Final Report for the Hacking Challenge Creation

Predictable Session IDs & Steganography

Alexander Daniel Nikolaos Lelidis, André Baptista Águas,
Ard Kastrati, Khalid Aldughayem, Uroš Tešić

January 17, 2018

Abstract

This challenge combines two different tasks from Web Security and Steganography. It addresses the problem of the predictable sessions and the security through obscurity. The idea of this challenge is first to bypass the authorization in the HTTP level by hijacking a session with a predictable session ID. Secondly, the attacker must extract the secret information which is hidden by using steganography instead of cryptography.

1 Requirements

- A program to reverse the Mersenne Twister MT-19937. A program that solves this can be found in `https://github.com/eboda/mersenne-twister-recover`.
- A program needed for requesting the 624 session IDs and storing them in a file so that the Mersenne Twister can recover the pseudorandom generator. A simple program that does this is given in the following:

```
import requests

with open("mersenneCookies.txt","w") as output:

  for i in range(625):

    r = requests.post('http://<ip address>:5000/login', data =
                     {'user':'user1', 'password':'password1'})
    output.write(str(r.cookies['netsecLoginCookie']) + "\n")
    requests.post('http://<ip address>:5000/logout',cookies=r.cookies)
```

  where `<ip address>` must be set to the IP address of server.
- A cookie editor in your web browser. We recommend you to use developer tools in Firefox (*Tools* → *Web Developer* → *Toggle Tools*).
- Basic knowledge about steganography.

2 Learning goals

There are three different learning goals in this challenge:

1. *If the attacker can control the pseudorandom generator, then he can control everything.* As discussed in the lecture, controlling the pseudorandom generators is the perfect attack, since it is almost impossible to detect. Generating not truly random numbers are very hard to detect.

2. *The security of an algorithm should rely solely on the secrecy of the key* [1]
   Never use algorithm whose security relies solely on the fact that the algorithm is secret. One speaks of security by obscurity. The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. However, *once* it is known that something secret is hidden there, steganography is insecure from todays point of view. With security through obscurity it is not possible to establish standard algorithms, which can be tested from others. As a countermeasure, Kerckhoffs's principles have been established. In 1833, Auguste Kerkhoffs stated six design principles for military ciphers in [2] and one of them is [1]:

   - The cipher method should not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

   In the scenario that is given in this project, *Alice* misapplied the purpose of steganography. *Alice's* goal is to keep her data confidential, in case someone manages to log into her account. However, in case someone attacks *Alice*, then *Alice is already a suspect* and hence the attacker can assume that secret data can be hidden by using steganography. Once this happens, the purpose of using steganography is pointless. It is worth mentioning that the Kerkhoff's principle has been often ignored, which led to fatal results.

3. *Adding more different unsecure mechanisms in the system doesn't make it more secure, since the attacker can simply bypass all of them one after another.*

3 Mission

Alice and Bob need to share secrets with each other. However, they don't trust the security of the chat in the social network they are using. One day Alice reads about the *steganography* and she was amazed. As a beginner in modern cryptography, she thought that this has to be the best way to share secrets. Without further thinking she arranges a meeting with Bob and they decide to hide their data in different pictures that they send to each other. On the one hand, the website is known to love the Mersenne Twister[1] whenever a random generator is needed and on the other hand Alice must log in every 5 minutes to this website. Alice was right about one thing: *The website is indeed not secure and there is a way to bypass the login page.* Ironically, the *steganography* is not much more secure than that. The secret organization "..." suspects that *Alice* is preparing to commit a crime. As a secret agent, an account has been already opened for you in the social network. The credentials are the following:

- username: `user1`
- password: `password1`

[1]The Mersenne Twister (MT 19937) is used as PRNG in many programming languages. However, given at least 624 outputs of a Mersenne Twister we can restore its internal state and find all other consecutive numbers.

