



CVE-2025-54068 : Deep dive into Livewire

Pass the salt 2026

2026/07/01

Who are we?



Rémi Matasse
Security Researcher
@Synacktiv



Pierre Martin
Security Researcher
@Depi

Table of content

- Introduction to Livewire
 - Livewire unmarshalling process
 - Synthesizers mechanism
 - First vulnerability : RCE if known APP_KEY
- CVE-2025-54068 : unmarshalling & weak typing
 - Livewire update mechanism
 - Bypass the APP_KEY limitation
 - CVE-2025-54068 : Automate the exploit with Livepyre
- What happened after the publication?
 - Reactions of the community
 - Livewire 4 mitigations
 - CISA flagged the vulnerability
- Conclusion and thoughts

Table of content - common thread



Introduction to Livewire



LIVEWIRE

downloads 85.66 M stable v4.3.1 license MIT

Introduction to Livewire



✓ Full-stack framework used to build real-time features on web-interfaces
build dynamic UI components without leaving PHP

- According to BuiltWith:
 - **660K websites instances of Laravel** currently live (📈 since 2025)
 - Among them **170K are based on Livewire** (📈 since 2025)
- The research we are presenting are based on Livewire 3

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

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

How Livewire works

- 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

```
1 // app/Livewire/Counter.php
2 <?php
3
4 namespace App\Livewire;
5
6 use Livewire\Component;
7
8 class Counter extends Component
9 {
10     public $count = 1;
11
12     public function increment()
13     {
14         $this->count++;
15     }
16
17     public function render()
18     {
19         return view('livewire.counter');
20     }
21 }
```

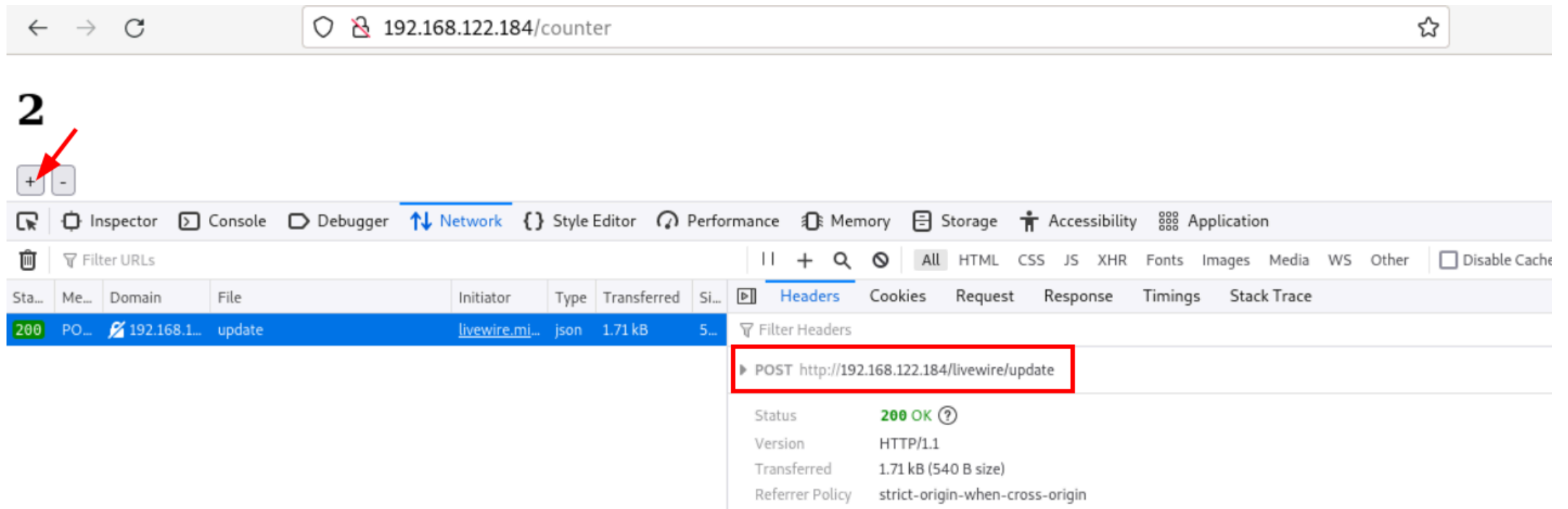
How Livewire works

- 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

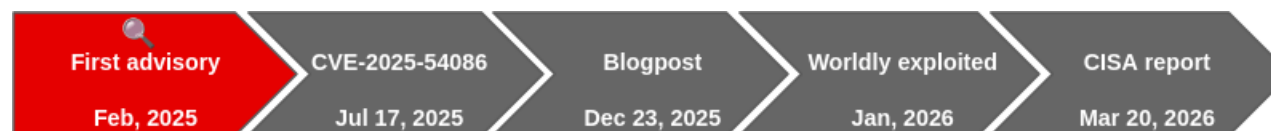
```
1 // routes/web.php
2 <?php
3
4 use Illuminate\Support\Facades\Route;
5 use App\Livewire\Counter;
6
7 Route::get('/counter', Counter::class);
8
```

```
1 // resources/views/livewire/counter.blade.php
2 <div>
3     <h1>{{ $count }}</h1>
4
5     <button wire:click="increment">+</button>
6     <button wire:click="decrement">-</button>
7 </div>
```

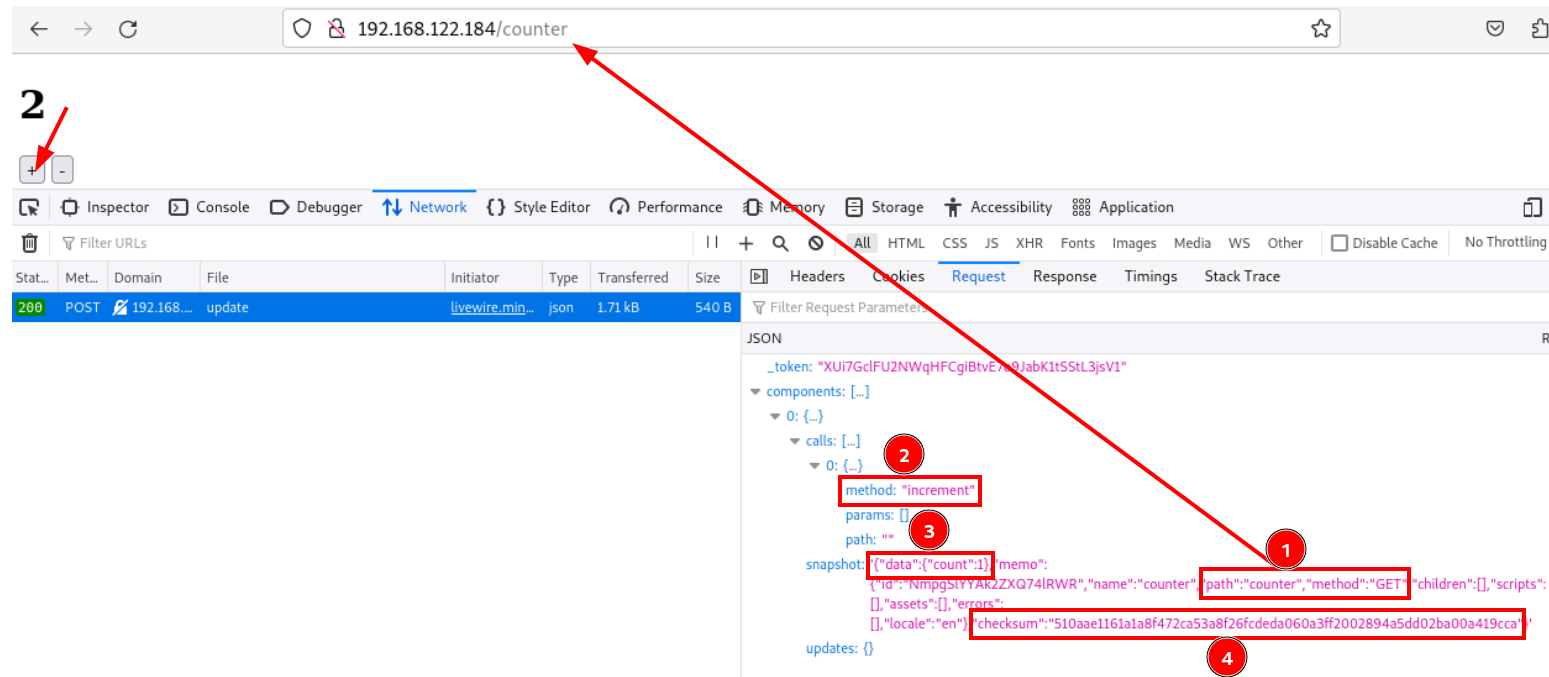
How Livewire works



❗ When interacting with a Livewire 3 component, /livewire/update is called.

