CakePHP Security Assessment

Mozilla

October 17, 2017 – Version 1.0

Prepared for

Gervase Markham, Mozilla
Larry E. Masters and Mark Story, CakePHP

Prepared by

Brandon Bernie
Paul Tetreau
Ryan Koppenhaver

Executive Summary

Synopsis

In the Summer of 2017, Mozilla engaged NCC Group to perform a security assessment of the CakePHP framework under the Secure Open Source track of the Mozilla Open Source Support program, which aims to help improve the security of Free and Open Source Software by funding security audits of vital projects.

CakePHP is a web application framework that provides a Model-View-Controller (MVC) architecture. It is written in PHP and licensed under the MIT license. Its home page is https://cakephp.org/, and the current version at the time of the test was 3.5.0-RC1.

NCC Group identified specific security flaws that could affect certain applications that use specific CakePHP features, as well as areas in which CakePHP's defaults are insecure.

The assessment was done by three consultants over three calendar weeks, from July 31 to August 18, 2017. The consultants reviewed the CakePHP source code and performed dynamic testing using the standard CakePHP tutorial application as well as other test applications built by the consultants.

Scope

NCC Group's evaluation included:

- The CakePHP framework: The web application framework provides "behind the scenes" elements such as the HTTP Server, Middleware, Router, MVC base classes; as well as developer APIs including Object-Relational Mapper (ORM) functionality and HTML helpers.
- cake: The CakePHP console provides command-line functionality such as a development server, route information, and an interactive REPL, as well as a framework to build custom scripts to interact with a CakePHP application.
- bake: The bake console application performs code generation based on a database schema or various command line arguments.

Additionally, the Bookmarker tutorial application[1] was used as a reference application, in conjunction with test applications generated using bake.

The assessment targeted the 3.5.0-RC1 release of CakePHP, version 1.3.7 of bake, and commit faf9e003 of `bookmarker-tutorial`.

Key Findings

The assessment did not uncover any large-scale, directly exploitable flaws in CakePHP that would affect all applications using the framework, but did expose various issues which could impact applications which use particular elements of CakePHP functionality.

In addition, NCC Group noted some areas in which writing a secure application would require developers to explicitly override defaults provided by the framework; these were reported as findings where they did not appear to be clearly documented, or where NCC Group believes them to be sufficiently dangerous as to warrant a direct call-out.

Some of the most notable issues found were:

- CakePHP provides insecure defaults for attribute accessibility in mass assignment. This could allow malicious users to modify data in an application that they should not be allowed to, such as changing the price of items in a shopping cart, or changing their own user account to be an administrator.
- Applications could be exposed to cross-site scripting vulnerabilities if untrusted URLs are passed to some of CakePHP's HTML helper methods, or if untrusted values are used in attribute keys.
- Applications using CakePHP's XML utility class could be exposed to a denial of service via an entity expansion attack which consumes a large amount of server memory relative to the request size.

Strategic Recommendations

- Support templates using a language which performs automatic HTML escaping, and make it the default. Manually escaping HTML metacharacters in templates is error-prone, and may not work correctly in corner cases such as variables interpolated inside HTML tags or scripts.
- Create consolidated documentation of security best practices. This will allow developers to quickly familiarize themselves with areas of concern, and prevent avoidable security mistakes.

[1] https://book.cakephp.org/3.0/en/tutorials-and-examples/bookmarks/intro.html

Dashboard

Target Metadata

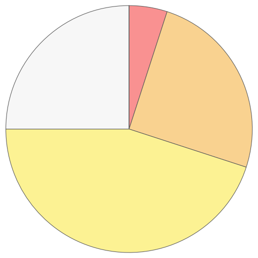
Name	CakePHP
Type	Web Development Framework
Platforms	PHP 7

Engagement Data

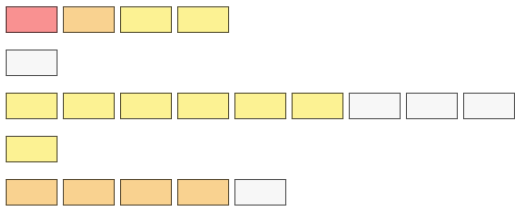
Type	Web Framework Security Assessment
Method	Source Review
Dates	2017-07-31 to 2017-08-18
Consultants	3
Level of effort	45 person-days

Finding Breakdown

Critical Risk issues	0
High Risk issues	1
Medium Risk issues	5
Low Risk issues	9
Informational issues	5
Total issues	**20**



Category Breakdown



Access Controls	4
Configuration	1
Cryptography	9
Data Exposure	1
Data Validation	5

Key

Critical 🟥 High 🟥 Medium 🟧 Low 🟨 Informational ⬜

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix A on page 31.

Title	ID	Risk
Mass Assignment Allowed by Default	001	High
Unsafe HTML Templates By Default	003	Medium
Multiple Html Helper Methods Do Not Escape URLs	004	Medium
HTML Helpers Do Not Escape Attribute Keys	005	Medium
Incorrect X-Forwarded-For Header Parsing Allows IP Spoofing	015	Medium
XML Entity Expansion Denial of Service	018	Medium
randomBytes Returns Insecure Random Numbers	009	Low
Comment Incorrectly Advertises Mitigation for User Enumeration Timing Attack	010	Low
Digest Authentication Uses Non-Constant-Time Comparisons	012	Low
Tutorial Application Allows Path Traversal in Pages Controller	014	Low
AutoLink Functions Use Predictable and Collision-Prone Hashes	016	Low
Inconsistent Configuration of Transport Encryption Protocols	019	Low
Form Validation Tokens Are Not Associated with Users	020	Low
Form Validation Tokens Are Vulnerable to Potential Hash Collisions	021	Low
Asset Middleware Can Serve Dot Files	022	Low
Setters of Entity Attributes Are Not Always Called	002	Informational
Insecure Default Hashing Algorithm	006	Informational
Certificate Authority Bundle Includes Untrustworthy CAs	007	Informational
Insecure UUID Function	008	Informational
Digest Authentication Uses MD5 Algorithm to Construct Nonces	011	Informational

Finding	**Mass Assignment Allowed by Default**
Risk	High Impact: High, Exploitability: Medium
Identifier	NCC-CakePHP_2017-001
Category	Access Controls
Location	• CakePHP: `src/Datasource/EntityTrait.php`: 118 • Bake: `src/Template/Bake/Model/entity.ctp`: 25 - 37
Impact	By default, any attribute of an entity may be modified by a request, i.e. a user updating their password would be able to set their privilege level to that of an administrator.