Your mission, should you choose to accept it, is to bypass the login page for *Alice*, read the data exchanged between them and extract the secret message that they exchanged with each other.

4 Mitigation

1. Use cryptographically secure pseudorandom number generator (CSPRNG).
2. Use standard encryption schemes from modern cryptography (instead of steganography) for confidentiality (such as AES, RSA, El-Gammal etc).

5 Type of Challenge

Online

6 Category

Web Security & Cryptography

7 Hints

- Mersenne Twister is used to generate the session keys!
- Consider the channels (red, green and blue) of the image *separately* and look if you can find information in some of the bits of every channel.

8 Setup

- Start the client and the server in VirtualBox[2] or VMWare. The credentials are:
  - Client

    ```
    password: 123
    ```
  - Server

    ```
    username: user
    password: ERenaRboRy
    ```
- Find the IP address of the server by running `ifconfig` in the server inside the server machine.
- Open the browser and type the `<ip address of the server>:5000`

[2]Make sure that the machine is connected with Host-only adapter. See README file if any problems occur.

9 Solution

9.1 Idea of the solution

After logging in, we see a harmless profile with chat and news. However, if we look in cookies, we will see a `netsecLoginCookie`. If we log out and log in again, we see that the value changed. Multiple logins tell us that the cookie is always less than 32-bits, and its value changes randomly. That randomness is the key. Website is written in python and flask framework. Python uses Mersenne twister as PRNG. MT has a period length of 2^{19937}. However, this doesn't guarantee its safety. There are two possible attacks on this PRNG:

- We can download Untwister program (`https://github.com/altf4/untwister`) presented at BSides in Las Vegas in 2014. We only need a few generated values to obtain the seed. Unfortunately, this is done by brute force. It is possible to do it this way because seed has 32-bits, but it still takes 3-4 hours in the general case. We decided to use a small seed value so this attack takes only a few seconds.

- If we take a look at Mersenne twister implementation, we can see that it is just a big XOR machine with 624 numbers as its state. Every number is transformed (tempered), and output as a random values, until all 624 numbers are used. The state is then transformed together, and the cycle continues. Tempering can be inverted, and we can easily reconstruct the entire state of the PRNG, and guess every subsequent number. This requires 625 consequent integers (state plus one more number in case of refresh). For this attack to work no one but us is allowed to log in when we poll the server. In case that the admin logs in, we must start again.

After we obtain the next state, we should replace the cookie in our browser with the next generated value, and wait for *Alice* to log in again. Once we gain access to the *Alice's* account, we can see that he is up to no good. By inspecting the chat, we obtain an image that shows the next target. The image looks like a city, but isn't identifiable. This is likely a steganography challenge. The data is hidden in LSB of every channel (red, green and blue) separately.

9.2 Step-by-step instructions

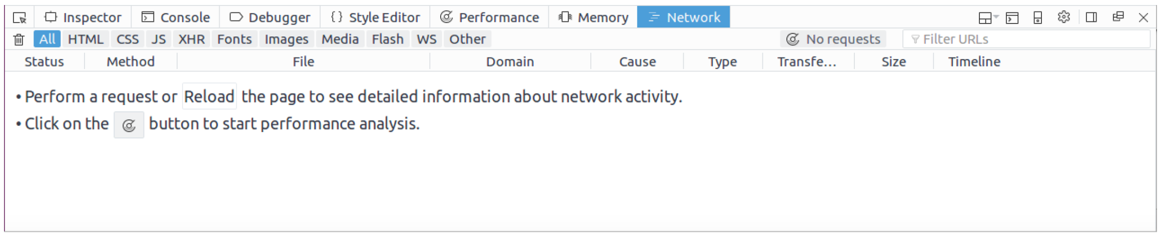
- Generate 625 session IDs by using python3 with the given program in the requirement section.

