Laravel meetup vol. 6
3. 12. 2020

Webnotion

Better security (not only) in Laravel

Ján Šmatlík

OWASP

- The Open Web Application Security Project®
- Non-profit organization for improve security of web applications
- Projects
  - OWASP Top Ten
  - Web Security Testing Guide
  - OWASP Cheat Sheet Series *cheatsheetseries.owasp.org*

Cross-site scripting (XSS)

- Inserting unwanted code to our app
- Types:
  - **Stored** `<script>alert(1)</script>`
    - saved in DB via user input / import
  - **Reflected** `example.com/search?q=<img+src=x+onerror=alert(1)>`
    - HTTP request, immediate response in an unsafe way
  - **DOM based** `$("#content").html(ajaxResponse);`
    - JS load data from untrusted source

XSS – how to prevent

- sanitize all!
  - inputs (request validate, xml validator)
  - outputs (blade)
    - don‘t use „noescape“ or pay extra attention
  - cookies
- don‘t send generated HTML from user input via AJAX
- don‘t forget on emails, feeds, etc.
- HTML purifier (wysiwyg/html input whitelisting)

Security HTTP headers

Testing tool: securityheaders.com

- ~~*X-XSS-Protection*~~ - deprecated/removed in browsers now
- **X-Content-Type-Options**: *nosniff*
- **X-Frame-Options**: SAMEORIGIN / DENY
- **Referrer-Policy**:
  - see MDN

Security HTTP headers (2)

```
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
```

- **Strict-Transport-Security (HSTS)**:
  - prevent from non-SSL access to app
  - browser will auto-redirect to secure version
  - „**includeSubDomains**“
  - „**preload**“ - browser will load secure version directly
    - preload list: *hstspreload.org*
    - ! cfg mistake + long „max-age“ may cause shutdown

Security HTTP headers (3)

- **Content-Security-Policy**
- Draft: **Permissions-Policy**
  - previously named Features-Policy
  - currently not implemented in any browser
  - will block specified features (camera, battery, mic, geolocation, ...)

CSP – easy implementation in Laravel

- external package, for example:

  spatie/laravel-csp
  - define project specific policies or use Basic
  - sets only CSP security headers
- alternatively use:

  bepsvpt/secure-headers
  - handles almost all security headers

CSP – demonstrations in IDE

- demonstration of **bepsvpt/secure-headers** package
- different sources and datatypes
- tracking issues via report-uri.com

Access control

- **Vertical access controls**
  - application privileges -> myaccount / admin stuff / ...
- **Horizontal access controls**
  - resources scope -> my data / department data / all data
- **Context-dependent access controls**
  - access based on state of application or user interaction
    - shop process -> cannot change cart during payment process

Access control vulnerabilities

- **Insecure Direct Object Reference** (IDOR)

  GET /order/{id}/detail

  - obviously broken horizontal AC
    - user has access to foreign subset of foreign data
    - example: User has privileges to view/manage Documents of foreign departments of Organization
  - use horizontal AC or combination
  - (optionally) use also UUID

Access control in Laravel

- use Gates and Policies for Horizontal + Vertical AC
- Save state to session/cache/db for Context-dependent AC

Access control in Laravel (2)

Laravel Gate example

```php
// in AuthServiceProvider boot() method or in separate Gate class
Gate::define("view-doc", function ($user, $doc) {
  return $user->department()->id == $doc->createdBy()->department_id;
});

// usage in controller method
public function view($doc)
{
    if (!Gate::allows("view-doc", $doc)) {
        return response("Forbidden", 403);
    }
}
```

Injections

```php
// DB raw - bad way
DB::select("SELECT * FROM tbl WHERE col = '".$value."'");
DB::select("SELECT * FROM tbl WHERE '".$column." = '".$value."'");

// DB raw - good (!!) way
DB::select("SELECT * FROM tbl WHERE col = ?", [ $value ]);
DB::select("SELECT * FROM tbl WHERE ? = ?", [ $column, $value ]);
// better way
DB::select("SELECT * FROM tbl WHERE col = :value", [ 'value' => $value ]);
// best way = not use raw queries if possible
Model::where('col', $value);
```

- SQL injection
  - use ORM (Eloquent/Doctrine) or PDO (non-framework)
  - don‘t use native queries if don‘t need
    - in these cases, properly escape all input variables
    - or use prepared statements or stored procedures

SQL injection via Validation Rule

```php
$id = $request->route('id');
$rules = [
    'username' => 'required|unique:users,username,' . $id,
];

$validator = Validator::make($request->post(), $rules);
if ($validator->fails()) {
    return response()->json($validator->errors(), 422);
}

$user = User::findOrFail($id);
$user->fill($validator->validated());
$user->save();
```

read more at: https://cyberpanda.la/blog/laravel-validation-rule-injection

XXE injection

XML External Entity Injection

- XML format exposure
- can access sensitive data
- PHP<8
  - use libxml explicitly
  - disable processing if contains DOCTYPE
- PHP>=8
  - libxml used by default

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>

```

```
libxml_disable_entity_loader(true)
```

Cross-site Request Forgery (CSRF)

- use middleware VerifyCsrf
  - inject: @csrf in every form
- Cookie XSRF-TOKEN for ajax requests
  - in Axios loaded automatically (VUE, Angular)
  - easy inject into JQuery, too

AJAX security

- use innerText (or framework equivalent) instead of innerHTML in most cases
- don‘t use eval()
- encode correctly HTML / CSS / XML / JSON
- don‘t rely on frontend validation or frontend logic only
  - always duplicate on server-side (ex.: Request Validations)

Ajax security (2)

