SYNACKTIV

Laravel - Deep dive in laravel encryption

Rémi Matasse & Mickaël Benassouli

Grehack - 2024

Introduction

SYNACKTIV

Who are we?

- Rémi (@_remsio_) and Micka (@Kainx42), security experts at Synacktiv.
- Company specialized in offensive security: penetration testing, reverse engineering, trainings, etc.
- Around 180 experts over 5 offices in France (Paris, Lyon, Toulouse, Rennes, Lille).
- Let's work together!

Table of content

SYNACKTIV

- Introduction to Laravel
  - Encryption mechanism
  - Laravel exposure to unserialize attacks
  - Quick presentation of a concrete `APP_KEY` usage
- Vulnerability research on Laravel `decrypt` attacks
  - research results
  - 3 vulnerabilities on big Laravel projects :)
  - exploitation demos with our new tool
- Analysis of the `APP_KEY` quality in publicly accessible Laravel applications
  - Analysis of previous attacks detected in the wild
  - How to find valid `APP_KEY`s
  - Result of our research on Laravel public exposure on this kind of leak

Introduction to Laravel

Introduction to laravel

SYNACKTIV

- Laravel is a free, open-source PHP framework designed for web application development. It follows the Model-View-Controller (MVC) architectural pattern, promoting organized and maintainable code.
- Laravel is popular for designing web applications such as e-commerce platforms, social networking platforms, APIs (Application Programming Interfaces), and Content Management Systems (CMS).
- Laravel applications often handle critical data, making them attractive targets for attackers.
- Laravel is utilized by **1,235,487** live and historical websites, according to *BuiltWith*.

> https://trends.builtwith.com/framework/Laravel

Encrypt function usage

SYNACKTIV

Laravel
Encrypt and Decrypt

- Laravel simplifies encryption through the `encrypt` function
  - Based on OpenSSL library to achieve a high-security standard.
- This function is loaded from the package `Illuminate\Encryption`

```php
// Encrypting data

$originalData = 'Hello, world!';
$encryptedData = encrypt($originalData);
```

- `encrypt` and its complementary function `decrypt` are loaded **by default**
  - `use Illuminate\Encryption` not required in a Laravel project
- Often used as an **integrity validator** to give or store sensitive data

Encrypted value format

SYNACKTIV

- Base64 string containing 4 values used by **AES-CBC-256**, other algorithms can be used but this is this one by default
  - iv : Initialization vector, generated randomly each time
  - value : the value is ciphered from iv + `APP_KEY`
  - mac : a validation mac to prevent padding oracle attacks
  - tag : used in other modes such as GCM

```
$ echo "eyJpdiI6IkJsYXJrVHlpcm0xT3pNbEIxc01neFE9PSIsInZhbHVlIjoicnBXNWEyR0drN2
hERlJsOFZDMUpuU2Nsd0hkRDIxWUJ2aWhTTVRmRG9aaz0iLCJtYWMiOiJkMTk5ZDg3NWY0NjE0ODJ
iNDcwZWMwNDkyMWRkYTM2ODIyODM3MWEzYmJjN2VjZGVmMzE4NmVjMDFjMDUyYjY0IiwidGFnIjoi
In0=" | base64 -d | jq
{
  "iv": "BlarkTyirm1OzMlB1sMgxQ==",
  "value": "rpW5a2GGk7hDFRl8VC1JnSclwHdD21YBvihSMTfDoZk=",
  "mac": "d199d875f461482b470ec04921dda36822837 1a3bbc7ecdef3186ec01c052b64",
  "tag": ""
}
```

Laravel exposure to unserialize attacks

SYNACKTIV

- Laravel contains a lot of gadgets that were never patched
- phpggc is a tool created by **Charles Fol** to craft unserialize payload
- This tool also contains the script test-gc-compatibility.py designed to test the usability of payloads on a given library

```
user@poc-laravel:~/Desktop/tool/phpggc$ docker run --entrypoint './test-gc-compatibility.py' phpggc laravel/laravel:10.3.3,9.5.2,8.6.11,7.30.1,6.20.1 Laravel/RCE4 Laravel/RCE8 Laravel/RCE9
Running on PHP version PHP 8.1.30 (cli) (built: Oct 28 2024 22:05:20) (NTS).
Testing 5 versions for laravel/laravel against 9 gadget chains.
```

laravel/laravel	Package	Laravel/RCE4	Laravel/RCE8	Laravel/RCE9	Laravel/RCE10	Laravel/RCE13	Laravel/RCE15	Laravel/RCE17	Laravel/RCE19	Laravel/RCE20
10.3.3	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
9.5.2	OK	OK	OK	OK	OK	OK	OK	OK	KO	OK
8.6.11	OK	OK	OK	OK	OK	OK	OK	KO	KO	OK
7.30.1	OK	OK	OK	OK	OK	OK	OK	KO	KO	OK
6.20.1	OK	OK	KO	OK	OK	OK	OK	KO	KO	OK

- The core developpers of Laravel were contacted several times to point out this point, however..

