Network Security Project2 Report

0416324 An-Fong Hwu 胡安鳳

(Note: this report is written in MD-Like format)

(懶人包: Lazy people's pack)Demonstration of the whole cracking process:

https://youtu.be/UhorgrUoGYo

The key used for encrypting the sensitive photo is:
and the final photo is

How do I implement the hacking of this project

* Firstly we have only the website and nothing
* Then we can use the wget -r http://140.113.194.66:20084/blog/ to crawl the content in the website.(r stands for recursively download the folders and subfolders)

```
alfons@alfons ~/Desktop/Programming/Network_Security_Spring_2018/Lab2  master  wget -r http://140.113.194.66:20084/blog/
--2018-04-21 09:00:36--  http://140.113.194.66:20084/blog/
Connecting to 140.113.194.66:20084... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '140.113.194.66:20084/blog/index.html'

140.113.194.66:2008     [ <=>                ]   7.06K  --.-KB/s    in 0s

2018-04-21 09:00:36 (446 MB/s) - '140.113.194.66:20084/blog/index.html' saved [7226]

Loading robots.txt; please ignore errors.
--2018-04-21 09:00:36--  http://140.113.194.66:20084/robots.txt
Reusing existing connection to 140.113.194.66:20084.
HTTP request sent, awaiting response... 200 OK
Length: 109 [text/plain]
Saving to: '140.113.194.66:20084/robots.txt'

140.113.194.66:2008 100%[===================>]     109  --.-KB/s    in 0s

2018-04-21 09:00:36 (16.3 MB/s) - '140.113.194.66:20084/robots.txt' saved [109/109]

--2018-04-21 09:00:36--  http://140.113.194.66:20084/blog/views/element/element.css
Reusing existing connection to 140.113.194.66:20084.
HTTP request sent, awaiting response... 200 OK
Length: 124311 (121K) [text/css]
Saving to: '140.113.194.66:20084/blog/views/element/element.css'

140.113.194.66:2008 100%[===================>] 121.40K  --.-KB/s    in 0.01s

2018-04-21 09:00:36 (9.15 MB/s) - '140.113.194.66:20084/blog/views/element/element.css' saved [124311/124311]

--2018-04-21 09:00:36--  http://140.113.194.66:20084/blog/views/element/vue.min.js
Reusing existing connection to 140.113.194.66:20084.
HTTP request sent, awaiting response... 200 OK
Length: 72696 (71K) [application/javascript]
Saving to: '140.113.194.66:20084/blog/views/element/vue.min.js'

140.113.194.66:2008 100%[===================>]  70.99K  --.-KB/s    in 0.001s

2018-04-21 09:00:36 (131 MB/s) - '140.113.194.66:20084/blog/views/element/vue.min.js' saved [72696/72696]
```

* The robots.txt tells us what content should not be scanned or displayed in the search engine, thus I think it will show some clue about the web vulnerabilities.
Reference to: https://blog.keniver.com/2017/03/robots-txt-%E7%9A%84%E4%BD%BF%E7%94%A8%E6%96%B9%E6%B3%95%E8%88%87%E5%AE%89%E5%85%A8%E6%B3%A8%E6%84%8F%E4%BA%8B%E9%A0%85/
Well goes the saying, “Never try to prove what nobody doubts”,and the robots.txt really does such thing.

```
User-agent: *
Disallow: /phpMyAdmin_NS_pRojEct_2017/
Disallow: /backup.tar.gz
Disallow: /blog/memorandum.txt
```

