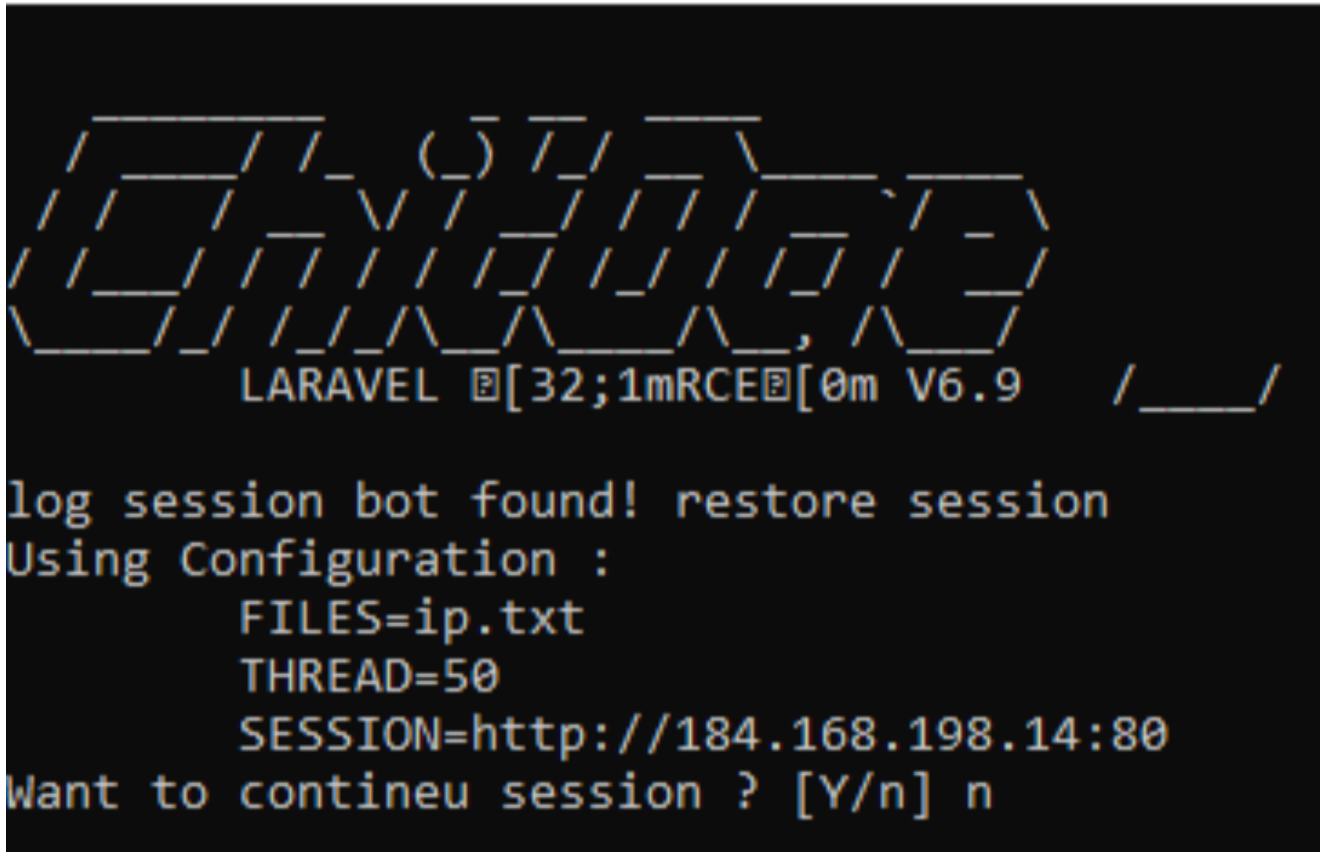


Laravel Apps Leaking Secrets

thedfirreport.com/2021/02/28/laravel-debug-leaking-secrets/

February 28, 2021

C:\Python27\python.exe



```
LARAVEL V6.9 / ____/  
log session bot found! restore session  
Using Configuration :  
    FILES=ip.txt  
    THREAD=50  
    SESSION=http://184.168.198.14:80  
Want to continue session ? [Y/n] n
```

An attacker logged in through RDP a few days ago to run a “smtp cracker” that scans a list of IP addresses or URLs looking for misconfigured Laravel systems. These attackers are looking for websites that have debug mode enabled, which allows the attacker to see their .env (config) file. The .env file includes AWS, O365, SendGrid, Twilio credentials and more.

What is Laravel?

Laravel is a free, open-source[3] PHP web framework, created by Taylor Otwell and intended for the development of web applications

<https://en.wikipedia.org/wiki/Laravel>

Laravel provides drivers for SMTP, Mailgun, Mandrill, Amazon SES, PHP's mail function, and sendmail, allowing you to quickly get started sending mail through a local or cloud based service of your choice.