Laravel exposure to unserialize attacks

SYNACKTIV

Decrypt function

SYNACKTIV

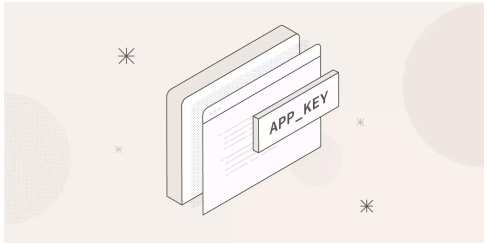
Laravel
Encrypt and Decrypt

```
namespace Illuminate\Encryption;
public function decrypt($payload, $unserialize = true)
    {
        $payload = $this->getJsonPayload($payload);

        $iv = base64_decode($payload['iv']);

        $decrypted = \openssl_decrypt( $payload['value'], $this->cipher, $this->key, 0, $iv );

        [...]

        return $unserialize ? unserialize($decrypted) : $decrypted;
    }
```

- **By default** the parameter `$unserialize` is set to **true**
- The value of `$this->key` is the `APP_KEY` of the Laravel application
- Therefore : **controlling a data passed to `decrypt` + getting `APP_KEY` = RCE**

What about the APP_KEY

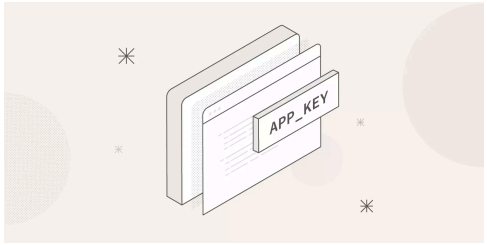
SYNACKTIV



- Getting the `APP_KEY` is the **primary** requirement to control decryption
- Without this key, any attempt to `decrypt` data will fail
- Laravel encryption is used as an **Integrity Check** which Ensures the application data remains unchanged during storage and transport

Storage Locations for APP_KEY

SYNACKTIV



Where is APP_KEY stored?

1. `.env` **File**
   - Located in the application's root directory
   - Commonly accessed via the `APP_KEY` or `APP_PREVIOUS_KEYS` parameters
   - **Security**: Important to restrict access to this file as it contains sensitive environment configurations
2. `config/app.php` **Configuration**
   - Laravel configuration file which loads `.env` variables
   - Key is defined by `'key' => env('APP_KEY')`
   - By default, Laravel dynamically references `APP_KEY` from `.env`, ensuring environment-specific key loading

Key Rotation

SYNACKTIV

- **APP_PREVIOUS_KEYS**: Is an array containing previous `APP_KEY` values
- **Purpose**: Facilitates key rotation without causing decryption failures on legacy data.
- However..

Key Rotation

SYNACKTIV

Securing the APP_KEY

SYNACKTIV

- **Security Best Practices**:
  - Store `.env` in restricted directories.
  - Avoid exposing `.env` in version control systems.
  - Rotate `APP_KEY` and use `APP_PREVIOUS_KEYS` as needed.
- `APP_KEY` is foundational to data confidentiality within Laravel applications, securing encrypted and decrypted data.

Attack surface analysis

SYNACKTIV

In the past : Cookie Exploitation - Before **Laravel 5.6.30**, session cookies were serialized, leaving them open to RCE. CVE-2018-15133

Now : Insecure Decryption Calls :

- *Risk* : Many Laravel applications use `decrypt()` without setting `unserialize=false`.
- *Example* : The `laravel-opcache` package uses `Crypt::decrypt()` with default settings, allowing attacker-controlled data to be decrypted and deserialized.

Laravel encrypted components (blogpost from **Timo Müller**) :

- Cookies
- Queues
- Signed Urls

> https://mogwailabs.de/en/blog/2022/08/exploiting-laravel-based-applications-with-leaked-app_keys-and-queues/

Laravel Crypto Killer

SYNACKTIV

- A tool we developed to manipulate/exploit Laravel ciphers
- 3 modes are available at the moment
  - **encrypt**
    - Used to mimic `Illuminate\Encryption` `encrypt` function
    - Can be used with the mode `--session_cookie` to exploit `SESSION_DRIVER=cookie`
  - **decrypt**
    - Used to mimic `Illuminate\Encryption` `decrypt` function
  - **bruteforce**
    - Used to perform a scaled bruteforced on one or more Laravel ciphers

Laravel Crypto Killer (Demo)

