Man in the Middle Attacks

Objectives

- Examine web parameter tampering attacks in the context of URL parameter tampering and cookie tampering.

- Explore how web developer tools can be used to manipulate web requests to achieve malicious outcomes.

References

- https://owasp.org/www-community/attacks/Web_Parameter_Tampering

Web Parameter Tampering Attacks

The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

Text copied from: https://owasp.org/www-community/attacks/Web_Parameter_Tampering

URL Reference Address

The link on the page directs us to a URL address

https://spacecadets-webvulns.chals.io/params?user=guest

https	The protocol
spacecadets-webvulns.chals.io	The hostname of the server
params	The page we are requesting on the server
?user=guest	The query string indicating we are setting a parameter user to the value ”guest”

Processing Parameters

On the server there is code running to check to see if the user is an administrator, by checking the parameter request. Only the administrator gets access to the flag.

```
@app.route('/params')
def params():
    user=request.args.get('user')
    if user!='admin':
        message=f"You are logged in as {user}, log in as admin to get the flag."
        return render_template('params.html',message=message)
    else:
        message=f"Congrats! Here is your flag: {params_flag}"
        return render_template('params.html',message=message)
```

Parameter Tampering

What if we modified (or tampered) the original request:

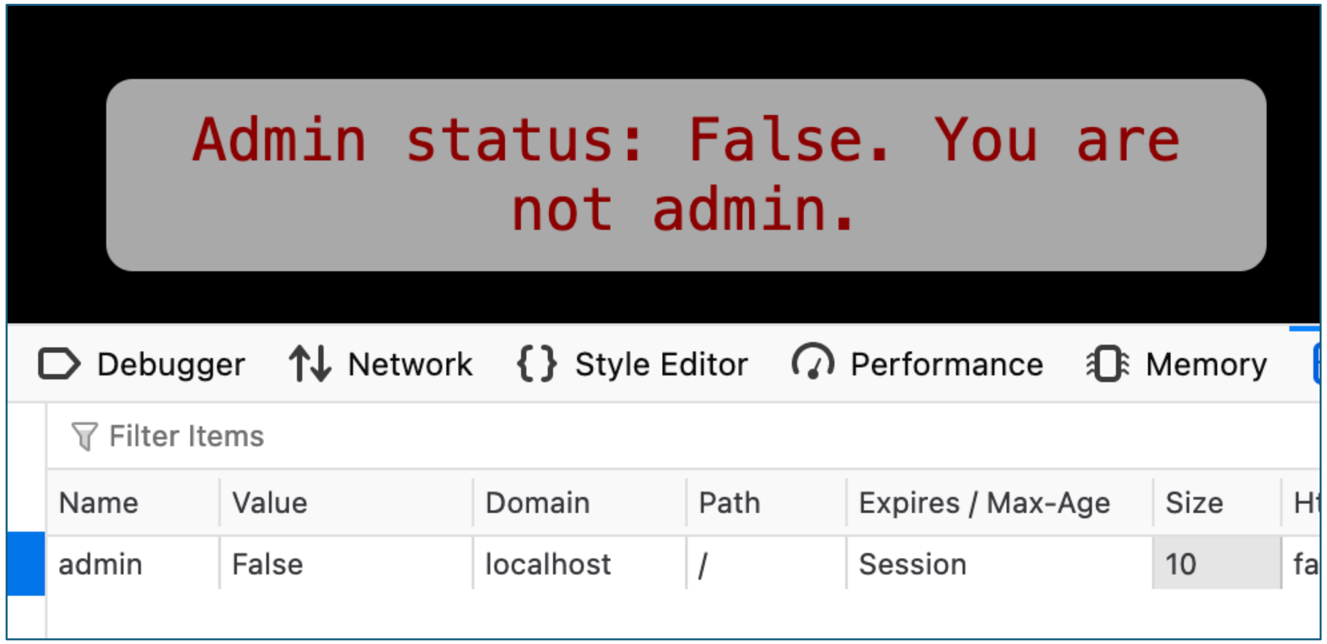
http:// spacecadets-webvulns.chals.io/params?user=guest

To change the user to admin

http:// spacecadets-webvulns.chals.io/params?user=admin

Cookies

- Browse to https://spacecadets-webvulns.chals.io/cookies
- The web application tells us that the administrator status is False.
- The application is determining this by reading a user-supplied cookie value.


Admin status: False. You are not admin.

Name	Value	Domain	Path	Expires / Max-Age	Size	Ht
admin	False	localhost	/	Session	10	fa

Cookie Tampering Attack

- Session cookies are temporary cookies used to store information on the client.