* I tried to download the memorandum.txt but in vain.
TA also tells us that the swap ot temp file may leak some important information.
Then I wget the .memorandum.txt.swp and tried some other temporary files or backup files again and successfully found an encrypted data in .memorandum.txt.swp

```
✘ alfons@alfons ~/Desktop/Programming/Network Security Spring 2018/Lab2  master  wget -r http://140.113.194.66:20084/blog/memorandum.txt
--2018-04-21 09:32:23--  http://140.113.194.66:20084/blog/memorandum.txt
Connecting to 140.113.194.66:20084... connected.
HTTP request sent, awaiting response... 404 Not Found
2018-04-21 09:32:23 ERROR 404: Not Found.

✘ alfons@alfons ~/Desktop/Programming/Network Security Spring 2018/Lab2  master  wget http://140.113.194.66:20084/blog/memorandum.txt
--2018-04-21 09:32:31--  http://140.113.194.66:20084/blog/memorandum.txt
Connecting to 140.113.194.66:20084... connected.
HTTP request sent, awaiting response... 404 Not Found
2018-04-21 09:32:31 ERROR 404: Not Found.

✘ alfons@alfons ~/Desktop/Programming/Network Security Spring 2018/Lab2  master  wget -r http://140.113.194.66:20084/phpMyAdmin_NS_pRojEct_
2017/
--2018-04-21 09:35:53--  http://140.113.194.66:20084/phpMyAdmin_NS_pRojEct_2017/
Connecting to 140.113.194.66:20084... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: '140.113.194.66:20084/phpMyAdmin_NS_pRojEct_2017/index.html'

140.113.194.66:20084/phpMyAdmin_NS_p     [ <=>                                                                ]  10.18K  --.-KB/s    in 0.001s

2018-04-21 09:35:53 (11.8 MB/s) - '140.113.194.66:20084/phpMyAdmin_NS_pRojEct_2017/index.html' saved [10429]

FINISHED --2018-04-21 09:35:53--
Total wall clock time: 0.3s
Downloaded: 1 files, 10K in 0.001s (11.8 MB/s)
 alfons@alfons ~/Desktop/Programming/Network Security Spring 2018/Lab2  master  wget -r http://140.113.194.66:20084/backup.tar.gz
--2018-04-21 09:36:06--  http://140.113.194.66:20084/backup.tar.gz
Connecting to 140.113.194.66:20084... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147004 (144K) [application/octet-stream]
Saving to: '140.113.194.66:20084/backup.tar.gz'

140.113.194.66:20084/backup.tar.gz   100%[=================================================================>] 143.56K  --.-KB/s    in 0.04s

2018-04-21 09:36:07 (3.70 MB/s) - '140.113.194.66:20084/backup.tar.gz' saved [147004/147004]

FINISHED --2018-04-21 09:36:07--
Total wall clock time: 0.1s
Downloaded: 1 files, 144K in 0.04s (3.70 MB/s)
 alfons@alfons ~/Desktop/Programming/Network Security Spring 2018/Lab2  master
```

* The encrypted sensitive data is the following (encoded in base64 format like the last project)

* I write a python code to process it (embedded the linux script in it)

```
"""
import base64
import binascii
import os
import sys
os.system('cat '+sys.argv[1]+' | base64 --decode > decoded_memo.bin') #execute linux command in python

with open('decoded_memo.bin', 'r') as fptr:
    # print fptr.read()
    all_data = str(fptr.read())
    # print(all_data)
    print(type(all_data))

    xor_key = sys.argv[2]

    key_len = len(xor_key)
    total_len = len(all_data)
    decrypted_data = "" #the xor-decrypted data will be here

    print('key', xor_key, 'Keylen', key_len, 'total data len', total_len)
    for i in range(0, total_len, key_len):
        for j in range (0, key_len, 1):
            #print('i ', i, ' and j ', j , '\n')
            decrypted_data += chr(ord(all_data[i + j]) ^ ord(xor_key[j])) #ord for char->int and chr vice versa

    with open('xorkey_decrypted_data.txt', 'w') as fptr2:
        fptr2.write(decrypted_data)
```

* The files is using XOR encryption, so I put the decoded_demo.bin to the website for cracking.
This website: https://wiremask.eu/tools/xor-cracker/ is used for XOR cracking.
Then there are 2 possible keys

Wiremask Articles Writeups Tools

		Start
3	11.4%	Start
6	**19.0%**	Start
9	8.9%	Start
12	14.2%	Start
15	6.9%	Start
18	10.6%	Start
24	8.2%	Start
30	6.5%	Start
36	5.3%	Start