SYNACKTIV

```
user@debian:~/Bureau/tools/laravel-crypto-killer$ ./laravel_crypto_killer.py -h
usage: laravel_crypto_killer.py [-h] {encrypt,decrypt,bruteforce} ...


 ___                              _       ___                _           ___     _   _
(O O)                            (_ )    ( O_`\             ( )_        ( O)   _(_ )(_ )
 | |      _ _ _ __  _ _ _   _  __ | |    | ( (_)_ __ _   _ _ _ | ,_) _  |  |/')(_)| | | |   __  _ __
 | |    /'_` ( '__/'_` ( ) ( )/'__`\| |    | |  _( '__( ) ( ( '_`\| |  /'_`\   |  , < | || | | | /'__`( '__)
<  |__ ( (_| | | ( (_| | \_/ (  ___/| |  <  (_( | |  | (_) | (_) | |_( (_) )  <  | \`\| || | | |(  ___| |
<_____/`\__,_(_) `\__,_`\___/`\____(___)  <_____/(_)  `\__, | ,__/`\__`\___/'  <__)  (_(_(___(___`\____(_)
                                                      ( )_| | |
                                                      `\___/(_)
        This tool was firstly designed to craft payload targetting the Laravel decrypt() function from the package Illuminate\Encryption.

        It can also be used to decrypt any data encrypted via encrypt() or encryptString().

        The tool requires a valid APP_KEY to be used, you can also try to bruteforce them if you think there is a potential key reuse from
 a public project for example.

        Authors of the tool : @_remsio_, @Kainx42
        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


options:
  -h, --help            show this help message and exit

subcommands:
  You can use the option -h on each subcommand to get more info

  {encrypt,decrypt,bruteforce}
    encrypt             Encrypt mode
    decrypt             Decrypt mode
    bruteforce          Bruteforce potential values of APP_KEY. By default, all the values from the folder wordlists will be loaded.
```

Vulnerability research on this type of pattern

Vulnerable patterns

SYNACKTIV

- In `AuthServiceProvider.php` : `Passport::withCookieSerialization();`
  - Revive the vulnerability CVE-2018-15133
- On default Laravel encryption via `decrypt` or `Crypt::decrypt`
  - Call to unserialize by default
- Stores the session in the cookie via `SESSION_DRIVER=cookie`
  - was by default until Laravel 6.12.0 during 2020

Vulnerable patterns - vulnerable products

SYNACKTIV

- Snipe-IT **CVE-2024-48987** - 11.1k stars on Github
  - In `AuthServiceProvider.php` : `Passport::withCookieSerialization();`
  - Revive the vulnerability CVE-2018-15133
- Invoice Ninja **CVE-2024-XXXXX** - 8.2k stars on Github
  - On default Laravel encryption via `decrypt` or `Crypt::decrypt`
  - Call to unserialize by default
- Crater **CVE-2024-XXXX** - 7.8k stars on Github (Fork : *Invoiceshelf*)
  - Stores the session in the cookie via `SESSION_DRIVER=cookie`
  - was by default until Laravel 6.12.0 during 2020

Snipe-IT

SYNACKTIV

2 files changed +1 -3 lines changed

app/Http/Middleware/EncryptCookies.php

```
@@ -20,5 +20,5 @@ class EncryptCookies extends BaseEncrypter
20 20         *
21 21         * @var bool
22 22         */
23    -      protected static $serialize = true;
   23 +      protected static $serialize = false;
24 24     }
```

app/Providers/AuthServiceProvider.php

```
@@ -87,11 +87,9 @@ public function boot()
87 87             ]);
88 88
89 89             $this->registerPolicies();
90    -          //Passport::routes(); //this is no longer required in newer passport versions
91 90             Passport::tokensExpireIn(Carbon::now()->addYears(config('passport.expiration_years')));
92 91             Passport::refreshTokensExpireIn(Carbon::now()->addYears(config('passport.expiration_years')));
93 92             Passport::personalAccessTokensExpireIn(Carbon::now()->addYears(config('passport.expiration_years')));
94    -          Passport::withCookieSerialization();
95 93
96 94
97 95             /**
```

Invoice Ninja

SYNACKTIV

3 files changed +44 -29 lines changed

routes/client.php

```
@@ -144,20 +144,20 @@
144 144     Route::get('unsubscribe/{entity}/{invitation_key}', [App\Http\Controllers\ClientPortal\InvitationController::class, 'unsubscribe'])->name('unsubscribe');
145 145 });
146 146
147     - Route::get('route/{hash}', function ($hash) {
    147 + // Route::get('route/{hash}', function ($hash) {
148 148
149     -     $route = '/';
    149 +     //     $route = '/';
150 150
151     -     try {
152     -         $route = decrypt($hash);
153     -     }
154     -     catch (\Exception $e) {
155     -         abort(404);
156     -     }
    151 +     //     try {
    152 +     //         $route = decrypt($hash);
    153 +     //     }
    154 +     //     catch (\Exception $e) {
    155 +     //         abort(404);
    156 +     //     }
157 157
158     -     return redirect($route);
    158 +     //     return redirect($route);
159 159
160     - })->middleware('throttle:404');
    160 + // })->middleware('throttle:404');
```