Description Mass assignment is often used by developers as a convenience mechanism for setting attributes of a model, but it can be a problem when a model's attributes are overexposed, allowing malicious users to unexpectedly modify sensitive model attributes, such as a user's privilege level.

Mass assignment is regulated by the `_accessible` associative array, which maps entity attributes to booleans that indicate whether or not that attribute may be assigned when the entity is updated. The ∗ character is a wildcard that matches any attribute.

The `_accessible` array in the `EntityTrait` trait defaults to allowing all attributes to be assigned:

```
protected $_accessible = ['*' => true];
```

Similarly, the Bake command defaults to allowing all attributes to be assigned except the primary key of the model:

```
$accessible = [];
if (!isset($fields) || $fields !== false) {
    if (!empty($fields)) {
        foreach ($fields as $field) {
            $accessible[$field] = 'true';
        }
    } elseif (!empty($primaryKey)) {
        $accessible['*'] = 'true';
        foreach ($primaryKey as $field) {
            $accessible[$field] = 'false';
        }
    }
}
```

For example, a user model might contain an `admin` attribute, with a value of `True` for administrators and `False` for other users, a `username` attribute, and a `password` attribute. If that entity were vulnerable to mass assignment, a password reset page that permits a user to update their password would also permit the user to modify their `admin` attribute, allowing for privilege escalation.

Mass assignment is a very dangerous default, made infamous by Ruby on Rails, whose Github repository was compromised[2] by an unauthorized user because of that default. This allowed the attacker to add their SSH keys to the trusted keys list of the repository.

Recommendation The accessible attributes list should be empty by default, requiring developers to manually

[2] https://github.com/rails/rails/commit/b83965785db1eec019edf1fc272b1aa393e6dc57

whitelist accessible attributes.

Consider throwing an exception when protected attribute assignment is attempted using the ORM, i.e. via `patchEntity`.

Finding	**Unsafe HTML Templates By Default**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-CakePHP_2017-003
Category	Data Validation
Location	Templates
Impact	Variables in templates are not escaped by default, increasing the likelihood of cross-site scripting vulnerabilities.

Description

When dereferencing variables in templates, values are not escaped by default. This creates an unsafe-by-default design where developers must explicitly call sanitizing functions to avoid being vulnerable to cross-site scripting when user-derived input is included in templates.

From the Views documentation[3]:

> *You should remember to always escape any user data before outputting it as CakePHP does not automatically escape output. You can escape user content with the h() function…*

Insecure defaults distribute the onus of secure coding across all users of a framework, increasing the likelihood of insecure applications through ignorance (not understanding insecure defaults) or accident (forgetting to use secure alternatives).

Recommendation

Safely encode variables in templates by default. Require developers to explicitly call functions with discouraging or risk-highlighting names[4] that signify that their input is not escaped.

[3] https://book.cakephp.org/3.0/en/views.html#view-variables
[4] For example: `dangerousUnsafeHTML($someVar)`

Finding	**Multiple Html Helper Methods Do Not Escape URLs**
Risk	**Medium** Impact: High, Exploitability: Low
Identifier	NCC-CakePHP_2017-004
Category	Data Validation
Location	Methods in `src/View/Helper/HtmlHelper.php`: • `image()` • `meta('icon', ...)` • `css()` • `script()`
Impact	If an application accepts a user-supplied URL, and passes it to the affected methods to generate HTML, a cross-site scripting vulnerability could result.

Description

As noted in finding NCC-CakePHP_2017-003 on the previous page, CakePHP's default templating system uses plain PHP templates, which do not perform automatic HTML-encoding of dynamic content. CakePHP does provide an `HtmlHelper` class (exposed as `$this->Html` in templates) with various methods to construct HTML around arbitrary input. In general, these methods perform HTML encoding of the content and attribute values passed into them.

However, the methods listed above accept a URL parameter and do not encode it. In these methods, the URL parameter is treated as a special case and passed through `assetUrl`, rather than `formatAttributes`, which is used in most other helper methods. This can allow for cross-site scripting attacks if a CakePHP application accepts a URL as user input and relies on the helper method to safely render it.

Reproduction Steps

Insert the following code in a CakePHP template:

```php
<?php $evil = 'x:"><script>alert(1)</script>'; ?>
<?= $this->Html->image($evil); ?>
```

Browse to the relevant page and observe the JavaScript alert.

Recommendation

Modify the affected methods to HTML-encode the URLs before writing them to the generated HTML.

Finding	**HTML Helpers Do Not Escape Attribute Keys**
Risk	**Medium** Impact: High, Exploitability: Low
Identifier	NCC-CakePHP_2017-005
Category	Data Validation
Location	• `src/View/Helper/HtmlHelper.php` • `src/View/StringTemplate.php`, 276-297
Impact	If an application accepts a user-supplied string, and passes it as an attribute key to the affected methods, a cross-site scripting vulnerability could result.

Description As noted in finding NCC-CakePHP_2017-003 on page 7, CakePHP's default templating system uses plain PHP templates, which do not perform automatic HTML-encoding of dynamic content. CakePHP does provide an `HtmlHelper` class (exposed as `$this->Html` in templates) with various methods to construct HTML around arbitrary input. In general, these methods perform HTML encoding of the content and attribute values passed into them. However, attribute keys are not escaped or sanitized.