- Get the state of the generator of the Mersenne Twister and predict some of the next session IDs, by using the Mersenne twister recoverer from `https://github.com/eboda/mersenne-twister-recover`. We recommend you to generate next 50 session IDs in advance. The programs that generates first 625 session IDs and the next 50 ones can be found in the client machine under the `Documents/attack` folder.

- Every 5 minutes *Alice* will log in to the website, so you may have to wait for *Alice* to log in for the first time. After 5 minutes, you can either try to change the cookie to the next one, or you can log out and log in again to see what is the current session ID in order to avoid any mistakes. After you log in after 5 minutes, you will know that the previous session ID is the session ID of the *Alice*.

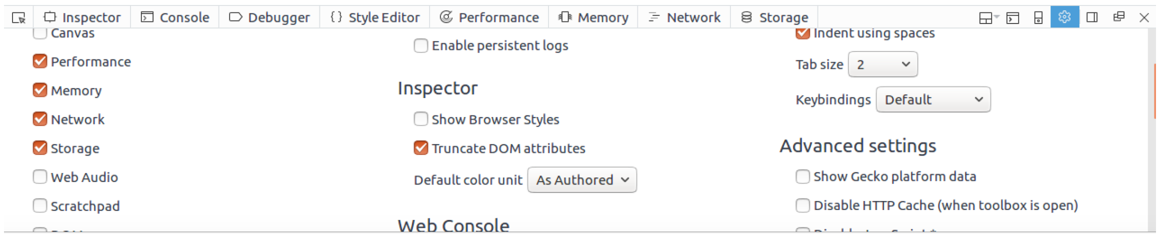
- Change your session ID to *admin's* session ID (Session hijacking). In order to edit the session ID, you must:

– Open developer tools (*Tools* → *Web Developer* → *Toggle Tools*)


Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
All HTML CSS JS XHR Fonts Images Media Flash WS Other
No requests
Filter URLs
Status Method File Domain Cause Type Transfe… Size Timeline

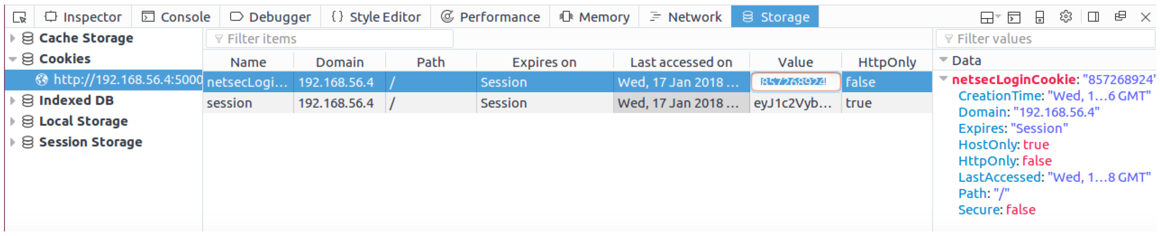
• Perform a request or Reload the page to see detailed information about network activity.
• Click on the button to start performance analysis.

– Enable the storage inspector via the settings of the developer tools



☐ Canvas	☐ Enable persistent logs	✓ Indent using spaces
✓ Performance	**Inspector**	Tab size 2
✓ Memory	☐ Show Browser Styles	Keybindings Default
✓ Network	✓ Truncate DOM attributes	**Advanced settings**
✓ Storage	Default color unit As Authored	☐ Show Gecko platform data
☐ Web Audio	**Web Console**	☐ Disable HTTP Cache (when toolbox is open)
☐ Scratchpad		