Crater --> InvoiceShelf

SYNACKTIV

1 file changed +1 -1 lines changed

.env.example

```
2  2   APP_KEY=base64:kgk/4DW1vEVy7aEvet5FPp5un6PIGe/so8H0mvoUtW0=
3  3   APP_DEBUG=true
4  4   APP_NAME="InvoiceShelf"
5  5   APP_LOG_LEVEL=debug
6  6   APP_TIMEZONE=UTC
7  7   APP_URL=http://invoiceshelf.test
8  8
9  9   APP_LOCALE=en
10 10  APP_FALLBACK_LOCALE=en
11 11  APP_FAKER_LOCALE=en_US
12 12
13 13  APP_MAINTENANCE_DRIVER=file
14 14  APP_MAINTENANCE_STORE=database
15 15
16 16  BCRYPT_ROUNDS=12
17 17
18 18  DB_CONNECTION=sqlite
19 19  DB_HOST=
20 20  DB_PORT=
21 21  DB_DATABASE=
22 22  DB_USERNAME=
23 23  DB_PASSWORD=
24 24
25 25  BROADCAST_CONNECTION=log
26 26  CACHE_STORE=file
27 27  QUEUE_CONNECTION=sync
28    - SESSION_DRIVER=cookie
   28 + SESSION_DRIVER=file
```

Does the vulnerability require the attack to know the APP_KEY environment variable?

Analysis of the APP_KEY quality in publicly accessible Laravel applications

Previous attack based on an APP_KEY leakage : Androxgh0st

SYNACKTIV

Attack performed during january 2024

- The Malware scans the internet for Laravel applications.
- Upon finding a Laravel application, it checks for exposed .env files to steal credentials and API keys.
- It may also send a POST request with the variable `0x[]` to trigger an error, aiming to identify sites with debug mode enabled, which can also expose credentials and API keys.
- If it successfully accesses the application key, it attempts to exploit a known Remote Code Execution (RCE) vulnerability in Laravel **v5.2** using the `XSRF-TOKEN` cookie.

Blogpost from **Stephen Rees-Carter**

> https://securinglaravel.com/laravel-security-androxgh0st-malware/

Use for attacker

SYNACKTIV

- Legitimate Data Serialized with `APP_KEY`
  - Attackers with a known `APP_KEY` can bypass security checks, embedding payloads in serialized data that Laravel may deserialize, potentially executing malicious code.
  - Serialized PHP objects with gadget chains (e.g., PHPGGC) can trigger arbitrary commands without detection by **WAFs**.
- Minimal Logging in Default Configurations:
  - Deserialization and command execution may leave no trace in logs, unlike standard POST requests from typical web shells.

Use for attacker

SYNACKTIV

Analysis performed by Synacktiv

SYNACKTIV

- **Objective**:
  1. Assess the robustness of `APP_KEY` values used in **publicly** available Laravel applications, identifying potential vulnerabilities in keys that are too weak or common
  2. Make a good **Wordlist** of `APP_KEY` for offline bruteforce
- **Tactics**:
  1. Exfiltrate Serialized Objects: Collecting serialized objects from exposed Laravel applications
  2. Prioritizing Common Targets: Targeting weak or common `APP_KEY` values