Reproduction Steps Insert the following code in a CakePHP template:

```php
<?php
  $evil_key = "><script>alert(1)</script>";
  $options = [ $evil_key => 'some value' ];
?>
<?= $this->Html->div('myclass', 'some content', $options); ?>
```

Browse to the relevant page and observe the JavaScript alert.

Recommendation In the `_formatAttribute` method in `StringTemplate.php`, ensure that `$key` matches a regular expression such as `\A(\w|[.-])+\z`.[5]

[5]This is somewhat more restrictive than the HTML Standard (https://html.spec.whatwg.org/#attributes-2), and may be relaxed if necessary for Unicode support.

Finding	**Incorrect X-Forwarded-For Header Parsing Allows IP Spoofing**
Risk	**Medium** Impact: High, Exploitability: Low
Identifier	NCC-CakePHP_2017-015
Category	Access Controls
Location	`src/Http/ServerRequest.php`: 550 - 561
Impact	An attacker may be able to bypass access restrictions based on IP addresses.

Description

The CakePHP framework provides applications with functionality to obtain the requestor's IP address. Generally, this information comes from the `REMOTE_ADDR` environment variable, the exception being when the application is sitting behind a proxy, such as nginx.

In order to pass a client's IP address to applications through the proxy, users must set the `trustProxy` configuration variable to true. When the application is configured in this way, `clientIP()` will take the first value from the potential list of values returned in the `X-Forwarded-For` HTTP header.

This behavior can be exploited by attackers due to the way proxies append values to this header. If an attacker sets the value of the `X-Forwarded-For` header in the request they make to the vulnerable application, the proxy will append the client's real IP address to the HTTP header.

```
X-Forwarded-For: spoofed.ip, real.ip
```

In this case, when `clientIP()` is called, a regular expression is used to extract the IP address of the client, albeit incorrectly. The expression `/(?:,.*)/` in the call to `preg_replace` will match the first comma and all subsequent characters in the `X-Forwarded-For` header value, causing them to be stripped. This leaves only the first (spoofed) IP address in the header to be returned by the function. If this function is used in a security context it could allow attackers to bypass access controls.

```
/**
  * Get the IP the client is using, or says they are using.
  *
  * @return string The client IP.
  */
public function clientIp()
{
    if ($this->trustProxy && $this->getEnv('HTTP_X_FORWARDED_FOR')) {
        $ipaddr = preg_replace('/(?:,.*)/', '', $this->getEnv('HTTP_X_FORWARDED_
➜   FOR'));
    } elseif ($this->trustProxy && $this->getEnv('HTTP_CLIENT_IP')) {
        $ipaddr = $this->getEnv('HTTP_CLIENT_IP');
    } else {
        $ipaddr = $this->getEnv('REMOTE_ADDR');
    }

    return trim($ipaddr);
}
```

Recommendation

Capture and return the last value in the list of values of the `X-Forwarded-For` HTTP header.

Finding	**XML Entity Expansion Denial of Service**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-CakePHP_2017-018
Category	Data Validation
Location	`src/Utility/Xml.php`: 104-131
Impact	An attacker can cause a severe denial-of-service by exhausting the system's memory.

Description

CakePHP handles user-controllable XML input and is vulnerable to an XML Entity Expansion (XEE) attack. This attack occurs when an insecurely configured XML parser processes an XML document that includes maliciously defined entities.

Because CakePHP does not allow recursive entity expansion, it is not subject to the more severe variant of XEE, the Billion Laughs attack.[6] However, CakePHP is vulnerable to a variation known as the Quadratic Blowup attack, which works by defining a single large entity and referring to it many times.

In the case of CakePHP, If an attacker defines an entity, such as "ent", as being 2ˆ16 characters long and refers to that entity 2ˆ16 times inside the root "data" element, they end up with an XML bomb attack of 385 KB in size that expands to ∼4.3 GB when parsed.

This example results in CakePHP consuming enormous amounts of system resources, eventually causing the framework to fail due to running out of memory.

The following example shows this type of entity expansion:

```
<?xml version="1.0"?>
<!DOCTYPE data [
  <!ENTITY ent "aaaaaaaaaaaaaaaaaa...">
]>
<data>&ent;&ent;&ent;&ent;&ent;&ent;...</data>
```

Reproduction Steps

See Appendix B on page 33.

Recommendation

Configure the XML parser to limit the size of the expanded XML document by setting the `parseHuge` flag to `false`, which limits LibXML from parsing a document that will expand to over 10 MB. If possible, disable `<!DOCTYPE>` processing entirely.

[6] https://en.wikipedia.org/wiki/Billion_laughs_attack

Finding	**randomBytes Returns Insecure Random Numbers**
Risk	Low Impact: High, Exploitability: Low
Identifier	NCC-CakePHP_2017-009
Category	Cryptography
Location	`src/Utility/Security.php`: 100-125
Impact	An attacker may be able to leverage weaknesses in the random number generator to obtain the values of OAuth secrets or CSRF tokens.
Description	If the CakePHP framework does not have access to `random_bytes()` (i.e. PHP < 7) or it cannot use `openssl_random_pseudo_bytes()` it will fall back to using `Security::insecureRandomBytes()`. Although the CakePHP documentation states that if neither source is available a warning will be emitted and an unsafe value will be used for backwards compatibility reasons,[7] this may not be sufficiently visible for administrators of the application at runtime.
Recommendation	If CakePHP cannot generate cryptographically secure random numbers then it must return an error to the user and halt the process.

[7] https://book.cakephp.org/3.0/en/core-libraries/security.html

Finding: Comment Incorrectly Advertises Mitigation for User Enumeration Timing Attack

Risk	Low Impact: Low, Exploitability: Low
Identifier	NCC-CakePHP_2017-010
Category	Cryptography
Location	`src/Auth/BaseAuthenticate.php`: 94-132
Impact	Applications using CakePHP's authentication system may be vulnerable to timing-based user enumeration attacks.