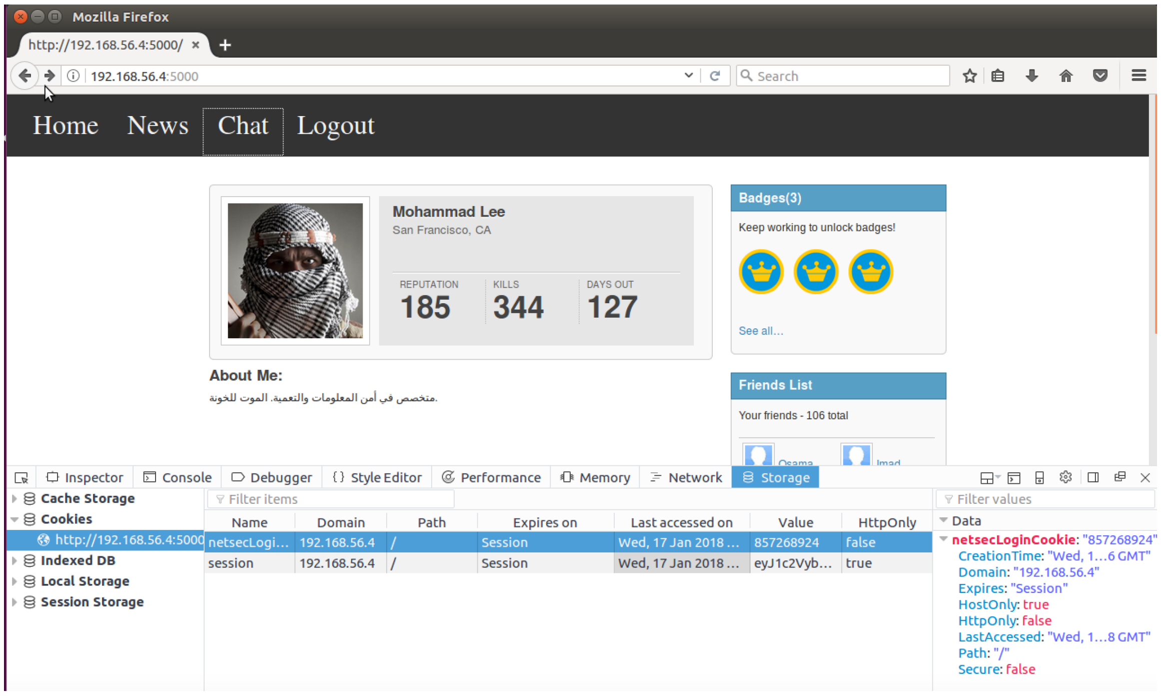
– Change the `netsecLoginCookie` to the *admin's* session ID.



Name	Domain	Path	Expires on	Last accessed on	Value	HttpOnly
netsecLogi…	192.168.56.4	/	Session	Wed, 17 Jan 2018 …	857268924	false
session	192.168.56.4	/	Session	Wed, 17 Jan 2018 …	eyJ1c2Vyb…	true

Data
netsecLoginCookie: "857268924"
CreationTime: "Wed, 1…6 GMT"
Domain: "192.168.56.4"
Expires: "Session"
HostOnly: true
HttpOnly: false
LastAccessed: "Wed, 1…8 GMT"
Path: "/"
Secure: false

– Open `http://<ip address>:5000`.



Mohammad Lee
San Francisco, CA

REPUTATION	KILLS	DAYS OUT
185	344	127

Keep working to unlock badges!