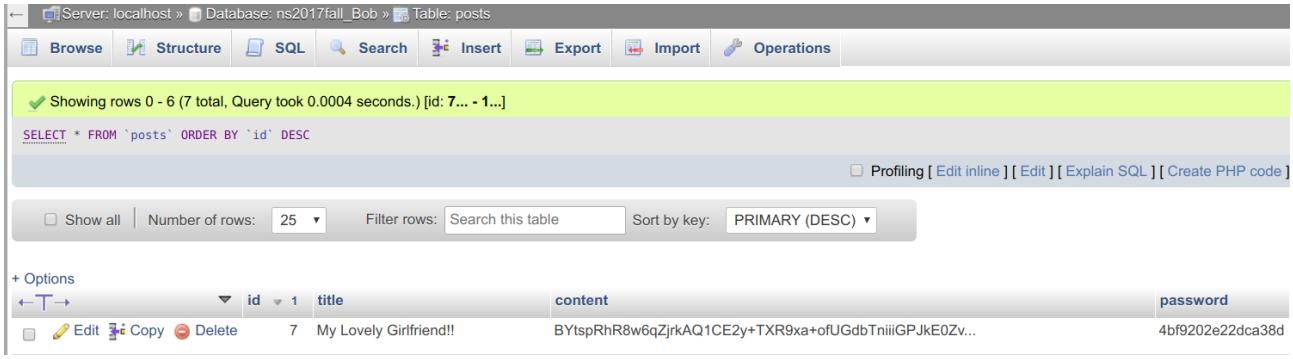
Possible keys

Keys		Decrypted File
rafted	72 61 66 74 65 64	Download
RAFTED	52 41 46 54 45 44	Download

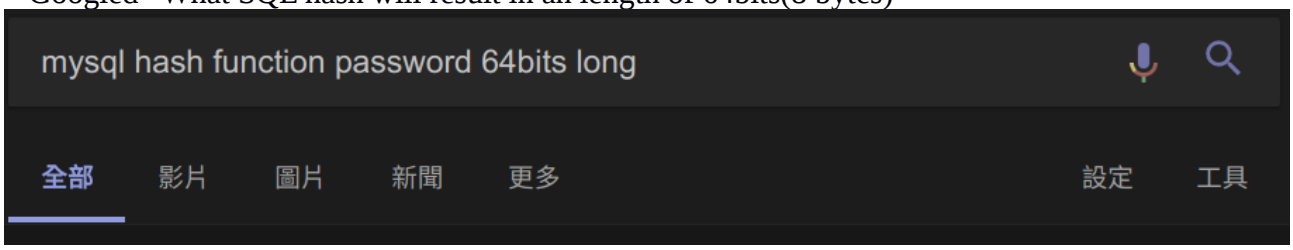
I tried rafted as key in my python script and successfully decrypted the swap file of memorandum.txt

```
 alfons@alfons  ~/Desktop/Programming/Network Security Spring 2018/Lab2  master ●  python xor_tool.py .memorandum.txt.swp rafted
<type 'str'>
('key', 'rafted', 'Keylen', 6, 'total data len', 10818)
 alfons@alfons  ~/Desktop/Programming/Network Security Spring 2018/Lab2  master ●  cat xorkey_decrypted_data.txt
2016.01.13
phpMyAdmin Account & Password
Account: BobIsGod
Password: mazesamphitheatreclobbers
```