Description

The method documentation comment for `_findUser` in `BaseAuthenticate.php` states:

> *Input passwords will be hashed even when a user does not exist. This helps mitigate timing attacks that are attempting to find valid usernames.*

However, this is not true; the function exits early if a matching username is not found:

```
        $result = $this->_query($username)->first();

        if (empty($result)) {
            return false;
        }
```

The relevant functionality seems to have been removed during refactoring performed in commit aa6088a3 in November, 2013.

If an application's user list is sensitive enough that a developer wishes to protect against timing-based user enumeration, they may be misled into assuming their application is safe and fail to implement their own constant-time login function.

Recommendation

At a minimum, correct the comment. While the previous functionality does not guarantee strict constant-time processing for all submitted usernames, consider re-adding it to make timing attacks more difficult.

Finding: Digest Authentication Uses Non-Constant-Time Comparisons

Risk Low Impact: Low, Exploitability: Low

Identifier NCC-CakePHP_2017-012

Category Cryptography

Location `src/Auth/DigestAuthenticate.php`: 124, 280

Impact If an application uses CakePHP's implementation of HTTP Digest Authentication, an attacker may be able to forge a login credential and impersonate a user.

Description In most programming language libraries, a string comparison will exit as soon as it finds two characters that do not match. A timing attack can be mounted using statistical analysis of the time it takes to compare the values. For some background information on string comparison timing attacks, see https://emerose.com/timing-attacks-explained.

The `DigestAuthenticate` class provides HTTP Digest Authentication. In Digest Authentication, the server provides a nonce value to the client, who returns a cryptographic digest (hash) of: the nonce, the user's password, and several other values. The server computes the same digest, and confirms that the supplied value matches. In the CakePHP implementation, the following code performs this procedure:

```
public function getUser(ServerRequest $request)
{
    $digest = $this->_getDigest($request);
    /* ... */

    if (!$this->validNonce($digest['nonce'])) {
        return false;
    }
    /* ... */

    $hash = $this->generateResponseHash($digest, $password, $request->getEnv('OR
➜   IGINAL_REQUEST_METHOD'));
    if ($digest['response'] === $hash) {
        return $user;
    }

    return false;
}
```

Note the use of a === comparison between the user-supplied response hash and the server's value. Additionally, the server generates nonces using a hash of a secret and an expiration time, and similarly validates it with a non-constant time comparison.

This allows an attacker to first attack the `validNonce` test to create a nonce valid for a time far in the future, and then, using this nonce, to attack the main hash comparison.

Recommendation Use a string comparison function that does not leak information about which characters match, frequently called a "constant time comparison" function.

The general behavior of a constant time comparison is to compare every character with no branching, rather than exiting as soon as a difference is detected:

```
// pseudo-code example
function constantTimeEquals(s1, s2) {
  if length(s1) != length(s2) { return false; }
  acc = 0;
  for( i = 0; i < length(s1); i++) {
    acc |= ( s1[i] ^ s2[i] ); //set bit(s) in acc if chars differ
  }
  return (acc == 0);
}
```

In PHP, the built-in `hash_equals` function should be used for this purpose.

Finding Tutorial Application Allows Path Traversal in Pages Controller

Risk Low Impact: Medium, Exploitability: Low

Identifier NCC-CakePHP_2017-014

Category Access Controls

Location `src/Controller/PagesController.php` in the Bookmarker tutorial application.

Impact If a developer re-uses the tutorial's "Pages" controller for a real application, an attacker could view arbitrary view templates in the application and test for the existence of template files elsewhere on the server.

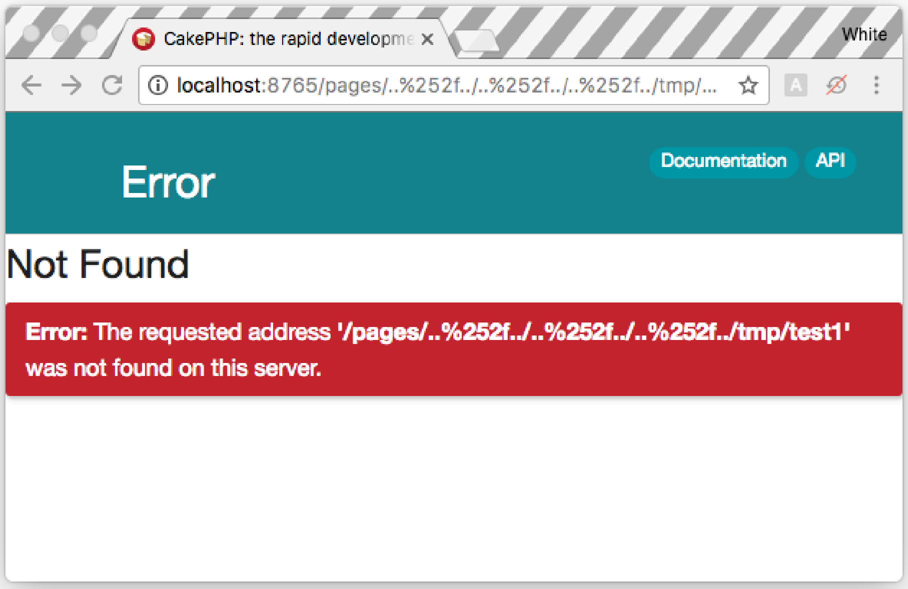
Description When accessing the filesystem, it is extremely important that an application performs security checks to ensure valid access. If a user-controlled filename is provided to the system, it opens a risk of file access that is not limited to the intended files.

The CakePHP "Bookmarker" tutorial application[8] includes a "Pages" controller similar to the one in the official skeleton app.[9] This controller uses the URL path to select a page template to render. Normally this is under `src/Template/Pages` in the application directory. Unlike the version of the controller in the skeleton app, the "Bookmarker" controller does not perform sufficient checks on the supplied path. By passing a path containing the `/../` directory traversal sequence (using double URL-encoded slashes), an attacker can point the renderer at files outside the intended directory.