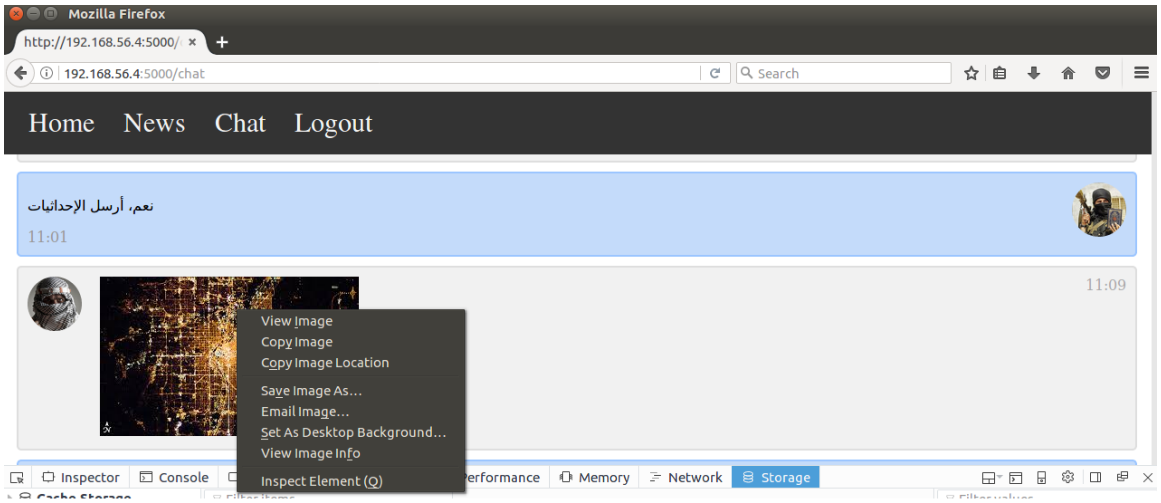
About Me:
متخصص في أمن المعلومات والتعمية. الموت للخونة.

Your friends - 106 total

Name	Domain	Path	Expires on	Last accessed on	Value	HttpOnly
netsecLogi…	192.168.56.4	/	Session	Wed, 17 Jan 2018 …	857268924	false
session	192.168.56.4	/	Session	Wed, 17 Jan 2018 …	eyJ1c2Vyb…	true

Data
netsecLoginCookie: "857268924"
CreationTime: "Wed, 1…6 GMT"
Domain: "192.168.56.4"
Expires: "Session"
HostOnly: true
HttpOnly: false
LastAccessed: "Wed, 1…8 GMT"
Path: "/"
Secure: false

• Go to the chat and download the image that was sent to you.



- Extract the secret data by seperating all three different channels (red, green, blue) from the image and get the LSB in every channel. You can do this by using the following program:

```
import png
import sys

filename = sys.argv[1]

imageReader = png.Reader(filename)

image = imageReader.asRGBA()

width = image[0]
height = image[1]
pixels = list(image[2])

binaryRed = ""
binaryGreen = ""
binaryBlue = ""


counter = 0

for i in range(height):

    row = pixels[i]

    for j in range(width):
        binaryRed += str(row[4*j]%2)
        binaryGreen += str(row[4*j+1]%2)
        binaryBlue += str(row[4*j+2]%2)

        counter += 1
```

```
binaryRed = [binaryRed[i:i+8] for i in range(0, len(binaryRed), 8)]
binaryGreen = [binaryGreen[i:i+8] for i in range(0, len(binaryGreen), 8)]
binaryBlue = [binaryBlue[i:i+8] for i in range(0, len(binaryBlue), 8)]

redText = "".join([chr(int(x,2)) for x in binaryRed])
greenText = "".join([chr(int(x,2)) for x in binaryGreen])
blueText = "".join([chr(int(x,2)) for x in binaryBlue])

print(redText)
print(greenText)
print(blueText)
```

- If we take least significant bit of every color in the image, the start of the red channel will form a string $38°53'52''$ N $77°2'11''$ W. These are the coordinates of the White House. Other two channels have easter eggs in them. Hurry and protect the president!

10 Implementation

The implementation of the system can be divided into two parts:

- *Creating a website with a chat in it and with predictable session IDs.*
  The website is very simple. Since a feature that provides a chat isn't really necessary, we created just a static chat, where the users supposedly can communicate. This means the chat can only be read what is written so far. Next implementation step, was to generate session IDs by using the Mersenne Twister. This was also very simple since the twister MT-19937 is already provided in python.

- *Generating an image where some information is hidden by using steganography.*
  We created a script, which hides the data in a picture. The created picture was then added inside the static chat afterwards.

References

[1] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).* Chapman & Hall/CRC, 2007.

[2] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–83, January 1883.