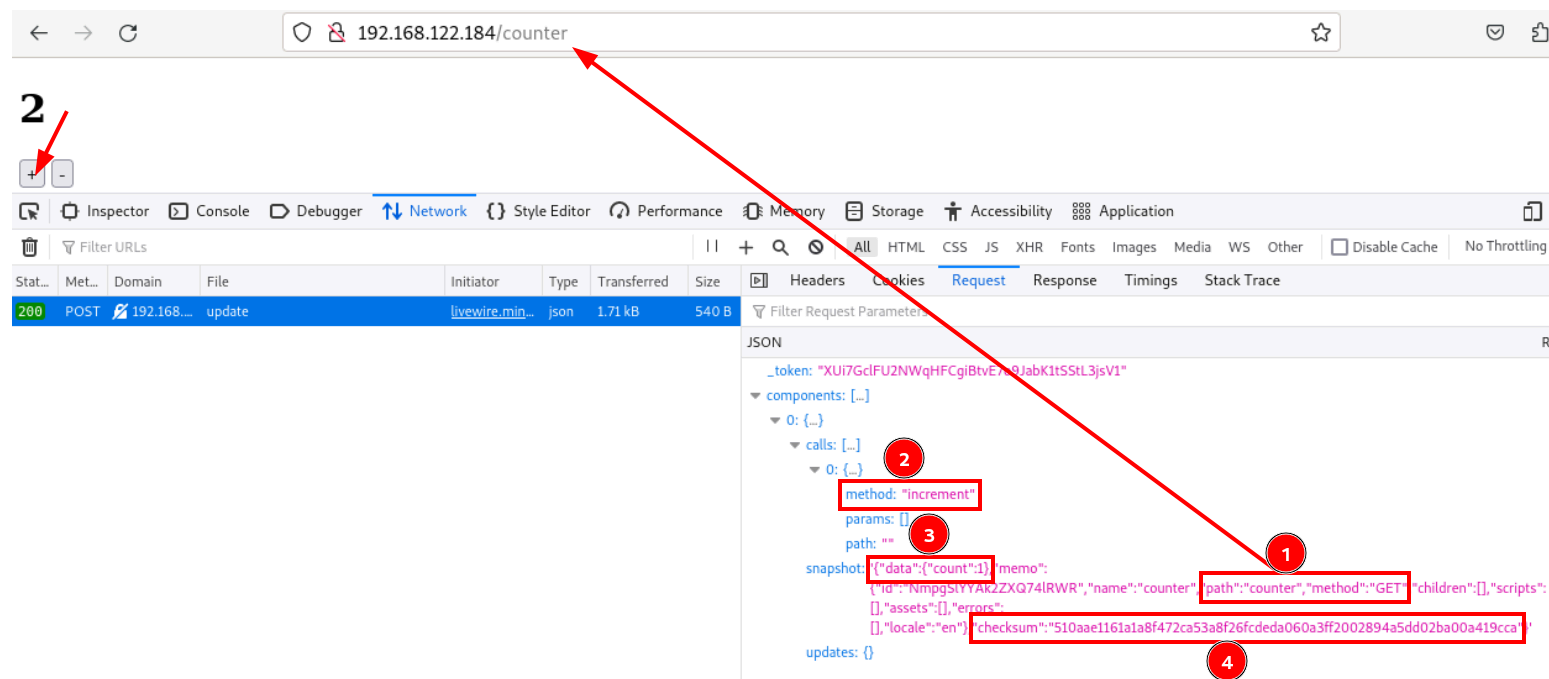


How Livewire works

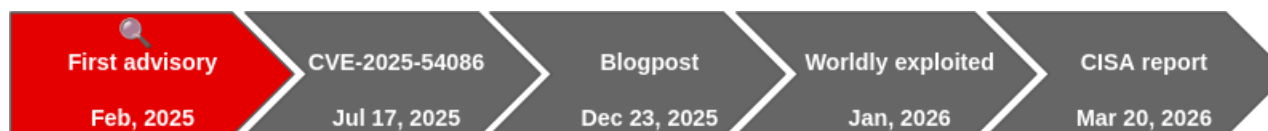


1. `path` and `method` are based on the current page of the request
2. `calls` contain the `method` called on the component, here `increment`

How Livewire works



3. **data** contains all the fields of the component, including its value in database
4. **checksum** validates the **snapshot** 's integrity including **data** , **path** , **method**



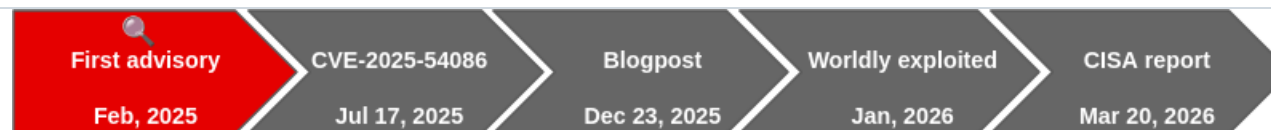
Synthesizers mechanism

- 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

- 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

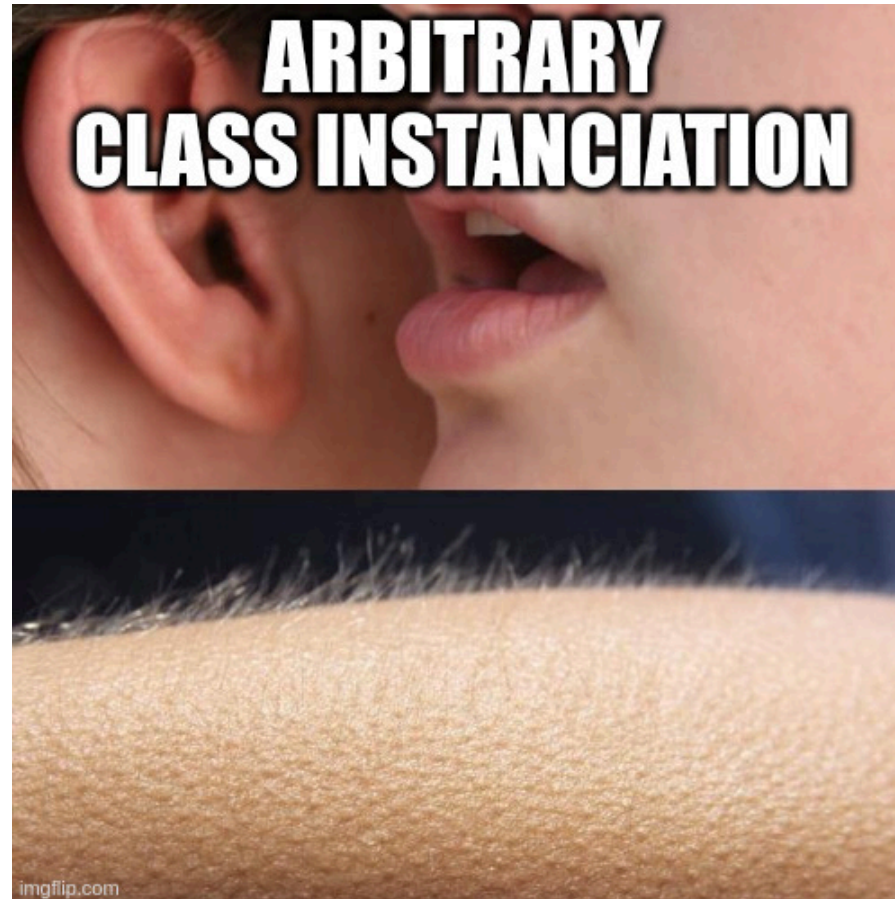
- `$key = 'clctn'` : the key used in JSON component to call a synthesizer
- `$value` : represents the serialized collection data
- `$meta` : array containing metadata
- `->$hydrateChild` : callback used individually to process each elements

```
1 <?php
2
3 namespace Livewire\Mechanisms\HandleComponents\Synthesizers;
4
5 class CollectionSynth extends ArraySynth {
6     public static $key = 'clctn';
7     [...]
8     function hydrate($value, $meta, $hydrateChild) {
9         foreach ($value as $key => $child) {
10             $value[$key] = $hydrateChild($key, $child);
11         }
12         return new $meta['class']($value);
13     }
14 }
```