Exploitation of this vulnerability is limited by two factors in the `View` class. First, code in `_getViewFileName` appends the configured extension (`.ctp` by default) to the file name. Second, `_checkFilePath` validates that the normalized path is within the configured templates directory. These limitations prevent an attacker from reading arbitrary files, but an attacker may still trigger the rendering of arbitrary application templates, which may expose sensitive information.

Additionally, because `_checkFilePath` throws a distinctive exception, an attacker can probe for the existence of `.ctp` files at any path. The following screenshots demonstrate the different outputs:

[8] https://book.cakephp.org/3.0/en/tutorials-and-examples/bookmarks/intro.html
[9] https://book.cakephp.org/3.0/en/controllers/pages-controller.html

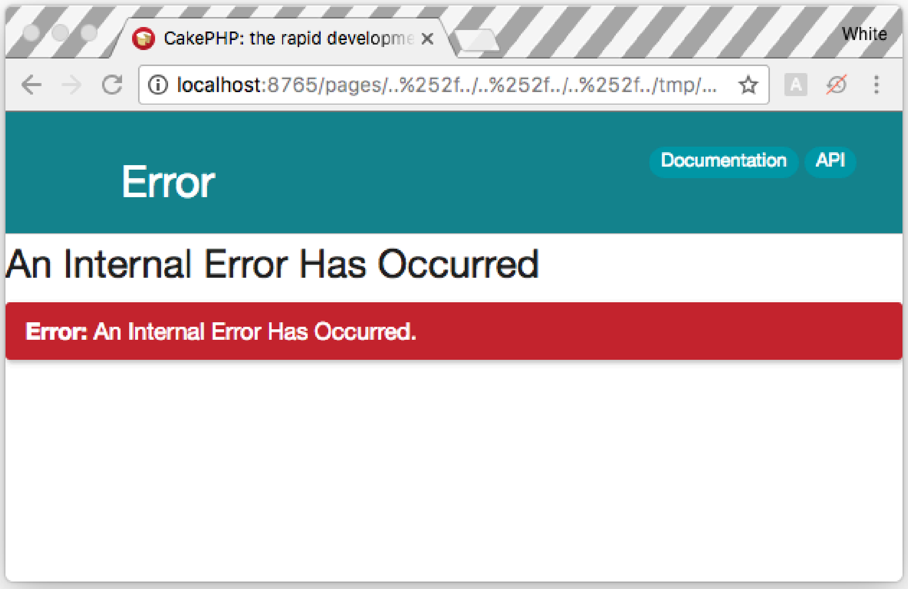

CakePHP: the rapid developme
White
localhost:8765/pages/..%252f../..%252f../..%252f../tmp/...
Documentation
API

Error

Not Found

Error: The requested address '/pages/..%252f../..%252f../..%252f../tmp/test1' was not found on this server.

Figure 1: File does not exist


CakePHP: the rapid developme
White
localhost:8765/pages/..%252f../..%252f../..%252f../tmp/...
Documentation
API

Error

An Internal Error Has Occurred

Error: An Internal Error Has Occurred.

Figure 2: File exists

Reproduction Steps

With the "Bookmarker" tutorial application running on localhost, issue the following commands:

```
# render "list tags" template
# note that tags themselves are not actually shown
curl -is 'http://localhost:8765/pages/..%252fTags/index'

# probe for .ctp files
touch /tmp/good.ctp
curl -is 'http://localhost:8765/pages/..%252f../..%252f../..%252f../tmp/good' \
  | head # returns 500 status
curl -is 'http://localhost:8765/pages/..%252f../..%252f../..%252f../tmp/bad' \
  | head # returns 404 status
```

Recommendation

Update the tutorial and the corresponding GitHub project[10] to perform an explicit test for the `/../` sequence in the Pages controller, and do not render the template if the sequence is found.

[10] https://github.com/cakephp/bookmarker-tutorial

Finding	**AutoLink Functions Use Predictable and Collision-Prone Hashes**
Risk	Low Impact: Low, Exploitability: Low
Identifier	NCC-CakePHP_2017-016
Category	Cryptography
Location	`src/View/Helper/TextHelper.php: 170`
Impact	Mixing trusted and untrusted text into `autoLink` functions may result in malicious links in trusted text, or trusted links in malicious text.

Description

The `autoLink` functions for automatically converting URLs and email addresses into links do so by first replacing those values with their MD5 hashes. The functions then replace those hashes with HTML to make them links. A hash of `b48def645758b95537d4424c84d1a9ff` would be replaced with `<a href="mailto:foo@example.com">foo@example.com</a>`.

The HTML is stored in an associative array before it replaces the hashes in the text, and this array is keyed with the hashes themselves. To replace the hashes in the text with the HTML, the keys of the associative array are iterated across, replacing any instances of them in the text with their associated values. Since the hashes are predictable, it is possible for a user to include MD5 sums of already existing URLs or email addresses in the text they submit to the application, which will then be replaced when the text is processed with an `autoLink` function.

Additionally, due to weakness in MD5, an attacker might be able to craft a malicious URL or email address that, when hashed, collides with an already present URL or email address. In this scenario, the last seen URL or email address will have its HTML link replace all others.

The order of operations of the `autoLink` functions is below:

1. The text is scanned and modified, with URLs and email addresses replaced by their MD5 hashes.
2. HTML links are constructed and placed into an associative array keyed on the MD5 hashes from the matches from the previous step.
3. The text is scanned and modified. Any keys from the associative array found in the text are replaced with their respective values.

Before `autoLink`:

```
foo@example.com b48def645758b95537d4424c84d1a9ff
```

After `autoLink`:

```
<a href="mailto:foo@example.com">foo@example.com</a> <a href="mailto:foo@example
➜  .com">foo@example.com</a>
```

Recommendation

Use an HMAC with a nonce generated from a cryptographically secure random number generator as its key to create MACs as placeholders. This provides both unpredictability and resistance to collisions, even if the underlying hashing function is compromised.