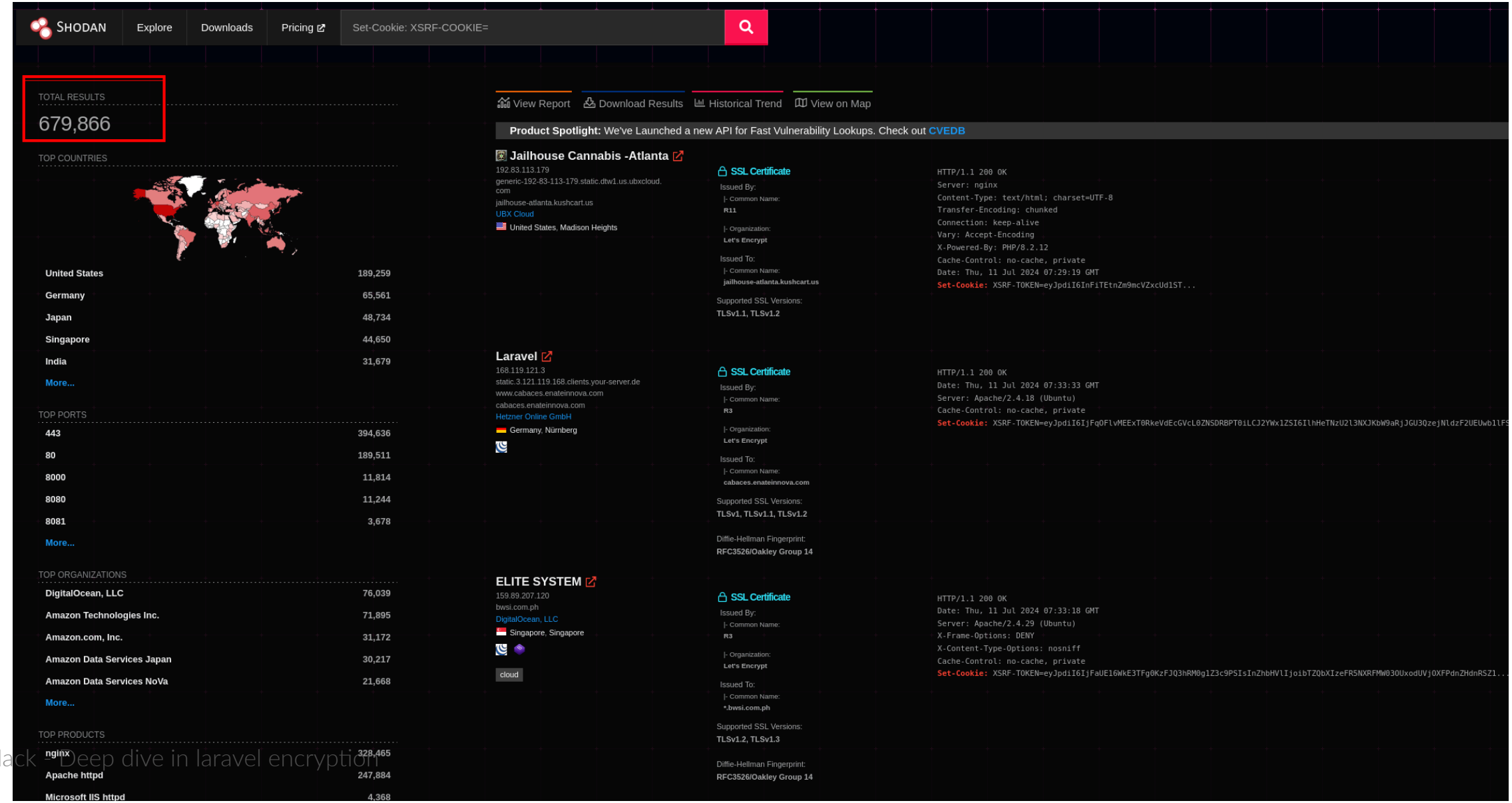
Find Token in the wild

SYNACKTIV

- Starting by exfiltrate all Laravel ciphered object foundable on the internet
- Leveraging Shodan for Discovery, as it indexes publicly exposed server information, including Laravel's `XSRF-TOKEN` and other cookies.
- Simple Pattern Matching for collecting serialized data :
  - `Set-Cookie: XSRF-TOKEN=`
  - `http.title:"Whoops! There was an error" http.status:500`

Let's GO Shodan

SYNACKTIV



Find APP_KEY in the wild

SYNACKTIV

- A significant number are vulnerable to brute-force attacks targeting `APP_KEY`

Description	Details
GitHub Dork	`app_key=base64: AND NOT "app_key=base64:*" AND NOT "app_key=base64:.*"`
Google Dork	`ext:env intext:APP_ENV= | intext:APP_DEBUG= | intext:APP_KEY=`
Known APP_KEY Repository	**BadSecrets** (not fully relevant for our case).
Leaked `.env` Files	Smart person would try to grab `.env` files from our nearly 700k IPs. But **of course**, we don't do that here :)

Find APP_KEY in the wild

SYNACKTIV

- Many `.env` files are accessible across the web, exposing sensitive environment configurations that should remain private
- Some `.env` files still leak through Google and GitHub, often due to accidental publication.
- Global scanning with **wordlists**: Attackers use wordlists to locate `.env` files containing critical secrets like `APP_KEY`s.

```
.env
.env.dev
.env.prod
.env.backup
.env.example
.env.local
...
```

The Danger of Default APP_KEY

SYNACKTIV

- Default `APP_KEY`s in production environments make applications easy targets
- Attackers can leverage Laravel encrypted cookies like `XSRF-TOKEN` to:
  - Reverse-engineer the key
  - Gain access to sensitive data
  - Compromise the system
- Laravel crypto killer was used to bruteforce

Laravel Cryto Killer - bruteforce

SYNACKTIV

- Our tool **Laravel Crypto Killer** was used to brute-force `APP_KEY`
- The option `--cipher_file` can be used to select a list of Laravel ciphered values
- The brute force will be multi threaded to gain sometime, however it is not fully optimized yet
  - the option `--threads` can be used to define more threads
  - The brute force takes us several hours anyway in our case