<https://laravel.com/docs/5.1/mail>

The debug option is turned off by default on Laravel systems, but it appears many users are enabling debug and not understanding the consequences. Here's an example of debug being enabled (set to true) in the .env config file.

```
<tr>
  <td>
    APP_DEBUG
  </td>
  <td>
    <pre class=sf-dump id=sf-dump-2067749625 data-indent-pad=" ">
      "<span class=sf-dump-str title="4 characters">true</span>
```

config file)

You can check to see if debug is enabled by checking for .env in the web root (site.com/.env) or by sending random data to the webserver and reviewing the response. In the response, you will see the debug option as well as all the information from the .env file, which includes the secrets.

Here's an example of a web response with debug enabled:

```
    AWS_ACCESS_KEY_ID
  </td>
  <td>
    <pre class=sf-dump id=sf-dump-1797855799 data-indent-pad=" ">
      "<span class=sf-dump-str title="20 characters">[REDACTED]</span>
      "
    </pre>
    <script>
      Sfdump("sf-dump-1797855799")
    </script>
  </td>
</tr>
<tr>
  <td>
    AWS_SECRET_ACCESS_KEY
  </td>
  <td>
    <pre class=sf-dump id=sf-dump-706539060 data-indent-pad=" ">
      "<span class=sf-dump-str title="40 characters">[REDACTED]</span>
      "
    </pre>
    <script>
      Sfdump("sf-dump-706539060")
    </script>
  </td>
</tr>
<tr>
  <td>
    AWS_DEFAULT_REGION
  </td>
  <td>
    <pre class=sf-dump id=sf-dump-840748221 data-indent-pad=" ">
      "<span class=sf-dump-str title="9 characters">us-east-1</span>
```

Here's an example of an .env file:

```
APP_NAME=[REDACTED]
APP_ENV=[REDACTED]
APP_KEY=[REDACTED]
APP_DEBUG=true
APP_URL=[REDACTED]

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=[REDACTED]
DB_USERNAME=[REDACTED]
DB_PASSWORD=[REDACTED]

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=ssl://smtp.googlemail.com
MAIL_PORT=465
MAIL_USERNAME=[REDACTED]
MAIL_PASSWORD=[REDACTED]
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=[REDACTED]
MAIL_FROM_NAME="[REDACTED]"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

FILESYSTEM_DRIVER=public

FIREBASE_PUSH_NOTIFICATION_KEY=[REDACTED]

JAVA_BRIDGE_URL=[REDACTED]
JAVA_BRIDGE_DRIVER=Pjb62
JAVA_BRIDGE_RESOURCE_PATH="/tmp/lib"
JAVA_BRIDGE_KEY_STORE_PATH="/tmp/lib"

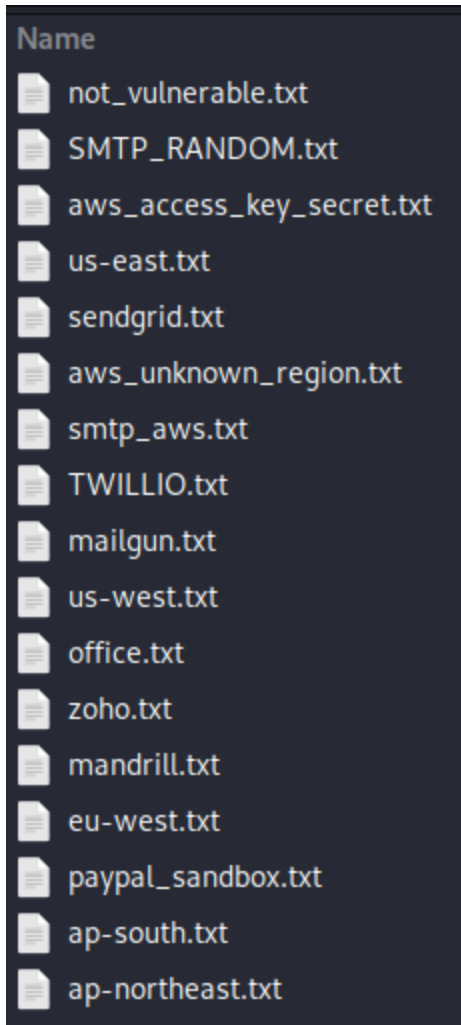
CYBER_SOURCE_ACCESS_KEY=[REDACTED]
CYBER_SOURCE_PROFILE_ID=[REDACTED]
```

The “smtp cracker” script, which by the way—is not a cracker, and grabs more than just smtp credentials; it uses the above methods to crawl a list of IPs/URLs looking for specific strings in the response such as PayPal, AWS_KEY, SES_KEY, Twilio, sendgrid, office365, zoho, mailgun and others.