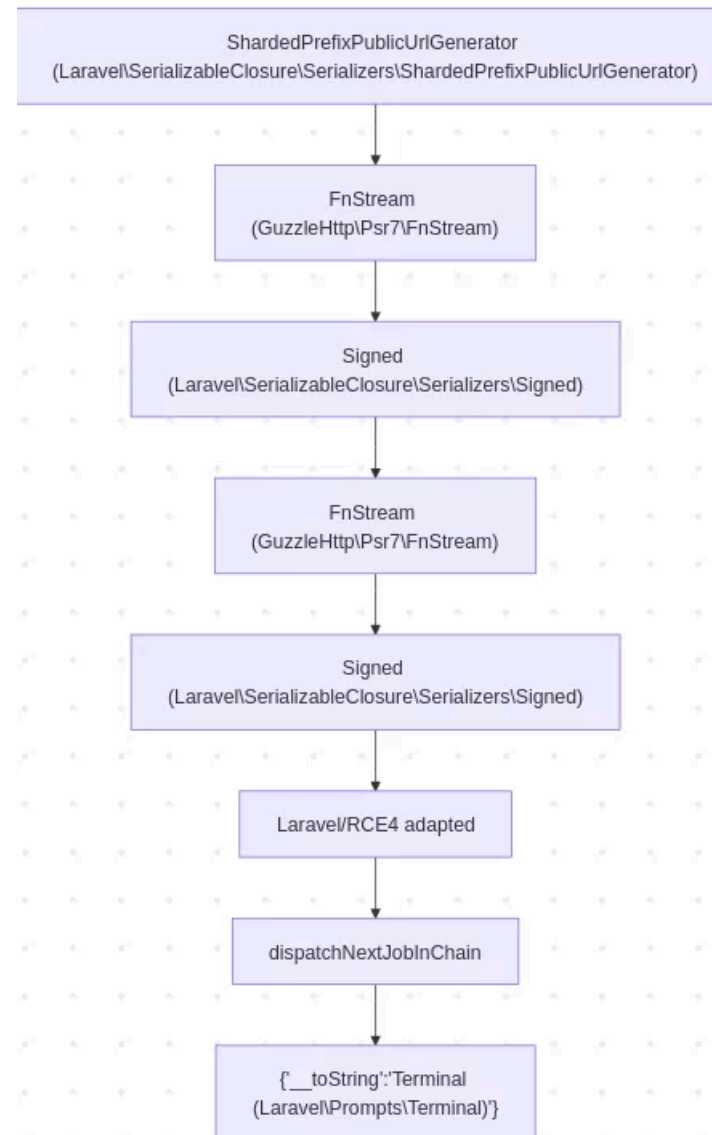
<https://github.com/livewire/livewire/blob/v3.6.4/src/Mechanisms/HandleComponents/Synthesizers/CollectionSynth.php>



Synthesizers mechanism



Chain of instantiation based on dependencies



[DEMO] Exploit an actual project: Snipe-IT

- If you have a valid request from Livewire, you only need the **APP_KEY** to run the exploit
 - the RCE can be played pre-authentication as long as you are in possession of these prerequisites
 - For example, here is a Livewire template valid on Snipe-IT

```
$ cat snipe-it_livewire.json
{
  "_token": "Qyx2gXfZ0zMf6dGoDf7Z2qLhX9PhgzYofE0ZsZ0z",
  "components": [
    {
      "snapshot": "{\"data\":{\"progress\":[],\"memo\":{\"id\":\"Y6a883cdUFy82whZ10JW\",\"name\":\"Importer\",
      \"path\":\"login\",\"method\":\"GET\",\"children\":[],\"scripts\":[],\"assets\":[],\"errors\":[],
      \"locale\":\"en\",\"checksum\":\"6681a672a6d6927531f7df23c775bc1ced2479b48c3fb0c23f3ed335a0011008\"},
      \"updates\": {},
      \"calls\": []
    }
  ]
}
```



Checksum verification

```
1  <?php
2  namespace Livewire\Mechanisms\HandleComponents;
3  use function Livewire\trigger;
4
5  class Checksum {
6
7      static function verify($snapshot) {
8          ① $checksum = $snapshot['checksum'];
9          unset($snapshot['checksum']);
10         trigger('checksum.verify', $checksum, $snapshot);
11         ② if ($checksum !== $comparator = self::generate($snapshot)) {
12             trigger('checksum.fail', $checksum, $comparator, $snapshot);
13         ③ throw new CorruptComponentPayloadException;
14         }
15     }
16 }
```

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

Full details in confs Nullcon/hack.lu 2025



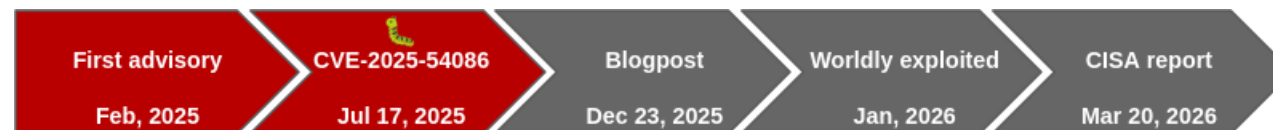
CVE-2025-54068 : unmarshalling & weak typing



Composition of an update request - calls

The **calls** field contains the functions called on the component

```
1 POST /livewire/update HTTP/1.1
2 Host: 192.168.122.184
3 [...]
4
5
6 {
7   "_token": "jMEN2kTQRrwSA5CgH5y8WWqbCpdb4Lx4iBznnlFD",
8   "components": [
9     {
10       "snapshot": "{ \"data\": { \"count\": 3 }, \"memo\": { \"id\": \"Y6a883cdUFy82whZ10JW\", \"name\": \"counter\", \"path\": \"counter\", \"method\": \"GET\", \"children\": [], \"scripts\": [], \"assets\": [], \"errors\": [], \"locale\": \"en\", \"checksum\": \"f8[...]7c\" }, \"updates\": { }, \"calls\": [
17         {
18           \"path\": \"\",
19           \"method\": \"increment\",
20           \"params\": [
21
22         ]
23       }
24     ]
25   }
26 }
```