Laravel Cryto Killer - bruteforce

SYNACKTIV

```
(env) user@poc-laravel:~/Desktop/local-tool/laravel-crypto-killer$ python3 laravel_crypto_killer.py bruteforce -t 3 --cipher_file cipherlist
s/tokens_output.txt
[*] The option --key_file was not defined, using files from the folder wordlists...
  0%|                                                                                                                  | 0/580461 [00:00<?, ?it/s]
[+] It is your lucky day! A key was identified!
Cipher : eyJpdiI6Ikhoa1llQzNadmdNdkdIcmo0RXpMR1E9PSIsInZhbHVlIjoiZmEyV1JpVnBGMkVYckoxQlVnZFM4ckNjd1ZXdHg0WTI3ZFEwb1g3TkdWN205MmQ0bjR2aXdTL05
CU3FQVVk2UStGYWVrc29mOEVOOXVYUS83bTQxWGowaU5VanF3Z1ovRFRCd2ZFaGMrNzdWUkNxbFZhbndFcm82RWNiUkhsK3kiLCJtYWMiOiI1ZGRkY2VhZjU4YzMwZTFlMTViN2ZlZDF
kZjU1M2EyMjk2YmY3YTM2YWExMjZkNmZlMzZlZDdkZmQ0MTFkMGQzIiwidGFnIjoiIn0%3D
Key : SomeRandomStringOf32CharsExactly
[*] Unciphered value
2b0b05fa23ce769182efd20cf7c1df9ae7189729|dWJd3oSa3CNrTgxoH53QYh0OklwQRowbF0syvvbd
[+] It is your lucky day! A key was identified!
Cipher : eyJpdiI6IkRtN1VtdXhBYm94UXdTM1BzOXBDcGc9PSIsInZhbHVlIjoidkp4T1oxbUQxaEI2d1drNUI2MGpSK2VXd2hVMlhwYzRCSGhWNC85bGJFNUt5SmpOWFlHc1FwRkx
sQlZ1SlJjb3dsQkw3YzNjMzBaczBIVGk3RERQUG5pNGZFZ0MySUg3UUFVMiswRlBxdjZnWnV6UzJxQTI2b0Zpa2RqT1Q0dHQiLCJtYWMiOiJiNmFmN2Q4MDQ1OTY0ZjBlZDBhNDM3MTQ
yOGIxNWM4MDRhY2M1MTNlM2NiNzA4MTM5MjYxZDlkZGQxZmQ2MDM0In0%3D
Key : base64:NEMESCXelEv2iYzbgq3N30b9IAnXzQmR7LnSzt70rso=
[*] Unciphered value
9b9c10a2ca741015741d6c60bb1edf582f9bb9a3|u8ON9nBjLq8yxyUDjAm3vGpSpXg7DtXFJ14QoYUB
  0%|                                                                                                  | 550/580461 [00:47<10:30:50, 15.32it/s]
[+] It is your lucky day! A key was identified!
Cipher : eyJpdiI6IkREUHJPc294Y1JPdmpVb1RMZkMwMEE9PSIsInZhbHVlIjoidUNqNFpRb1FKWFh1QTdIbjdLNlZncUcyRnRuT2xoWXp3UmtpNzNwdnoyQy9zWWFYQjVLRmpiMzd
pVEdxQzl0djhXVGR5QzVqNWxic2pxTW01ME1ZUUdlQ1NKL3FiZk8xR0h1cUYwTkM4WXEycFNNQ2hJZmRHbGQyT0xFK3FwS3giLCJtYWMiOiIyOTU3MTQyZWU3YjJlMjlmZDdhMDM2M2M
2ZmM2NzgyNzIwMjQzYmI3M2RiMjdlMjA1MjgwOWIwZmZhNmQ4ZmRjIiwidGFnIjoiIn0%3D
Key : base64:/VR7itPRMggN281EBagG3/F3YV+RaTb8u973wcwHc/U=
[*] Unciphered value
014213f0ee5a59a74bbed8b4dee45df7c08322ec|9NtLlpB3WOCxioLf7vfZXeh4ed88IdodKxcIPFuJ
  0%|                                                                                                  | 797/580461 [01:04<9:35:09, 16.80it/s]
[+] It is your lucky day! A key was identified!
Cipher : eyJpdiI6IkZxTDJVSGxQcjZNcVk0ODhjbEppb0E9PSIsInZhbHVlIjoibUowanFPQnFCd1MyNFZjOFYwUlRTUTlDQllzOTdRK2R6amxTZlFyK0RuZTFLR25JbjBibmFRdFp
hMVBKNmxJOHdseDFRcjlHNkY1bW10aXVwSlFkbVdyRUJkMXpnU1RsSUZlWjgzY1IxZU81VCtMeEJ0cDYrcHI3OVhhajFSTEsiLCJtYWMiOiIyNzQwZWRhNzk2YTg0NDFmOGE2OTQyYTc
zMmIxNzg4YzlkYTkwMzRhMGJkMzJkMmZkZmY5Yzc4ODVhMTBhZjBlIiwidGFnIjoiIn0%3D
Key : base64:W8UqtE9LHZW+gRag78o4BCbN1M0w4HdaIFdLqHJ/9PA=
[*] Unciphered value
4ab8d850859340bf637c74a41096648dfd22a460|Wne2jqunVQJLUWrYHpxUPo7U5rWE2CiMt7F6yMdo
  0%|                                                                                                  | 862/580461 [01:08<9:04:23, 17.74it/s]
```