Here’s part of the script, where it looks for a string in mailhost and outputs the secret(s) to the appropriate file.

```
else:
    # mod aws
    if '.amazonaws.com' in mailhost:
        getcountry = reg('email-smtp.(.*?).amazonaws.com', mailhost)[0]
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
        save = open('Results/'+getcountry[:-2]+'.txt', 'a')
        save.write(remover+'\n\n')
        save.close()
        remover = str(build).replace('\r', '')
        save2 = open('Results/smtp_aws.txt', 'a')
        save2.write(remover+'\n\n')
        save2.close()
    elif 'sendgrid' in mailhost:
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
        save = open('Results/sendgrid.txt', 'a')
        save.write(remover+'\n\n')
        save.close()
    elif 'office365' in mailhost:
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
        save = open('Results/office.txt', 'a')
        save.write(remover+'\n\n')
        save.close()
    elif '1and1' in mailhost or '1und1' in mailhost:
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
        save = open('Results/1and1.txt', 'a')
        save.write(remover+'\n\n')
        save.close()
    elif 'zoho' in mailhost:
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
        save = open('Results/zoho.txt', 'a')
        save.write(remover+'\n\n')
        save.close()
    elif 'mandrillapp' in mailhost:
        build = 'URL: '+str(url)+'\nMETHOD: '+str(method)+'\nMAILHOST: '
        remover = str(build).replace('\r', '')
```

The output of the script is saved in a folder named Results. The results are divided into groups as seen above.



Inside those files contain the secrets and method used to extract the information. This is a partial example of the sendgrid file:

```
1 URL: http:// [REDACTED]:80
2 METHOD: debug
3 MAILHOST: smtp.sendgrid.net
4 MAILPORT: 587
5 MAILUSER: [REDACTED]
6 MAILPASS: [REDACTED]
7 MAILFROM: [REDACTED]
8 FROMNAME: [REDACTED]
9
10 URL: http:// [REDACTED]:80
11 METHOD: debug
12 MAILHOST: smtp.sendgrid.net
13 MAILPORT: 465
14 MAILUSER: apikey
15 MAILPASS: [REDACTED]
16 MAILFROM: [REDACTED]
17 FROMNAME: [REDACTED]
18
19 URL: http:// [REDACTED]:80
20 METHOD: /.env
21 MAILHOST: smtp.sendgrid.net
22 MAILPORT: 587
23 MAILUSER: apikey
24 MAILPASS: [REDACTED]
25 MAILFROM: [REDACTED]
26 FROMNAME: [REDACTED]
27
28 URL: http:// [REDACTED]:80
29 METHOD: debug
30 MAILHOST: smtp.sendgrid.net
31 MAILPORT: 587
32 MAILUSER: apikey
33 MAILPASS: [REDACTED]
34 MAILFROM: [REDACTED]
```

This is a partial example of the office365 text file:

```
1 URL: http:// [REDACTED] :80
2 METHOD: /.env
3 MAILHOST: smtp.office365.com
4 MAILPORT: 587
5 MAILUSER: [REDACTED]
6 MAILPASS: [REDACTED]
7 MAILFROM:
8 FROMNAME:
9
10 URL: http:// [REDACTED] :80
11 METHOD: debug
12 MAILHOST: smtp.office365.com
13 MAILPORT: 587
14 MAILUSER: [REDACTED]
15 MAILPASS: [REDACTED]
16 MAILFROM: [REDACTED]
17 FROMNAME: [REDACTED]
18
19 URL: http:// [REDACTED] 1:80
20 METHOD: debug
21 MAILHOST: smtp.office365.com
22 MAILPORT: 587
23 MAILUSER: [REDACTED]
24 MAILPASS: [REDACTED]
25 MAILFROM:
```

We are in the process of contacting over 100 people/organizations who's systems are leaking secrets via Laravel debug in hopes that they will remediate the issue and change their passwords. If anyone needs help fixing this issue, please use the [Contact Us](#) form to get in touch.

MITRE ATT&CK

Initial Access

RDP login as local admin from 64.86.198[.]22. No brute force attempts were seen from this IP.

Execution

The attacker installed and used Python 2.7 to execute the smtp cracker script.

Web requests to root/.env or post data that includes 0x[]:androxgh0st (hard coded string in smtp.py)

IOCs

If you would like access to our internal MISP and/or threat feeds please see [here](#).

File

filename	smtp.py
md5	39e1ec2c704bcdb57034da4ac288446e
sha1	352e9c78f08574adbaa0aaf49c19d5853bb4be36
sha256	478290f801dafc9086810f857c123d74690920a4c44fb573cd01463c1b6fb432

Network

type	value	comment
ip-src	64.86.198.22	RDP Unauthorized Access

Valid Accounts - T1078
Python - T1059.006
Scripting - T1064
User Execution - T1204
Graphical User Interface - T1061
Command and Scripting Interpreter - T1059

[Link to IOCs and artifacts](#)