Composition of an update request - snapshot

- The **snapshot** field contains the context of the request :
 - The value of the component as a JSON object in **data**
 - The **path** and **method** of the HTTP request
 - Other information related to **children** , **scripts** , **assets** linked to the request
 - The full content of the **snapshot** is signed by a checksum to protect against request tempering

```
1  POST /livewire/update HTTP/1.1
2  Host: 192.168.122.184
3  [...]
4
5
6  {
7      "_token": "jMEN2kTQRrwSA5CgH5y8WwqbCpdb4Lx4iBznnlFD",
8      "components": [
9          {
10             "snapshot": "{ \"data\": { \"count\": 3 }, \"memo\": { \"id\": \"Y6a883cdUFy82whZ10JW\", \"name\": \"counter\", \"path\": \"counter\", \"method\": \"GET\", \"children\": [], \"scripts\": [], \"assets\": [], \"errors\": [], \"locale\": \"en\", \"checksum\": \"f8[...]7c\" }, \"updates\": {}, \"calls\": [ { \"path\": \"\", \"method\": \"increment\", \"params\": [] } ] } }"
11             { \"id\": \"Y6a883cdUFy82whZ10JW\",
12               \"name\": \"counter\", \"path\": \"counter\",
13               \"method\": \"GET\", \"children\": [],
14               \"scripts\": [], \"assets\": [], \"errors\": [],
15               \"locale\": \"en\", \"checksum\": \"f8[...]7c\" },
16             \"updates\": {},
17             \"calls\": [
18                 {
19                     \"path\": \"\",
20                     \"method\": \"increment\",
21                     \"params\": []
22                 }
23             ]
24         }
25     ]
26 }
```



Prerequisites to exploit - APP_KEY

- Exploit prerequisites inside the `snapshot` field :
 - **Laravel `APP_KEY` required to sign the `snapshot` checksum**
 - A valid Livewire request



Composition of an update request - updates

- The **updates** field contains the fields and values to update on the component
 - No limitation on the number of fields to update
 - No control on the type of the data most of the time
 - Most of the time overlooked, the data is usually updated via function in **calls**

```
1 POST /livewire/update HTTP/1.1
2 Host: 192.168.122.184
3 [...]
4
5
6 {
7   "_token":"jMEN2kTQRrSA5CgH5y8WwqbCpdb4Lx4iBznnlFD",
8   "components":[
9     {
10       "snapshot":{"data":{"count":3},"memo":
11         {"id":"Y6a883cdUFy82whZ10JW",
12           "name":"counter","path":"counter",
13           "method":"GET","children":[],
14           "scripts":[],"assets":[],"errors":[],
15           "locale":"en","checksum":"f8[...]7c"},
16       "updates":{},
17       "calls":[
18         {
19           "path":"",
20           "method":"increment",
21           "params":[]
22         }
23       ]
24     }
25   ]
26 }
```



Prerequisites to exploit - CVE-2025-54068

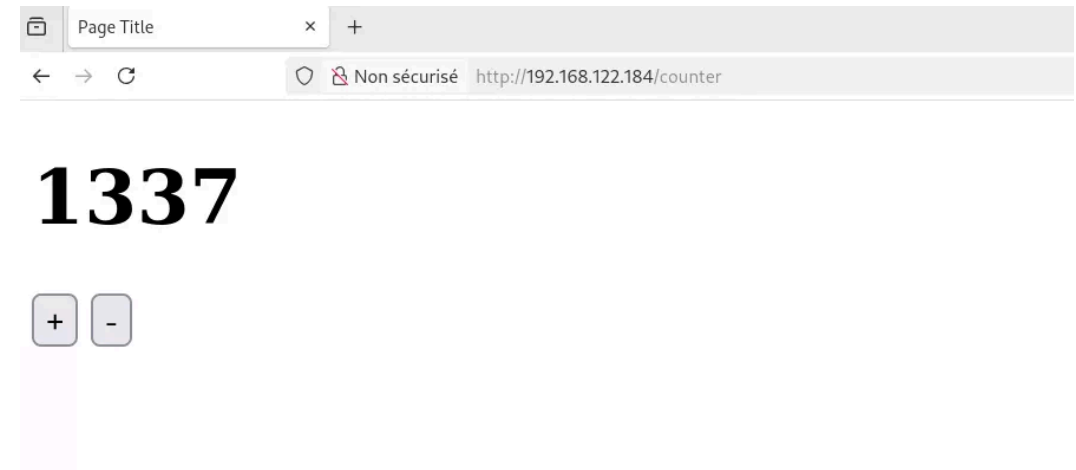
- Exploit prerequisites inside the `updates` field :
 - ~~Laravel APP_KEY~~ required to ~~to sign the~~ `snapshot checksum`
 - A valid Livewire request



Update mechanism - weak typing

The content of the snapshot can be updated directly in the **updates** field :

```
1 POST /livewire/update HTTP/1.1
2 Host: 192.168.122.184
3 Content-Length: 407
4 Cookie: XSRF-TOKEN=eyJ[...]%3D
5
6 {
7   "_token": "KAzJ4mh08NzK8hMkAPjs1aNo6hG2w740HoBDzSzA",
8   "components": [
9     {
10      "snapshot": "{\"data\":{\"count\":1},\"memo\":
11 {\"id\":\"4TRWeXVBaMBRhs1VgVi\",\"name\":\"counter\",
12 \"path\":\"counter\",\"method\":\"GET\",\"children\":[],
13 \"scripts\":[],\"assets\":[],\"errors\":[],\"locale\":\"en\",
14 \"checksum\":\"9bfe798289569477fcf865a952a4a8ddeb3c4150df56b4b404e8df400d0aa0be\"}",
15      "updates": {
16        "count": 1337
17      },
18      "calls": []
19    }
20 ]
21 }
```



Update mechanism - weak typing

However, Livewire components are prone to weak typing :

```
1 POST /livewire/update HTTP/1.1
2 Host: 192.168.122.184
3 Content-Length: 407
4 Cookie: XSRF-TOKEN=eyJ[...]%3D
5
6 {
7   "_token": "KAzJ4mh08NzK8hMkAPjs1aNo6hG2W740HoBDzSzA",
8   "components": [
9     {
10      "snapshot": "{\"data\":{\"count\":1},\"memo\":",
11      {"id\":\"4TRWeXVBaMBrHslVgVi\",\"name\":\"counter\",
12      \"path\":\"counter\",\"method\":\"GET\",\"children\":[]",
13      \"scripts\":[],\"assets\":[],\"errors\":[],\"locale\":\"en\"},
14      \"checksum\":\"9bfe798289569477fcf865a952a4a8ddeb3c4150df56b4b404e8df400d0aa0be\"}",
15      "updates": {
16        "count": "Wait.. I should be an integer"
17      },
18      "calls": []
19    }
20 ]
21 }
```



Vulnerability analysis - controlled data

- When updating a value, **snapshot** data will be sent to the **hydrateForUpdate** :
 - \$raw** : Value of the data already defined in the snapshot
 - \$path** : Name of the component, in our case it will be counter
 - \$value** : Value defined inside the updates field
 - \$context** : Global Livewire context

```
1 <?php
2
3 class HandleComponents extends Mechanism
4 {
5     protected function hydrateForUpdate($raw, $path, $value, $context)
6     {
7         $meta = $this->getMetaForPath($raw, $path); // Synthesizer data?
8         // If we have meta data for this property, we use that to get a synth
9         if ($meta) {
10             return $this->hydrate([$value, $meta], $context, $path);
11         }
12         [...]
13     }
14
15     protected function getMetaForPath($raw, $path)
16     {
17         [...]
18         [$data, $meta] = Utils::isSyntheticTuple($raw) ? $raw : [$raw, null];
19         [...]
20         return $meta;
21     }
22 }
23
24 }
```



Vulnerability analysis - Synthesizers

- `isSyntheticTuple` checks if `$payload` (`snapshot` data element) matches the following structure:
 - It is an array
 - It contains exactly 2 elements
 - The second element is an array containing the key 's'

```
1 <?php
2
3 class BaseUtils
4 {
5     static function isSyntheticTuple($payload) {
6         return is_array($payload)
7             && count($payload) === 2
8             && isset($payload[1]['s']);
9     }
10 }
```