Finding Inconsistent Configuration of Transport Encryption Protocols

Risk Low Impact: High, Exploitability: Low

Identifier NCC-CakePHP_2017-019

Category Cryptography

Location `src/Netwok/Socket.php`: 86 - 97

Impact The use of insecure communication protocols increases the risk of compromise.

Description There are multiple protocols in the TLS/SSL family: SSL v2, SSL v3, and TLS v1.0-1.3. Of these:

- **SSL v2** is insecure and must not be used.
- **SSL v3** is insecure when used with HTTP and weak when used with other protocols. It is obsolete and should not be used.
- **TLS v1.0** is largely still secure; there are no known major security flaws when it is used for protocols other than HTTP. When used with HTTP, it can be made secure with careful configuration and use of the correct cipher suites.
- **TLS v1.1** and **TLS v1.2** have no known security issues.
- **TLS v1.3** is in a "draft" state, and is not supported in PHP.

CakePHP's Socket utility class defines supported encryption methods using the following PHP constants:

```
/**
* Contains all the encryption methods available
*
* @var array
*/
protected $_encryptMethods = [
    // @codingStandardsIgnoreStart
    'sslv2_client' => STREAM_CRYPTO_METHOD_SSLv2_CLIENT,
    'sslv3_client' => STREAM_CRYPTO_METHOD_SSLv3_CLIENT,
    'sslv23_client' => STREAM_CRYPTO_METHOD_SSLv23_CLIENT,
    'tls_client' => STREAM_CRYPTO_METHOD_TLS_CLIENT,
    'sslv2_server' => STREAM_CRYPTO_METHOD_SSLv2_SERVER,
    'sslv3_server' => STREAM_CRYPTO_METHOD_SSLv3_SERVER,
    'sslv23_server' => STREAM_CRYPTO_METHOD_SSLv23_SERVER,
    'tls_server' => STREAM_CRYPTO_METHOD_TLS_SERVER
    // @codingStandardsIgnoreEnd
];
```

The meanings of the `*_SSLv23_*` and `*_TLS_*` constants vary across the versions of PHP supported by CakePHP:

- Prior to 5.6.7:[11]
  - SSLv23 allowed either SSLv2 or SSLv3.
  - TLS allowed TLSv1.0 or greater.
- From 5.6.7 to 7.1.10:[12]
  - SSLv23 allowed TLSv1.0 or greater, but not either version of SSL.
  - TLS specified TLSv1.0 exactly.
  - TLS_ANY was introduced to allow TLSv1.0 or greater.

[11] https://github.com/php/php-src/blob/PHP-5.6.6/main/streams/php_stream_transport.h
[12] https://github.com/php/php-src/commit/10bc5fd4

- As of 7.2:[13,14]
  - All three of these constants allow TLSv1.0 or greater.

As a consequence, while it is possible to configure a secure version of TLS for connections, the correct value to do so varies. Many users will, if they specify "tls", be limited to TLSv1.0. Users on very early versions of PHP 5.6 may accidentally configure the use of insecure SSL protocols. There is no way for users to specify TLSv1.1+ or TLSv1.2 only.

Additionally, CakePHP does not provide granular controls over the cipher suites (a combination of algorithms that provide encryption, authentication, and communications integrity when negotiating a secure TLS/SSL connection) used with the socket.[15] This makes it impossible for users to make appropriate security decisions regarding the type of encryption used to protect their application's communications.

Recommendation Provide an unambiguous way for users to specify use of the best available protocol. Ideally, this should be the default, and opting out of a newer protocol or into an older protocol should require an extra flag or method call. Currently, TLS v1.2 should be the primary protocol, with TLS v1.1 being available as its fallback.

Additionally, the class should allow users to specify cipher suites, with a secure default. The page "Security/Server Side TLS" on the Mozilla Wiki[16] can be used as a reference.

[13] https://github.com/php/php-src/commit/bec91e11
[14] As of early October 2017, PHP 7.2 is in Release Candidate status.
[15] In the absence of explicit configuration, PHP will use OpenSSL's `DEFAULT` set of cipher suites.
[16] https://wiki.mozilla.org/Security/Server_Side_TLS

Finding	**Form Validation Tokens Are Not Associated with Users**
Risk	Low Impact: Medium, Exploitability: Low
Identifier	NCC-CakePHP_2017-020
Category	Access Controls
Location	`src/View/Helper/SecureFieldTokenTrait.php`: 55 - 61
Impact	Form validation tokens may potentially be used to escalate privileges within applications.

Description

Form validation tokens may be used by an application to prevent form fields from being tampered with. The following data is collected, signed, and then embedded in forms:

```
$hashParts = [
  $url,
  serialize($fields),
  $unlocked,
  Security::getSalt()
];
```

The `$url` and `Security::getSalt()` are not visible to users and not embedded directly into the form as part of the token, but are collected with the other data to be signed.[17] The signature is included in the token.

A page might be designed to present two different "views" of the same form, showing different protected fields and the same field names or users with two different privilege levels. An "edit user" page, for example, might only allow a user to change their own password, while allowing an administrator to change username and password. In the event that an administrator's form validation token is compromised, an unprivileged user could replace their own form validation token with that of the administrator's, granting them additional privileges on their own edit user page.

Recommendation

In addition to the URL, include the user's session in the data that is signed, but not embedded into forms.

[17] See finding NCC-CakePHP_2017-021 on the following page for the security implications of the signature implementation.

Finding	**Form Validation Tokens Are Vulnerable to Potential Hash Collisions**
Risk	Low Impact: Medium, Exploitability: Low
Identifier	NCC-CakePHP_2017-021
Category	Cryptography
Location	`src/View/Helper/SecureFieldTokenTrait.php`: 55 - 61
Impact	Collisions in SHA-1 may be exploited to bypass form validation.