Laravel Cryto Killer - bruteforce

SYNACKTIV

Laravel Cryto Killer - bruteforce

SYNACKTIV

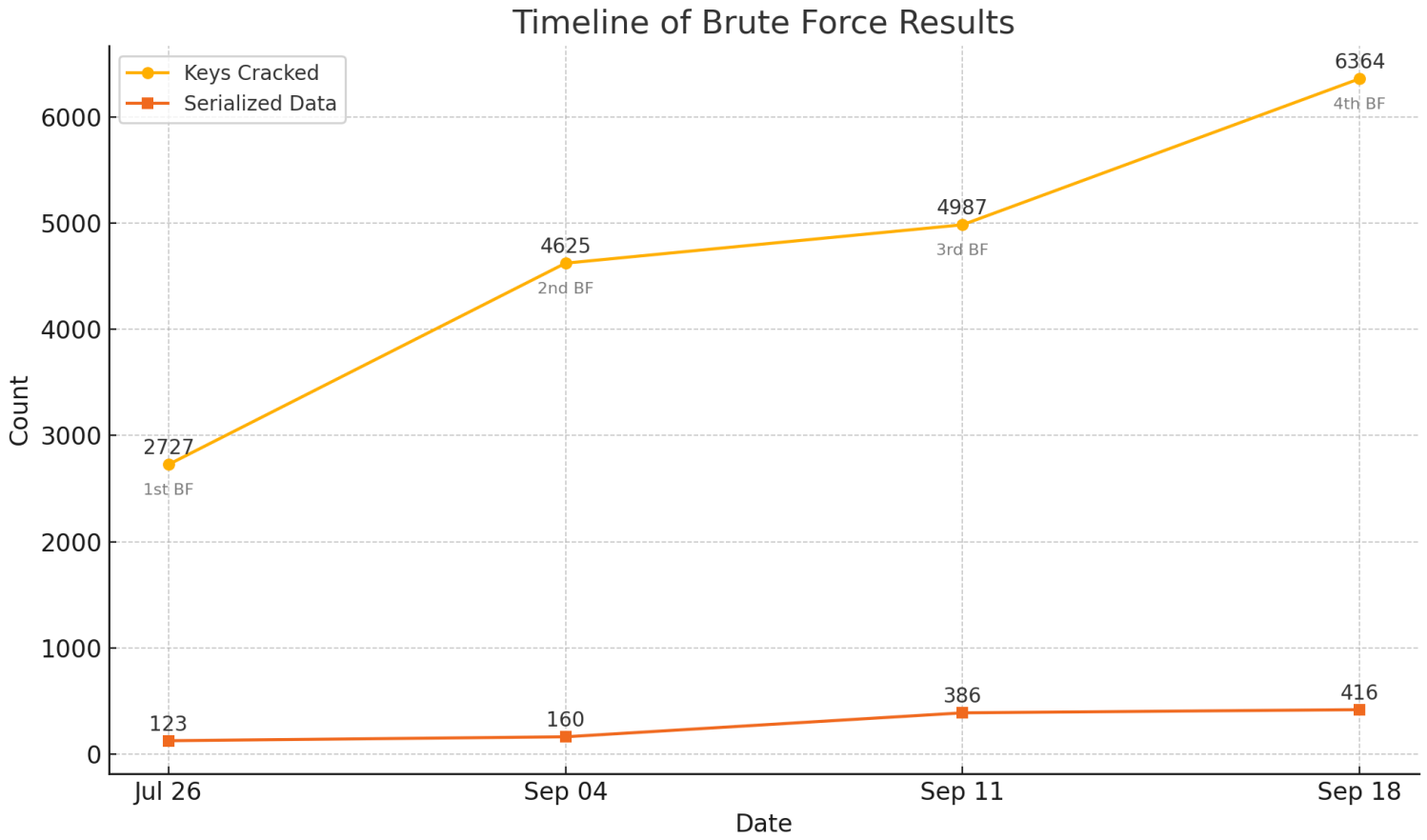
- Here is the result of our forth brute force

```
$ cat results/results_580461_v4.0.json | tail
            "value": "3b4e03323f4fb5af7ed84090e39d311b28149cf0|APV5J6SR7AUyK5Woy0cdHyHOdXb1mXQeDw8wG3SR000f\u000f",
            "cipher": "eyJpdiI6Ilg3L0JvSytwY2EvblFDd1NDcDhBWEE9PSIsInZhbHVlIjoiNEJaSTYzSkRBVXFmYlZzNjBrNW
            J5RTVXR0FpNG5kN25Kb012OFNWUS9pZ2hnNDBxdS9pajFLL2dsbVVHem15QWZpOFdvU1ZSTzRBOU1DR05jSTh0ND
            MvQm5DcFNVQjMzeXRkU3VqT1cySEpVZlpGREhqVHhvNWZFa3pPOGgxN1ciLCJtYWMiOiI5ZjA3YWQyZTFkMWUwYjM3
            ZTAwMWUwYjVjNWQwYzVhMjM5NmY0MGQwMmQyNTdmZDY4NjY0MDJlM2I2YWMyMDM5IiwidGFnIjoiIn0%3D",
            "is_serialized_data": false
        }
    ],
    "command_used": "./laravel_crypto_killer.py bruteforce --cipher_file cipherlists/tokens_output.txt",
    "number_of_ciphers_loaded": 580461,
    "number_of_key_cracked": 6364,
    "number_of_serialized_data": 416
```

