusion and may have been avoided if the requirements were stated more clearly.
Other requirements were hard to evaluate as the verifier had to consider the entire application as a whole, starting from pure responses evaluated in the browser to actual code reviews, and sometime checking default configurations of the web server. Although some requirements

seemed to be fulfilled inspecting responses from the web server, such as Requirement 8.13, they might not be explicitly fulfilled by the application and are simply based on default configurations of the web server.

Tightly-coupled Security Systems

The difficulties in checking some verification requirements arises mostly from tight vs. loose coupling. For example, in this case session management is not implemented by the application itself, but rather depends on the implementation given by the underlying system. As such, the security depends on whether the session management provided by a certain version of the underlying system is secure, as well as whether the system is configured securely. The application is tightly coupled to the underlying system. To check these kinds of requirements, you have to go through the whole application stack to pin down the exact position when the requirement actually fails or passes.

Organisation

We think our work division, providing all members requirements from nearly all categories, was a suitable way of making it possible for all members to gain a better understanding of the whole process of verification.However, seeing that for some of the requirements members needed help from other members, a possible improvement would be to perform the verification work in pairs: this makes it easier to handle unclarities in verification requirements and increases the chances of valid the judgements.

TestCMS and Web Development

It appears the main issue with TestCMS is that the developers attempted to create a complex application with only few members and little professional experience. Such projects are great for learning, but are generally not great when it comes to security. These projects hardly ever lead to products ready for production use -- especially regarding security.

In cryptography the adage *don't roll your own crypto* is often spoken. For small, inexperienced teams this might be extended to *don't roll your own framework*. Instead, if a team wishes to rapidly create a system ready for production, using a ready-made framework would generally better suit their needs. For example, for content management a system such as WordPress or Django would be a good choice. Such systems are open source efforts, are designed to be secure, and will receive security patches when new vulnerabilities are discovered. Of course a downside is that a security vulnerability discovered in such a platform is directly applicable to all applications based on it, but *security through obscurity* is only a protection as long as an application remains unpopular.