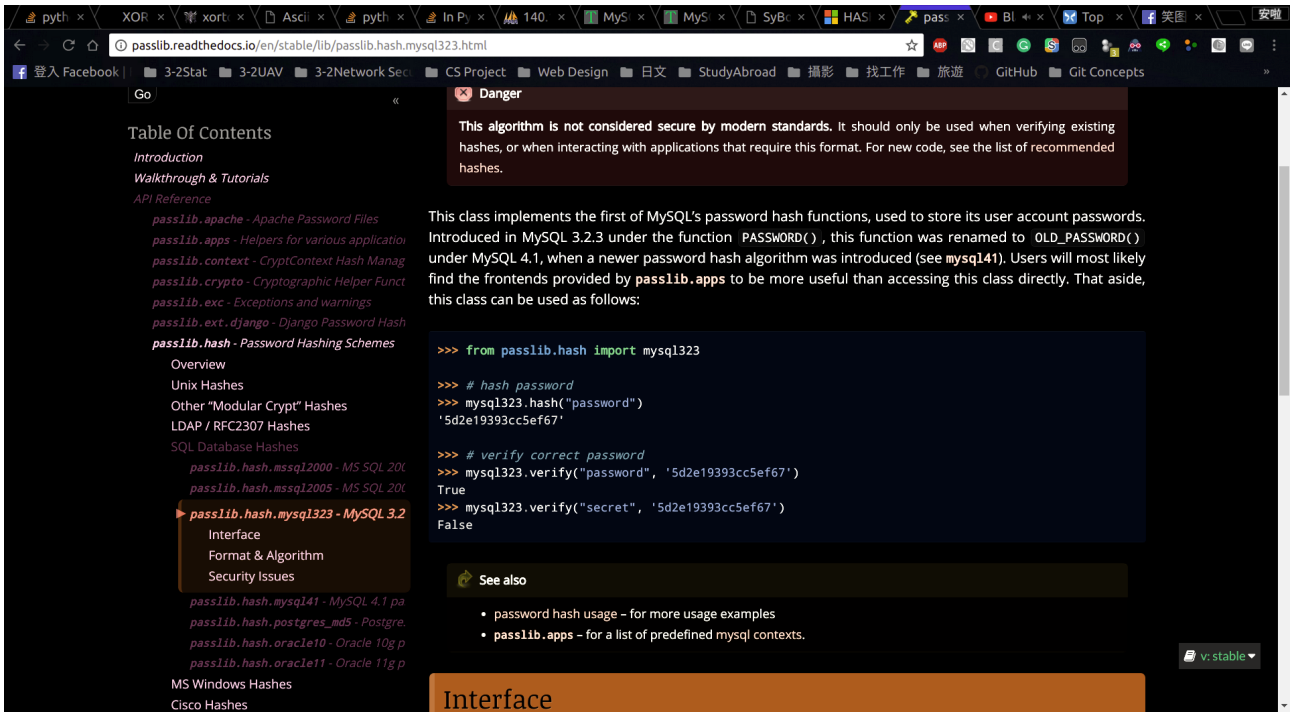
* After the account name and password of the database is revealed, use them to login to the phpmyadmin in out project and we can see the password after hashing.



* Googled “What SQL hash will result in an length of 64bits(8 bytes)”



And I found this information



It is 64bits long, hashed with mysql323 hashing.

* Then search for sql323 hashing cracker and found this website: https://www.tobtu.com/mysql323.php which provides us with the collidor of mysql323 hashing