Description

Form validation tokens are given to the user alongside a signature, intended to prevent tampering. The tokens include information about form fields that are allowed or not allowed to be modified. The signature is a SHA-1 hash of the token with a secret appended.

While this method seems secure, a SHA-1 collision could be abused to forge validation tokens. The output of SHA-1 is its own internal state, allowing hashes to be "extended" by setting the internal state of SHA-1 to that of any output of SHA-1, and then providing additional input.

This issue is mitigated by the format requirements of the security token, which add a great deal of difficulty to finding a collision, as an invalid format will cause errors before the signature is validated.

The following pseudo-code demonstrates a SHA-1 length extension:

```
$hashFunction = sha1->new();
$state = $hashFunction->hash('foobar');
echo $state; // 8843d7f92416211de9ebb963ff4ce28125932878

$hashFunction = sha1->new();
$state = $hashFunction->hash('foo');
echo $state; // 0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33

$hashFunction = sha1->setState($state);
$state = $hashFunction->hash('bar');
echo $state; // 8843d7f92416211de9ebb963ff4ce28125932878
```

An attacker that can find a collision, such that `sha1($collision) === sha1($token)`, can forge the security token by simply replacing it with `$collision`.

When the server validates the collided token and its signature, it will append the secret to the token, hash it with SHA-1, and then compare the output to the signature that the attacker has also provided. Since the internal state of SHA-1 after processing `$collision` is identical to the internal state after processing `$token`, the output of SHA-1 will be identical after the secret is appended to either.

HMAC functions are immune to these issues, even if collisions are found in their underlying hash functions.

Recommendation

Use an HMAC function to sign form validation tokens.

Finding	**Asset Middleware Can Serve Dot Files**
Risk	Low Impact: Low, Exploitability: Medium
Identifier	NCC-CakePHP_2017-022
Category	Data Exposure
Location	`src/Routing/Middleware/AssetMiddleware.php`
Impact	If an application has sensitive internal "dot files" under the web root directory, an attacker will not be prevented from retrieving them.
Description	Web servers are often configured to refuse to serve files and directories with a leading dot character, such as `.htaccess` or `.git/`, as these files often contain sensitive information. CakePHP's asset middleware does not enforce such a restriction.
Recommendation	Modify the `__invoke` method of `AssetMiddleware` to disallow serving of files where any component of the path begins with a dot.

Finding	**Setters of Entity Attributes Are Not Always Called**
Risk	**Informational** Impact: Medium, Exploitability: Unknown
Identifier	NCC-CakePHP_2017-002
Category	Data Validation
Location	`src/ORM/Marshaller.php`: 574 - 579
Impact	Unexpected results may occur within an application that relies upon setters to always be called.
Description	When updating an object, the ORM does not mark `NULL`, scalars,[18] or objects[19] as "dirty" if they are identical to already persisted values. If something is not marked "dirty," it is not treated as modified and various actions are not taken, such as calling their associated setters. **Note:** This is not an intrinsic security issue.
Recommendation	Document this behavior.

[18] https://secure.php.net/manual/en/function.is-scalar.php
[19] https://secure.php.net/manual/en/function.is-object.php

Finding Insecure Default Hashing Algorithm

Risk Informational Impact: High, Exploitability: Unknown

Identifier NCC-CakePHP_2017-006

Category Cryptography

Location `src/Utility/Security.php`: 33

Impact An attacker may be able to compromise the security of applications built with the CakePHP framework by colliding SHA-1 hashes.

Description A cryptographic hash is a function that takes a string of bytes and returns a small, fixed-size value. Hash functions guarantee that the same input always results in the same output. When used for security, the most important property of a hash function is that it is impossible for an attacker to produce two inputs that hash to the same value (called a collision). Hashes that do not have this property are considered to be insecure.

The CakePHP framework provides a `Security::hash` function, which defaults to the SHA-1 algorithm:

```
/**
 * Default hash method. If `$type` param for `Security::hash()` is not specified
 * this value is used. Defaults to 'sha1'.
 *
 * @var string
 */
public static $hashType = 'sha1';
```

Theoretical attacks on SHA-1 have been known to exist since 2005,[20] and have recently been demonstrated in practice.[21] SHA-1 has been deprecated for signature purposes by the United States National Institute of Standards and Technology (NIST) since 2011.[22] Consequently, applications should migrate to the SHA-2 or SHA-3 algorithm families for secure hashing.

Recommendation Use a hashing algorithm without known vulnerabilities, such as SHA-2 (SHA-256 / SHA-512) or SHA-3.

[20] https://en.wikipedia.org/wiki/SHA-1#Attacks
[21] https://shattered.io/
[22] https://csrc.nist.gov/publications/detail/sp/800-131a/archive/2011-01-13

Finding	**Certificate Authority Bundle Includes Untrustworthy CAs**
Risk	**Informational** Impact: Informational, Exploitability: Informational
Identifier	NCC-CakePHP_2017-007
Category	Configuration
Location	`config/cacert.pem`
Impact	An attacker may be able to use a collision attack to forge a valid certificate signed using SHA-1 and perform a server-impersonation attack.
Description	CakePHP uses Mozilla's Certificate Authority (CA) Bundle from January 20, 2016. This bundle is out of date and includes a number of untrustworthy CA Root Certificates from WoSign and StartCom. Mozilla discovered multiple problems in the SSL certificate issuance process of WoSign, a China-based certificate authority. The investigation also revealed that in 2015, WoSign silently acquired StartCom, a CA based in Israel, without disclosing the deal to browser vendors who operate certificate root programs.[23] Some of the most egregious violations of the WoSign CA include 64 cases of WoSign issuing certificates signed with SHA-1 after the sunset date, January 1, 2016. Additionally, WoSign backdated these certificates to hide their misdeeds.
Recommendation	Update CakePHP's CA bundle to Mozilla's current CA Root Certificate bundle. Develop a process that will ensure that CakePHP's CA bundle will remain up to date.