Vulnerability analysis - Synthesizers

- As an example, the following **snapshot** contains the element **stdClass** which will be considered as a **std** synthesizer

```
1 {
2   "data": {
3     "count": 1,
4     "stdClass": [
5       {
6         "foo": "bar"
7       },
8       {
9         "s": "std"
10      }
11    ]
12  },
13  "memo": {
14    "id": "f4Ed5D3MrzEN48IeQyt0",
15    "name": "counter",
16    "path": "counter",
17    "method": "GET",
18    [...]
19  },
20  "checksum": "ed40577cb46a002e...6f823f20bc315660847958bc1ca"
21 }
```

Vulnerability analysis - Synthesizers

- It is possible to cast elements as arrays inside the `updates` field
- It is also possible to add as many sub-arrays as we like in the updated component

```
1 POST /livewire/update HTTP/1.1
2 Content-Length: 407
3
4 {
5   "_token": "KAZJ4mh08NzK8hMkAPjslaNo6hG2W740HoBDzSZA",
6   "components": [
7     {
8       "snapshot": "{\"data\":{\"count\":1},[...]}",
9       "updates": {
10         "count": []
11       },
12       "calls": []
13     }
14   ]
15 }
16
17 HTTP/1.1 200 OK
18 Set-Cookie: [...]joiIn0%3D; .....; path=/; samesite=lax
19
20 {
21   "components": [
22     {
23       "snapshot": "{\"data\":{\"count\":[[],{\"s\":\"arr\"}]}"
24     },
25     [...]
26   ]
27 }
```

Vulnerability analysis - The logical bug

\$raw : Value of the data already defined in the snapshot

\$value , **\$tuple** : Value defined inside the updates field

1. If **\$value** is an **array** , the value from **updates** will be passed to **hydrate** (weak typing)
2. The second element value ' s s' of **\$tuple** from **updates** is used as the synthesizer
3. **Since the hydrate is recursive, any child of the array will can be considered as a synthesizer**

```
1 <?php
2
3 class HandleComponents extends Mechanism
4 {
5     protected function hydrateForUpdate($raw, $path, $value, $context)
6     {
7         ① $meta = $this->getMetaForPath($raw, $path);
8         return $this->hydrate([$value, $meta], $context, $path);
9     }
10
11     protected function hydrate($valueOrTuple, $context, $path)
12     {
13         [...]
14
15         ② [$value, $meta] = $tuple;
16         $synth = $this->propertySynth($meta['s'], $context, $path);
17
18         ③ return $synth->hydrate($value, $meta, function ($name, $child)
19             use ($context, $path) {
20                 return $this->hydrate($child, $context, "{$path}.{$name}");
21             });
22     }
23
24 }
```

Vulnerability analysis - The logical bug

\$raw : Value of the data already defined in the snapshot

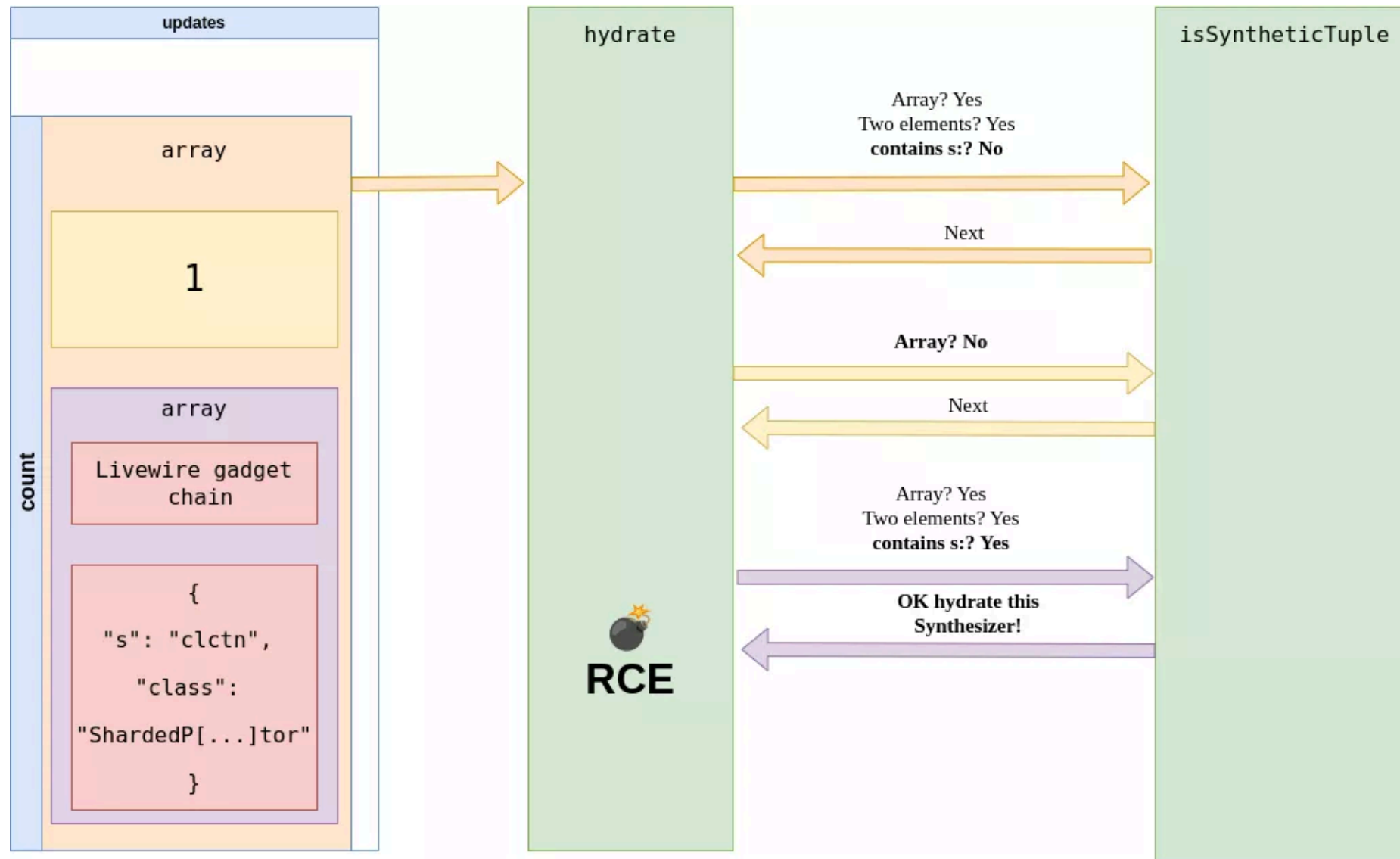
\$value , **\$tuple** : Value defined inside the updates field

1. If **\$value** is an **array** , the value from **updates** will be passed to **hydrate** (weak typing)
2. The second element value ' s s' of **\$tuple** from **updates** is used as the synthesizer
3. **Since the hydrate is recursive, any child of the array will can be considered as a synthesizer**

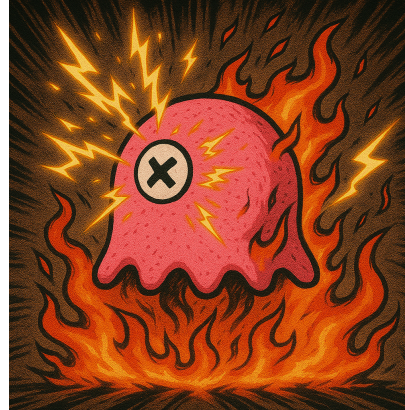
```
1 <?php
2
3 class HandleComponents extends Mechanism
4 {
5     protected function hydrateForUpdate($raw, $path, $value, $context)
6     {
7         ① $meta = $this->getMetaForPath($raw, $path);
8         return $this->hydrate([$value, $meta], $context, $path);
9     }
10
11     protected function hydrate($valueOrTuple, $context, $path)
12     {
13         [...]
14
15         ② [$value, $meta] = $tuple;
16         $synth = $this->propertySynth($meta['s'], $context, $path);
17
18         ③return $synth->hydrate($value, $meta, function ($name, $child)
19         use ($context, $path) {
20             return $this->hydrate($child, $context, "{$path}.{ $name}");
21         });
22     }
23
24 }
```