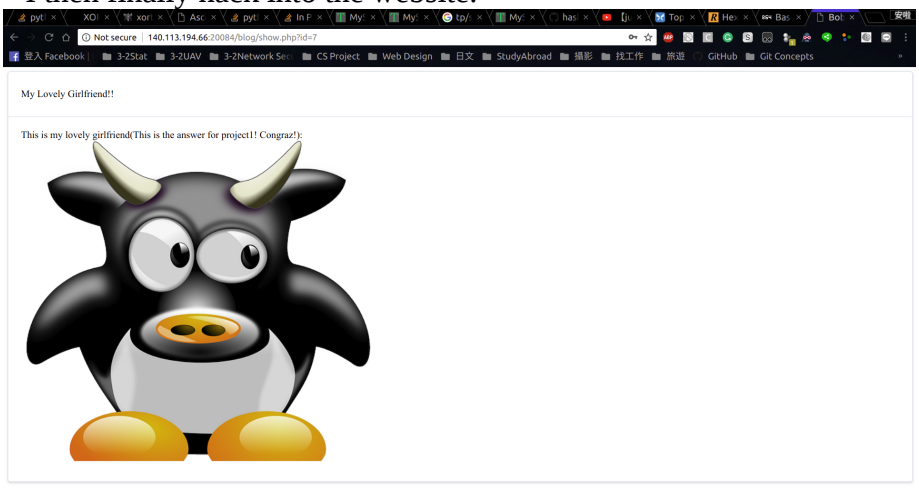
```
Thread creation failed: 11
Thread creation failed: 11
Thread creation failed: 11
^C
 alfons@alfons  ~/MySQL323 Collider  ./mysql323collider64 --hash 4bf9202e22dca38d --memory 500 --threads 1
Initializing...
Took 3.35 sec
784.3 Tp/s [100.0%] ^C
Crack time:    64.460 seconds
Average speed: 760.1 Tp/s
Resume code:   31811901
 alfons@alfons  ~/MySQL323 Collider  ./mysql323collider64 --hash 4bf9202e22dca38d --memory 500 --threads 4
Initializing...
Took 3.39 sec
2.829 Pp/s [24.7% 25.2% 25.1% 25.0%] ^C
Crack time:    17.033 seconds
Average speed: 2.725 Pp/s
Resume code:   30140183
 alfons@alfons  ~/MySQL323 Collider  ./mysql323collider64 --hash 4bf9202e22dca38d --memory 500 --threads 6
Initializing...
Took 3.35 sec
3.015 Pp/s [22.5% 13.8% 13.7% 13.7% 13.8% 22.5%] ^C
Crack time:    15.559 seconds
Average speed: 3.012 Pp/s
Resume code:   30432347
 alfons@alfons  ~/MySQL323 Collider  ./mysql323collider64 --hash 4bf9202e22dca38d --memory 500 --threads 8
Initializing...
Took 3.37 sec
3.226 Pp/s [12.2% 12.5% 12.5% 12.7% 12.4% 12.6% 12.6% 12.4%]
4bf9202e22dca38d:2162657b4c2f4a56226638762e2e:!be{L/JV"f8v..

Crack time:    190.386 seconds
Average speed: 3.251 Pp/s
 alfons@alfons  ~/MySQL323 Collider  ./mysql323collider64 --hash 4bf9202e22dca38d --memory 2000 --threads 8
Initializing...
Took 14.59 sec
9.189 Pp/s [13.1% 11.9% 12.8% 12.6% 12.8% 11.9% 12.4% 12.6%]
4bf9202e22dca38d:23556345724c5c4422583e3c4846:#UcErL\D"X><HF

Crack time:    223.132 seconds
Average speed: 10.16 Pp/s
 alfons@alfons  ~/MySQL323 Collider
```

Usage(in linux)

```shell
chmod + x mysql323collider64
./mysql323collider64 –hash <hashed value we want to collide> --memory <memory limit for rainbow table> --thread <threads for parallel processing>
```

The result will be
[hashed value we want to collide]:[cracked key in hex]:[cracked key converted from hex to ascii]
and there are several keys with the same hashed values which provide us an opportunities to log in for picture due to hash collision.

* I then finally hack into the website.



What I have learned in this project
* Some useful scripts in Linux such as cat base64file.txt | base64 --decode > output.txt then it easily decodes the encoded data and output and the wget command to crawl down the website.
* XOR manipulating with self-written python script for arbitrary key length and corresponding encrypted data.
* The ability to google tons of problems.
* Common web vulnerabilities.
* What does robots.txt do in website.

How to prevent or patch these vulnerabilities?
* Never leave the swap file, tmp file and backup files in the website, store in other more secured way, otherwise it may leak some sensitive information.
* Fix the robots.txt with the following way https://blog.keniver.com/2017/03/robots-txt-%E7%9A%84%E4%BD%BF%E7%94%A8%E6%96%B9%E6%B3%95%E8%88%87%E5%AE%89%E5%85%A8%E6%B3%A8%E6%84%8F%E4%BA%8B%E9%A0%85/
robots.txt 應該視為引導爬蟲索引網站的工具, 而不是保護網站的工具, 任何不希望出現在網路上的資料不應該上傳到網路上, 假若逼不得已必須放到網路上, 必須使用 Http Authorization 之類的存取保護機制保護資料.
* Hash the sensitive data in database with stronger hashing algorithm such as SHA-512, therefore even the database is compromised, the intruder is still not able to crack the confidential data.