[23] https://www.pcworld.com/article/3137240/security/google-to-untrust-wosign-and-startcom-certificates.html

Finding	**Insecure UUID Function**
Risk	Informational Impact: Medium, Exploitability: Informational
Identifier	NCC-CakePHP_2017-008
Category	Cryptography
Location	`src/Utility/Text.php`: 42-62
Impact	Users of CakePHP may inadvertently make themselves vulnerable to UUID collision attacks that could result in unauthorized access.

Description

CakePHP includes a function to generate Universally Unique Identifiers (UUIDs). This function, `Text::uuid()`, produces version 4 UUIDs, which are based on pseudo-random numbers.[24] It uses `mt_rand`, which is not a cryptographically secure random number generator, to generate these:

```
public static function uuid()
{
  return sprintf(
    '%04x%04x-%04x-%04x-%04x-%04x%04x%04x',
    // 32 bits for "time_low"
    mt_rand(0, 65535),
    mt_rand(0, 65535),
    // 16 bits for "time_mid"
    mt_rand(0, 65535),
    // 12 bits before the 0100 of (version) 4 for "time_hi_and_version"
    mt_rand(0, 4095) | 0x4000,
    // 16 bits, 8 bits for "clk_seq_hi_res",
    // 8 bits for "clk_seq_low",
    // two most significant bits holds zero and one for variant DCE1.1
    mt_rand(0, 0x3fff) | 0x8000,
    // 48 bits for "node"
    mt_rand(0, 65535),
    mt_rand(0, 65535),
    mt_rand(0, 65535)
  );
}
```

Internally, the CakePHP framework does not use UUIDs generated by `Text::uuid()` in any security related contexts. It uses them as part of the `Message-ID` mail header as well as for seeding the random number generator used in `Security::insecureRandomBytes()`.

The comments for this function warn users against using its return value as a seed for cryptographic operations, however this may not be sufficient to protect against misuse. Users of the CakePHP framework may unintentionally use this function in non-obvious security related contexts and become vulnerable.

If a user has preconceived notions of what a version 4 UUID should be they may use them in a way that makes them vulnerable. For example, a user makes a photo-sharing application where each photo corresponds to a URL of the format "https://example.com/photo/{v4 uuid}". If they read the Wikipedia entry for Universally Unique Identifier,[25] they may assume that the chance of a collision occurring is so low they do not need to worry about someone brute forcing their way into obtaining access to the photos.

[24] https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_4_.28random.29
[25] https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_4_.28random.29

Recommendation In the short term, the comments for this function should be more explicit in their warnings about using `uuid()` in any security related contexts.

Long term, the CakePHP framework should rewrite this function to use a cryptographically secure random number generator and generate version 4 UUIDs compliant with RFC4122.[26]

[26] https://tools.ietf.org/html/rfc4122

Finding	**Digest Authentication Uses MD5 Algorithm to Construct Nonces**
Risk	Informational Impact: Low, Exploitability: None
Identifier	NCC-CakePHP_2017-011
Category	Cryptography
Location	`src/Auth/DigestAuthenticate.php`: 250-257
Impact	This is an informational finding only, however it is considered a best practice to avoid the use of the deprecated MD5 algorithm.
Description	HTTP Digest Authentication uses a nonce value to protect against replay attacks. The specification[27] does not define how these values are to be generated or verified, but the use of a signed timestamp is a common pattern. CakePHP implements this pattern with the signature `MD5( timestamp || secret )`. While there are no known practical attacks against the construct used here, the MD5 algorithm is considered insecure.[28] Practical collision attacks against MD5 have been demonstrated, and a theoretical preimage attack has been shown to exist.
Recommendation	Replace this use of MD5 with an HMAC construction using a modern cryptographic hash such as SHA-2 (256 or 512) or SHA-3.

[27] https://tools.ietf.org/html/rfc2617#section-4.3
[28] https://en.wikipedia.org/wiki/MD5#Security, https://www.kb.cert.org/vuls/id/836068

Appendix A: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

Appendix B: XML Denial of Service Reproduction Steps

The following steps can be used to reproduce finding NCC-CakePHP_2017-018 on page 11:

1. Save the following Python script as `build_quadxml.py`:

```
#!/usr/bin/python

def main():
    print "<?xml version=\"1.0\"?>"
    print "<!DOCTYPE data ["
    print "<!ENTITY a "
    print "\"" + 'a'*(2**16) + "\">"
    print "]>"
    print "<data>"
    print "&a;"*((2**16))
    print "</data>"

if __name__ == "__main__":
    main()
```

2. Run the following command:

```
python build_quadxml.py > /tmp/quad.xml
```

3. Create a CakePHP application. Ensure that the SimpleXML extension is installed and enabled.[29]

4. Create the following files in the application's directory:

   `src/Controller/PocController.php`:

```
<?php
namespace App\Controller;
use App\Controller\AppController;

class PocController extends AppController
{
    public function quadxml() { }
}
?>
```

   `src/Template/Poc/quadxml.ctp`:

```
<?php
use Cake\Utility\Xml;
use Cake\Utility\Exception\XmlException;

print "XML DoS - Quadratic Blowup";
$xml = Xml::build('/tmp/quad.xml');
print $xml;
?>
```

5. Run the application with `./bin/cake server`, and browse to http://localhost:8765/poc/quadxml

6. Observe that an error page is returned after a delay of up to 60 seconds.

7. Additionally, check the CakePHP console output for output similar one of the following messages:

```
[Fri Aug 11 14:50:46 2017] PHP Fatal error:  Maximum execution time of 60 seconds exceeded in
/home/b/cakephp/bookmarker/src/Template/Poc/xml.ctp on line 486
```

[29] On Ubuntu: `apt install php7.0-xml`

```
mmap() failed: [12] Cannot allocate memory
```