Vulnerability analysis - The logical bug



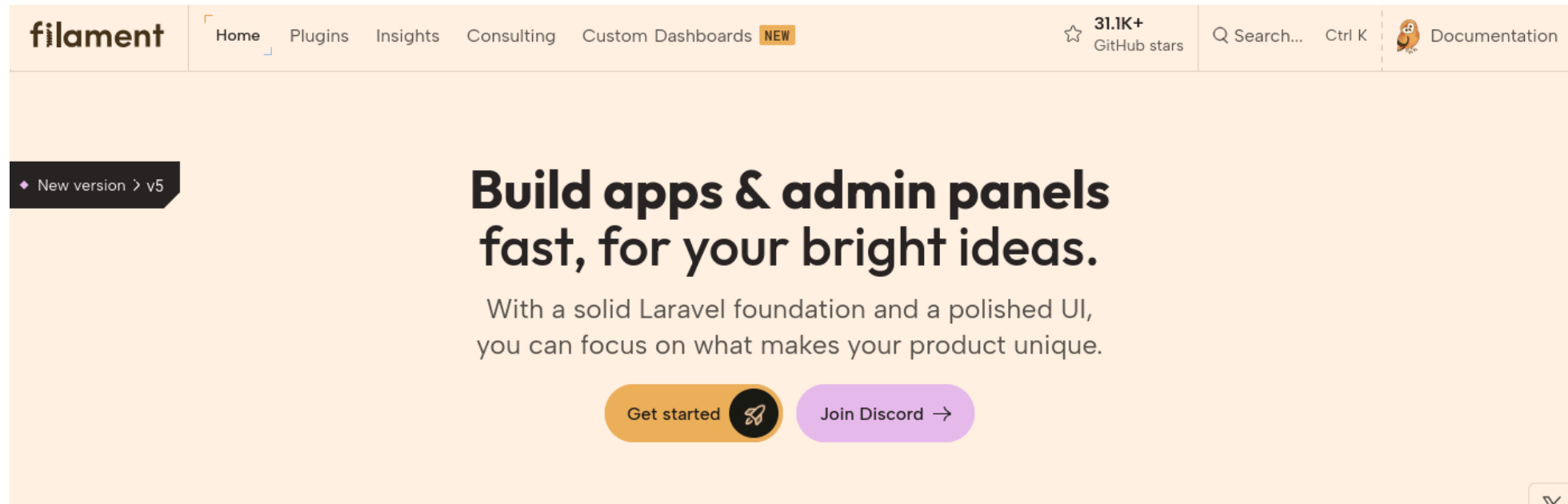
[DEMO] A new exploitation tool - Livepyre



- Without APP_KEY
 - Exploit **CVE-2025-54068**, the only requirement is the URL of the vulnerable application
- With the APP_KEY
 - Exploits the **snapshot**, allowing you to get RCE on applications based on Livewire v3.



[DEMO] A new exploitation tool - Filament



- Filament is one of the most used administration panels
- The default login mechanism was based on Livewire 3 in Filament 3.



What happened after the publication?



Reactions - The patched

Most developers were not impacted since they patched on the release of the CVE

✓ Running composer audit or dependabot regularly will prevent most of the critical vulnerabilities exploited in the wild

 · 1y ago
Great, now I have to work on a Saturday
⬆ 34 ⬇

 · 1y ago
It Will wait until the monday
⬆ 8 ⬇

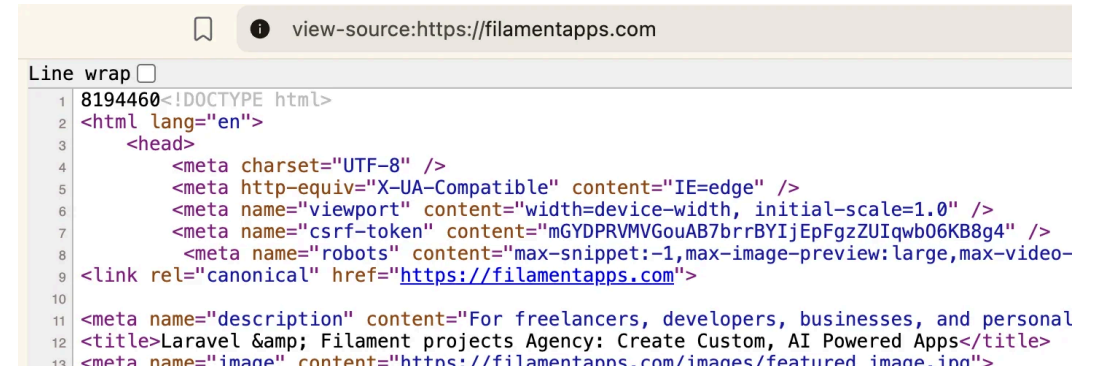
 · 1y ago
I heard the hackers all unionized and only work 9-5 Mon-Fri now, you should be fine to wait until Monday, just come in at 8 maybe to beat them to it!
⬆ 4 ⬇

 · 1y ago
Hooray for dependabot, I've already deployed this fix
⬆ 20 ⬇



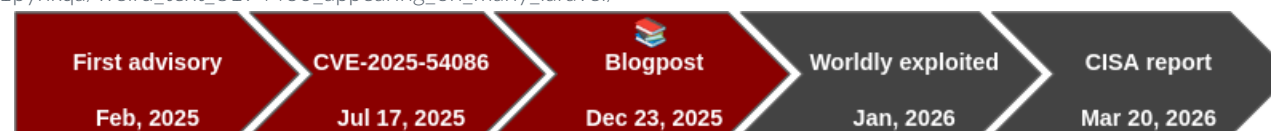
Reactions - The backdoored

Some developers did not apply patches in time, most of them took it as a way to improve



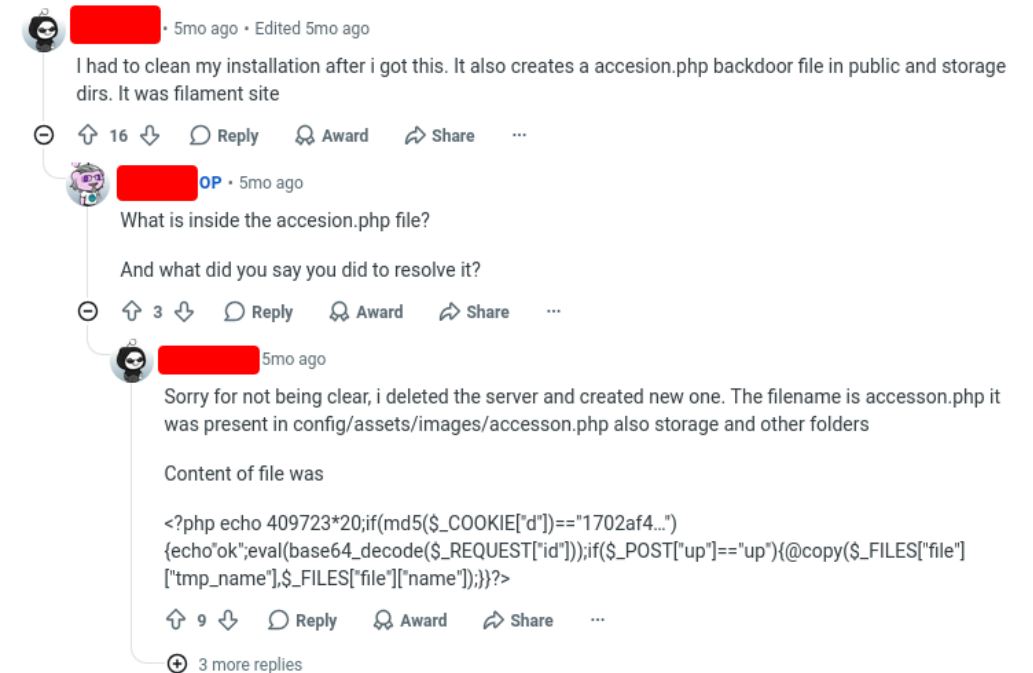
```
Line wrap
1 8194460<!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta name="csrf-token" content="mGYDPRVMVGouAB7brBYIjEpFgzZUIqwb06KB8g4" />
8     <meta name="robots" content="max-snippet:-1,max-image-preview:large,max-video-
9 <link rel="canonical" href="https://filamentapps.com">
10
11 <meta name="description" content="For freelancers, developers, businesses, and personal
12 <title>Laravel & Filament projects Agency: Create Custom, AI Powered Apps</title>
13 <meta name="image" content="https://filamentapps.com/images/featured_image.png">
```

https://www.reddit.com/r/laravel/comments/1pyhkqa/weird_text_8194460_appearing_on_many_laravel/