```
1 const axios = require('axios');
2
3 axios.get('path').then(resp => {
4     this.formattedTitle = resp.formattedTitle
5 });
6
7 // vulnerable:
8 <h1 v-html="formattedTitle"></h1>
```

```
1 const axios = require('axios');
2
3 axios.get('path').then(resp => {
4     this.titleParts = resp.titleParts
5 });
6
7 // safe
8 <h1>
9   <a href="{{ titleParts.parentUrl }}">{{ titleParts.parentTitle }}:</a>
10  {{ titleParts.currentTitle }}
11 </h1>
```

Session cookie settings

- HttpOnly (no JS access), Secure (only via HTTPS)
- SameSite -> strict / lax / none MDN Diagrams
- in Laravel:

config/session.php

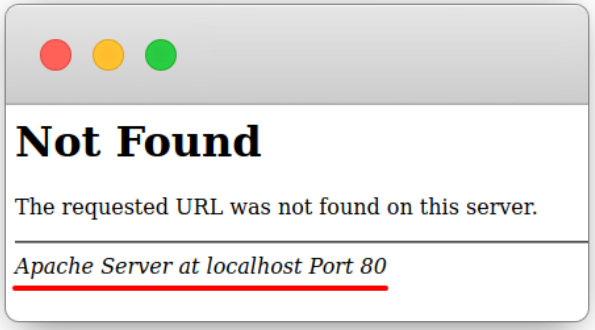
```php
// ...
'encrypt'   => true,
'secure'    => true,
'http_only' => true
'same_site' => "strict", // or "lax"
```

- also applicable for all important app cookies

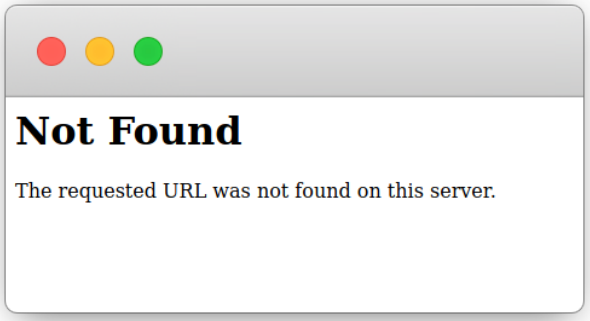
Full Path Disclosure

- vulnerable versions of server packages / app libraries
- accessible phpinfo()
- breakable code to show any error (incorrect QS / Post data / cookie value) -> *for example: sent „foo“ instead of „1“*
- inproper .env settings, server cfg
- publicly accessible .git / .svn / .env / ...
- vulnerable versions of 3rd party apps in same context (sys user or virtual host) - phpmyadmin / adminer

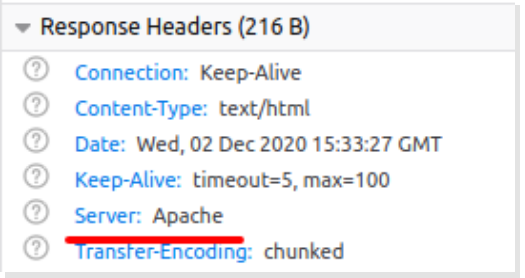
Full Path Disclosure (2)



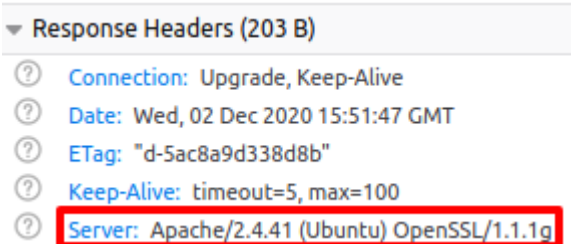
ServerSignature: on / off



ServerToken: prod



ServerToken: full



Proper env configuration

- PHP config

```
expose_php = off
disable_functions = ... # use defaults from vendor
log_errors = on
error_reporting = E_ALL
display_errors = off
```

- Laravel config

```
APP_ENV = prod
APP_DEBUG = false
APP_KEY = ... # per env, unique, not public
```

- Apache config

```
ServerSignature = Off
ServerToken Prod
```

- Nginx config

```
server_tokens off;
```

SSL configuration & testing

- SSL is must! (SEO, browsers warnings, user privacy ofc.)
- Current standards: TLS 1.2 & TLS 1.3
  - ! only secure ciphers
- check your server configuration: ssllabs.com/ssltest
- check issued certificates for domains: crt.sh
  - potentional security risk (exposing non-public subdomains -> wildcard certs)

Free SSL certificates

- **Let‘s Encrypt**
  - free single/multi domain and wildcard certs
  - since 1/2021 new default ROOT CA: ISRG Root X1
    - will be untrusted in Android <7.1.1
      - use Firefox Mobile or newer device (christmas gift ☺)
- **ZeroSSL** - ACME API, only 3x 90d free, premium packages
- **Buypass** - ACME API, unlimited, but no wildcard

security.txt

- proper security vulnerabilities disclosure
  - Should be directly addressed to person responsible for security of application
- generator + info at: securitytxt.org
- is that used by someone?
  - yes, some of ethical hackers, already do

Usefull links and tools

- https://owasp.org/www-project-web-security-testing-guide/latest/
- https://cheatsheetseries.owasp.org
- https://portswigger.net/web-security
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite
- https://ssllabs.com/ssltest
- https://hstspreload.org
- https://report-uri.com
- https://crt.sh
- https://michalspacek.cz
- https://scotthelme.co.uk

Discussion

- How you deal with security in your team?
- Do you use CSP?
- What I forgot to mention?
- Something else?

Thanks for your attention!

email: jan@smatlik.com

presentation: jan.smatlik.com/larameet6