Laravel Crypto Killer - bruteforce

SYNACKTIV

- Offline bruteforce of XSRF Token or any serialized value with the help of a custom Wordlist
- Timeline differents actions with brute force for *580461* Ciphers loaded (>1% of all the exposed Laravel instances)



APP_KEYs identified by the attack

SYNACKTIV

Occurrences	Key
561	base64:W8UqtE9LHZW+gRag78o4BCbN1M0w4HdaIFdLqHJ/9PA=
491	base64:SbzM2tzPsCSlpTEdyaju8l9w2C5vmtd4fNAduiLEqng=
313	base64:U29tZVJhbmRvbVN0cmluZ09mMzJDaGFyc0V4YWN0bHk=
216	SomeRandomString
152	base64:RR++yx2rJ9kdxbdh3+AmbHLDQu+Q76i++co9Y8ybbno=
147	SomeRandomStringWith32Characters

APP_KEYs identified by the attack

SYNACKTIV

Occurrences	Key	Solution
561	base64:W8UqtE9LHZW+gRag78o4BCbN1M0w4HdaIFdLqHJ/9PA=	Advanced Stock Management Point of Sale Invoicing Application
491	base64:SbzM2tzPsCSlpTEdyaju8l9w2C5vmtd4fNAduiLEqng=	Frequently used in bootstrap project
313	base64:U29tZVJhbmRvbVN0cmluZ09mMzJDaGFyc0V4YWN0bHk=	base64 value of SomeRandomStringOf32 CharsExactly
216	SomeRandomString	Default APP_KEY on older Laravel Version
152	base64:RR++yx2rJ9kdxbdh3+AmbHLDQu+Q76i++co9Y8ybbno=	Invoice Ninja
147	SomeRandomStringWith32Characters	Was a default Laravel APP_KEY

IOC implication

SYNACKTIV

- Laravel session and `XSRF-TOKEN` cookies are publicly indexed on platforms like Shodan, allowing stealthy `APP_KEY` brute-forcing without direct server interaction.
- Common web ports often bypass firewall filtering.
- With a valid `APP_KEY`, attackers can exploit pre-authenticated `decrypt` calls to compromise the server.

An offline attack is stealthier than other types of attack. (**Winston Churchill**)

Mitigation for Laravel

SYNACKTIV

1. Always generate strong, unique `APP_KEY` (`php artisan key:generate`)
2. Regularly audit applications for weak or default keys and apply a rotate strategy
3. Banish the usage of unserialize in any application based on Laravel
4. Use `encryptString` and `decryptString` instead of `encrypt` and `decrypt`
5. Restrict Access to Environment Files

Conclusion

Conclusion - Our Research

SYNACKTIV

- Analyze of decrypt functions and its flaw
- Analyze of Laravel publicly exposed instances (Token and Key)
- Evaluation for opportunist risk on the wild

Conclusion - Actually

SYNACKTIV

- Mismanaged `APP_KEY`s pose serious security threats
- Proper security practices must be followed:
  - Use strong, unique keys
  - Regular audits
  - Avoid relying on default settings

Conclusion - Future

SYNACKTIV

- File read vulnerabilities will enhanced risk of future exploitation
- Improvement of laravel crypto killer :
  - Improve brute force method for large scaling
  - Add an option to retrieve `XSRF-TOKEN` from an URL or List
- Improvement of wordlist :
  - Optimize open source crawling
  - Update Wordlist for relevant bruteforce (Top 10 Key)
- Improvement for Laravel
  - Deactivate usage serialization by default
  - Communicate about `decrypt` risks
  - Patch pop chains :)

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com

SYNACKTIV

https://github.com/synacktiv/laravel-crypto-killer

Bibliography

SYNACKTIV

- Laravel Crypto Killer Tools : https://github.com/synacktiv/laravel-crypto-killer
- PHPGGC : https://github.com/ambionics/phpggc
- Laravel Encryption documentation : https://laravel.com/docs/11.x/encryption
- Blogpost from Timo Müller about exploits based on APP_KEY : https://mogwailabs.de/en/blog/2022/08/exploiting-laravel-based-applications-with-leaked-app_keys-and-queues/
- CVE-2018-15133 : https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-15133
- Commit removing SESSION_DRIVER=cookie by default : https://github.com/laravel/laravel/commit/eca7bc7d66cef63500a62e8deaa3fce1772f1641

Bibliography

SYNACKTIV

- Patch Snipe-IT : https://github.com/snipe/snipe-it/commit/09abcb44bb303019e43b005a866dfff54a2ede84#diff-9d0a077acda375ef8f06f27e2f823037207bcbaa21c2c87cf4c113a13a3eb0c4L94
- Patch InvoiceNinja : https://github.com/invoiceninja/invoiceninja/commit/d9302021472c3e7e23bac8c3d5fbec57a5f38f0c
- Patch InvoiceShelf : https://github.com/InvoiceShelf/InvoiceShelf/commit/a64701bda570629757d3dd5a9277584fc0aeb34c