Reactions - The backdoored

Some developers did not apply patches in time, most of them took it as a way to improve



https://www.reddit.com/r/laravel/comments/1pyhkqa/weird_text_8194460_appearing_on_many_laravel/



Reactions - The backdoored

Some developers did not apply patches in time, most of them took it as a way to improve

⚠ Many documented exploitation mentioned forgotten application

←  r/laravel • 12d ago

An old app with a livewire dependency that hadn't been updated in 1-2 years lead to my entire VPS being compromised

Discussion

First time this has happened to me. I have a single VPS where I was hosting about a dozen personal projects.

One of them I hadn't touched in 1-2 years.

This afternoon I got a string of emails telling me about AWS support tickets being opened up to gain production access to AWS SES, DKIMs being approved in multiple new zones for one of my SES identities, and then that 57,000 emails had been sent in less than a day (typical weekly traffic is ~13).

So after scrambling around, it seems like my AWS SES access key was leaked. I disabled it, then started trying to figure out how the hell it got leaked.

The only place it's stored is in one of my project's .env, which is managed through Laravel Forge's .env var panel.

So I went ahead and had codex SSH into my server and try to find any evidence of compromise, and yeah, my whole server was fucked. Multiple malicious processes running, open connections to external IPs, a bunch of php files that were generated over the last couple of weeks (not by me), all pointing to the old project I hadn't updated.

Turns out the issue was an old livewire vulnerability, attackers somehow exploited it to upload (and execute? I don't fully understand how they did it) arbitrary php files.

So they had access to everything, including all my .env vars across all my apps.

Went ahead and nuked the server, and I'm now running around rolling all the keys.

Fortunately, the damage wasn't too bad before I caught it.

Valuable lesson learned today - even if your side projects are dead, keep them regularly updated.

I don't really want to give up on the idea of hosting all my dumb projects on a single VPS, but this has now given me some pause. Isolating them would have mitigated this.

https://www.reddit.com/r/laravel/comments/1trkw4i/an_old_app_with_a_livewire_dependency_that_hadnt/

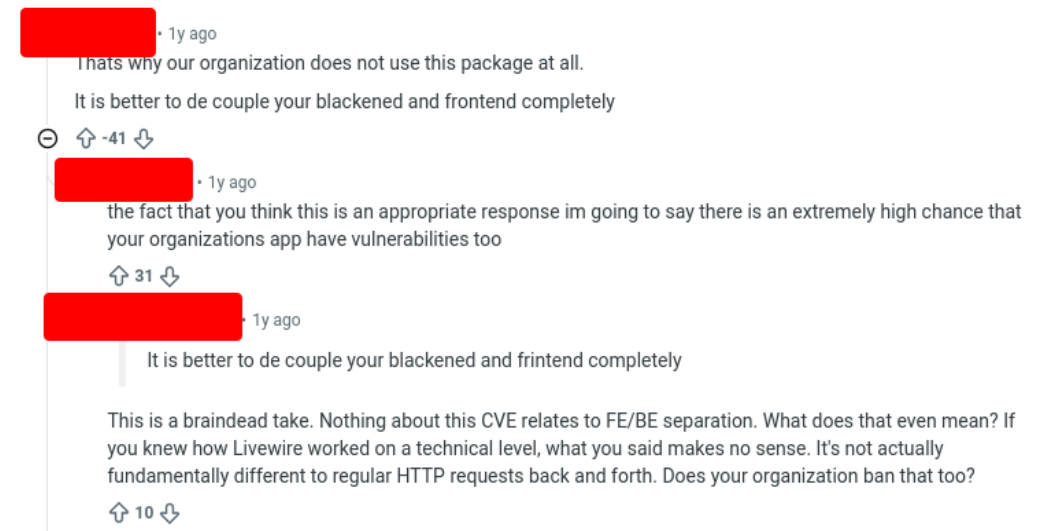
84 43 Share



Reactions - The software hater

Some other developers blamed everything directly on Livewire

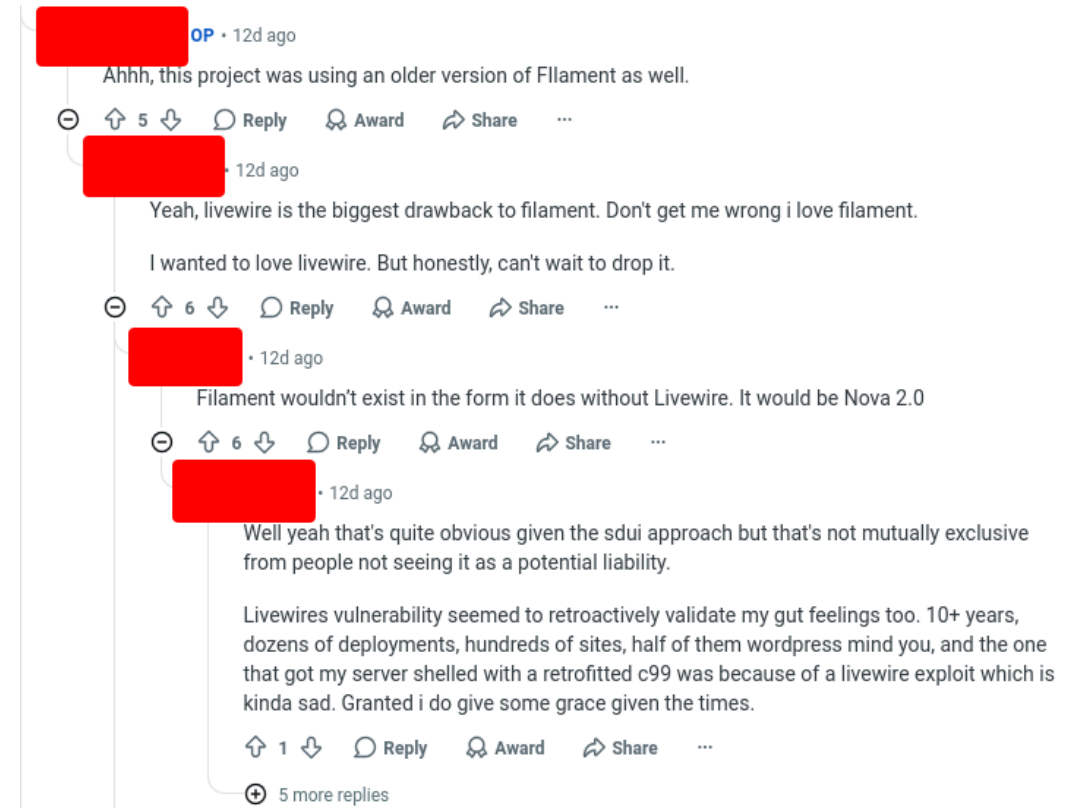
❗ Blaming the technology instead of applying patches won't save you from 1days



Reactions - The software hater

Some other developers blamed everything directly on Livewire

❗ Blaming the technology instead of applying patches won't save you from 1days



How Livewire 4 evolved after this issue?