- The are used to track or personalize our web experience. For example, a cookie might indicate that the preferred language is en-US (US English)

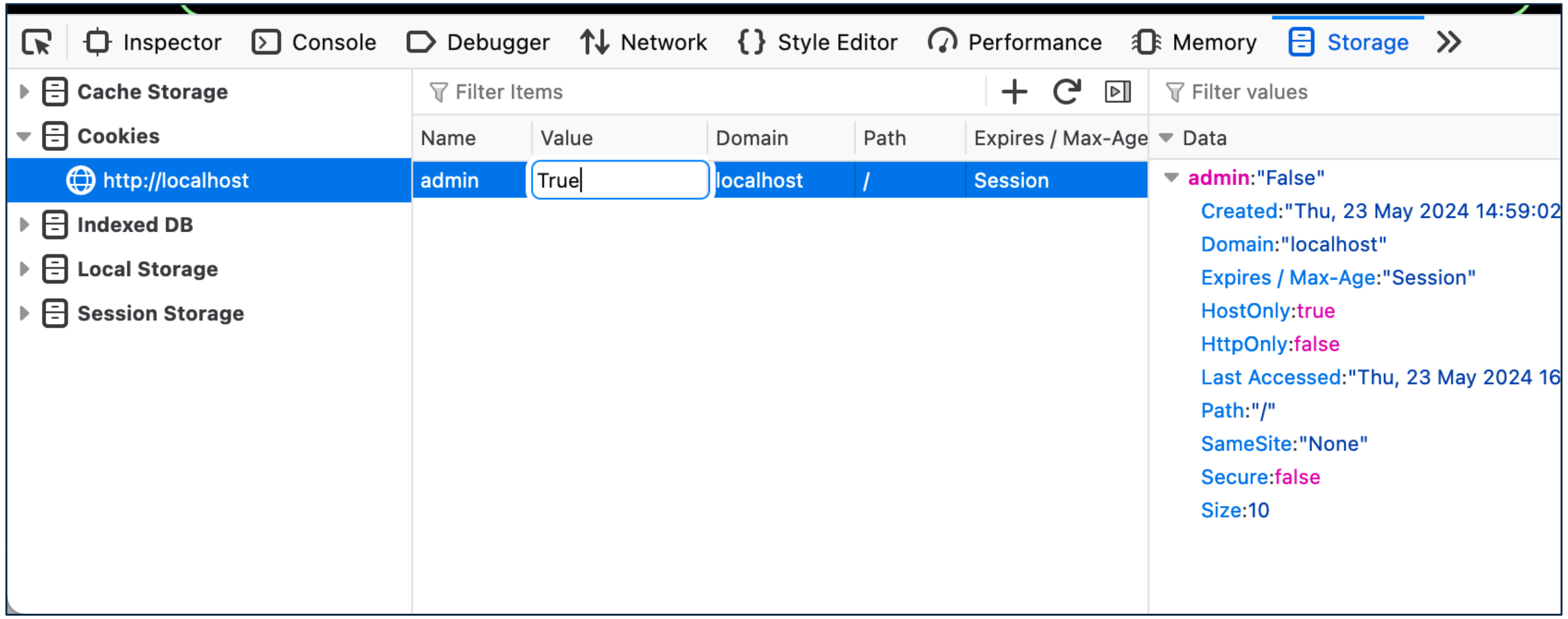
Cookie Tampering Attack

- However, this application has made a mistake and used a user-controlled cookie to determine if the user is an administrator

```
@app.route('/cookies')
def cookies():
    admin = request.cookies.get('admin', 'False')
    if admin != 'True':
       message = f"Admin status: {admin}. You are not admin."
       response = make_response(render_template('cookies.html', message=message))
       response.set_cookie('admin', str('False’))
    else:
        message = f"Congrats! Here is your flag: {cookies_flag}."
        response = make_response(render_template('cookies.html',message=message))
    return response
```

Cookie Tampering Attack

- We can modify the value of a cookie using the browser's Web Developer Tools (typically accessible via Function-F12)



Name	Value	Domain	Path	Expires / Max-Age
admin	True	localhost	/	Session

```
Data
admin:"False"
  Created:"Thu, 23 May 2024 14:59:02
  Domain:"localhost"
  Expires / Max-Age:"Session"
  HostOnly:true
  HttpOnly:false
  Last Accessed:"Thu, 23 May 2024 16
  Path:"/"
  SameSite:"None"
  Secure:false
  Size:10
```