When we disclosed the bugs in 2025, only the CVE was considered harmful by Livewire team

- In February 2026, they changed their mind and added a deny-list
- So now, in-depth security measures are applied even for the `APP_KEY` leak scenario

```
1 // src/Mechanisms/HandleComponents/SecurityPolicy.php
2 <?php
3 namespace Livewire\Mechanisms\HandleComponents;
4 class SecurityPolicy
5 {
6     protected static $deniedClasses = [
7         // Known serialization gadgets
8         'Illuminate\Broadcasting\PendingBroadcast',
9         'Illuminate\Broadcasting\BroadcastEvent',
10
11         // Known exploit chain gadgets
12         'GuzzleHttp\Psr7\FnStream',
13         'League\Flysystem\UrlGeneration\
14         ShardedPrefixPublicUrlGenerator',
15         'Laravel\Prompts\Terminal',
16     ];
```

<https://github.com/livewire/livewire/pull/9961>

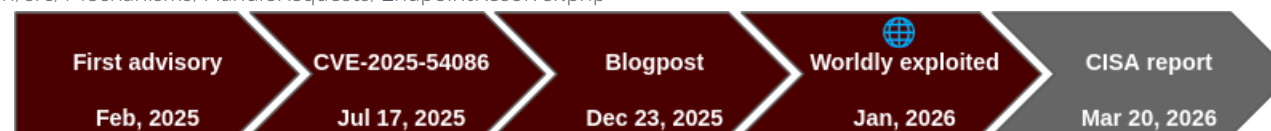


How Livewire 4 handles this issue?

- Livewire endpoint was `/livewire/update` by default on Livewire 3
 - It is now hosted on `/livewire-<hash_based_on_APP_KEY>/update`
 - Using the `APP_KEY` to generate a publicly known route might cause issues

```
1 // src/Mechanisms/HandleRequests/EndpointResolver.php
2 <?php
3
4 namespace Livewire\Mechanisms\HandleRequests;
5
6 class EndpointResolver
7 {
8     public static function prefix(): string
9     {
10         $hash = substr(hash('sha256', config('app.key') .
11                             'livewire-endpoint'), 0, 8);
12
13         return '/livewire-' . $hash;
14     }
15
16     /**
17      * Get the path for the update endpoint.
18      */
19     public static function updatePath(): string
20     {
21         return static::prefix() . '/update';
22     }
23 }
```

<https://github.com/livewire/livewire/blob/main/src/Mechanisms/HandleRequests/EndpointResolver.php>



Last update, the CVE got flagged by the CISA

During march 2026, the CVE-2025-54068 has been flagged by the American's Cyber Defense Agency as a known exploited vulnerability.

It usually occurs when a vulnerability is actively exploited by APTs.

ALERT

CISA Adds Five Known Exploited Vulnerabilities to Catalog

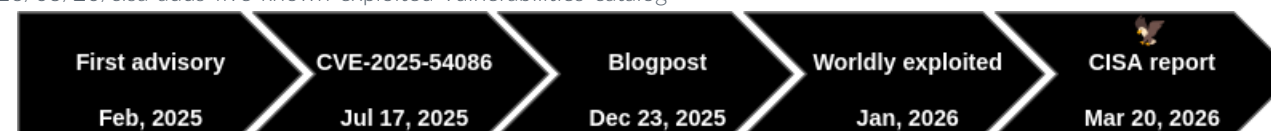
Release Date: March 20, 2026

CISA has added five new vulnerabilities to its [Known Exploited Vulnerabilities \(KEV\) Catalog](#), based on evidence of active exploitation.

- [CVE-2025-31277](#) ☐ Apple Multiple Products Buffer Overflow Vulnerability
- [CVE-2025-32432](#) ☐ Craft CMS Code Injection Vulnerability
- [CVE-2025-43510](#) ☐ Apple Multiple Products Improper Locking Vulnerability
- [CVE-2025-43520](#) ☐ Apple Multiple Products Classic Buffer Overflow Vulnerability
- [CVE-2025-54068](#) ☐ Laravel Livewire Code Injection Vulnerability

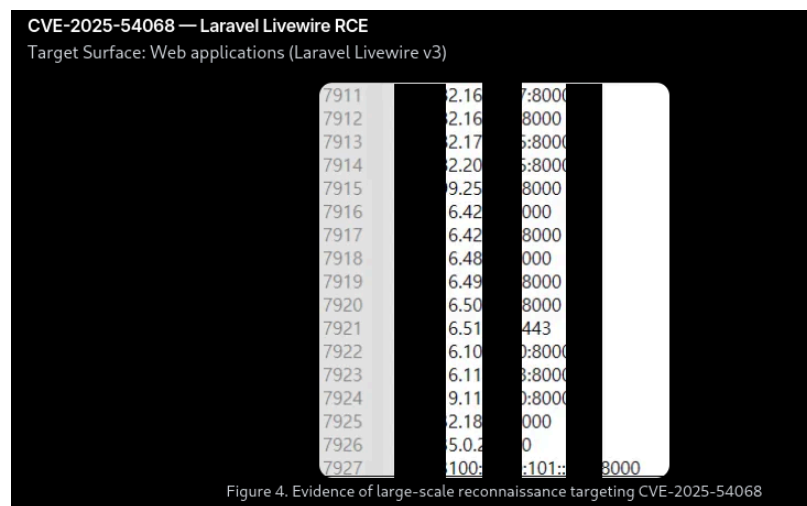
These types of vulnerabilities are frequent attack vectors for malicious cyber actors and pose significant risks to the federal enterprise.

<https://www.cisa.gov/news-events/alerts/2026/03/20/cisa-adds-five-known-exploited-vulnerabilities-catalog>



Last update, the CVE got flagged by the CISA

Exploits based on CVE-2025-54068
attributed to MuddyWater (Iranian APT)



Vulnerability Exploitation and Reconnaissance Strategy

The campaign demonstrated a coordinated approach combining large-scale reconnaissance with rapid weaponization of newly disclosed high-risk vulnerabilities.

The actor conducted extensive scanning across more than 12,000 internet-exposed systems by leveraging at least five distinct CVEs, targeting a wide range of platforms, including:

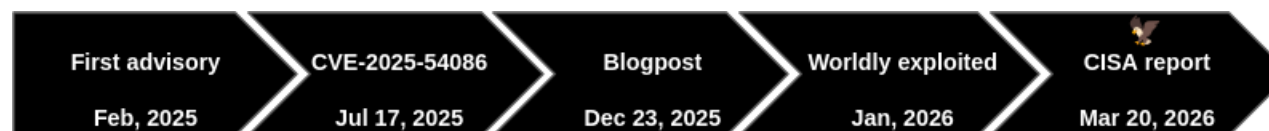
- Web applications
- Email servers
- IT management and monitoring systems
- Workflow and automation platforms

Rather than exploiting all identified targets, the operation followed a two-phase model: broad reconnaissance to identify vulnerable systems, followed by selective exploitation of high-value targets.

This transition from indiscriminate scanning to targeted intrusion reflects a structured and resource-efficient operational strategy.

The scale and organization of scanning activity—combined with vulnerability-specific target grouping—indicate an automated and continuously operating reconnaissance capability.

<https://oasis-security.io/blog/260414-Iran>



Conclusion

Conclusion - Research full of surprises

- Try harded 2 months on the exploitation chain
 - Almost missed the pre-authenticated chain
 - Close to give up several times before success
- Livewire patched the CVE pretty quickly
- We did not think that the rest of the payload would get patched
- We also did not expect this bug to get flagged by the CISA

Conclusion - How do I patch?

- First make sure you never run vulnerable outdated dependencies in your PHP projects

```
$ composer audit
```

- Make sure you generated a unique `APP_KEY`

```
$ php artisan key:generate
```

❗ If your server was exploited by CVE-2025-54068, make sure to regenerate a new APP_KEY!

Conclusion - More details?

- There is so much more to say about this research



 If you want more information please check our blogpost, and other talks!



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



<https://x.com/synacktiv>



<https://synacktiv.com>