

## Урок 3. HTTP

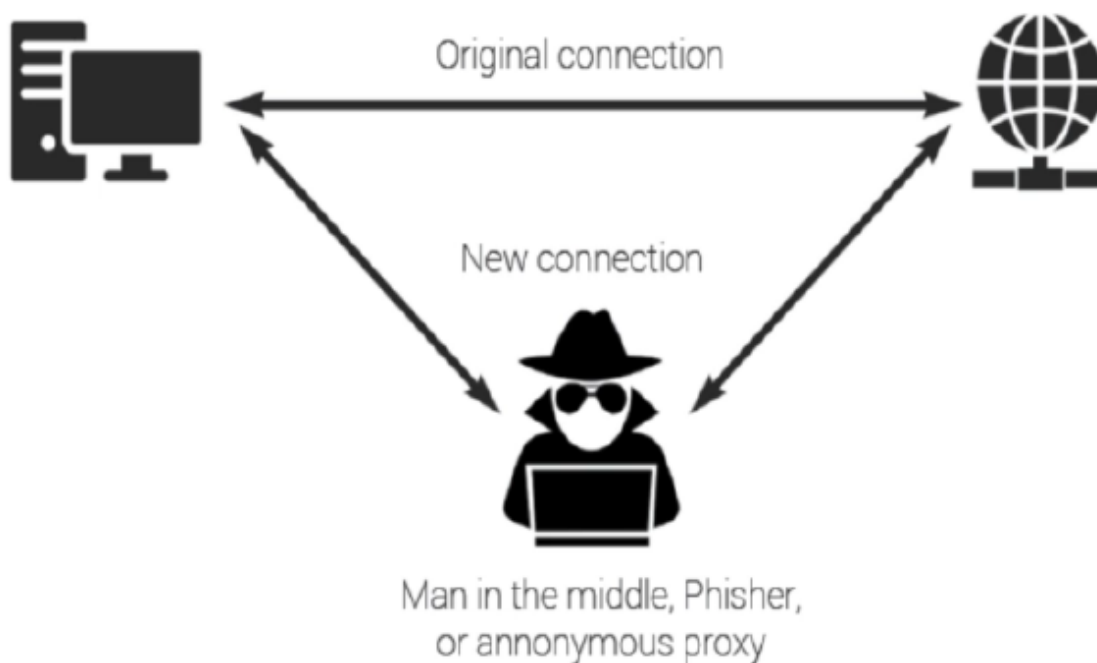
### Заметки

Referer:

```
rel = noreferrer
```

```
referrerpolicy = no-referrer
```

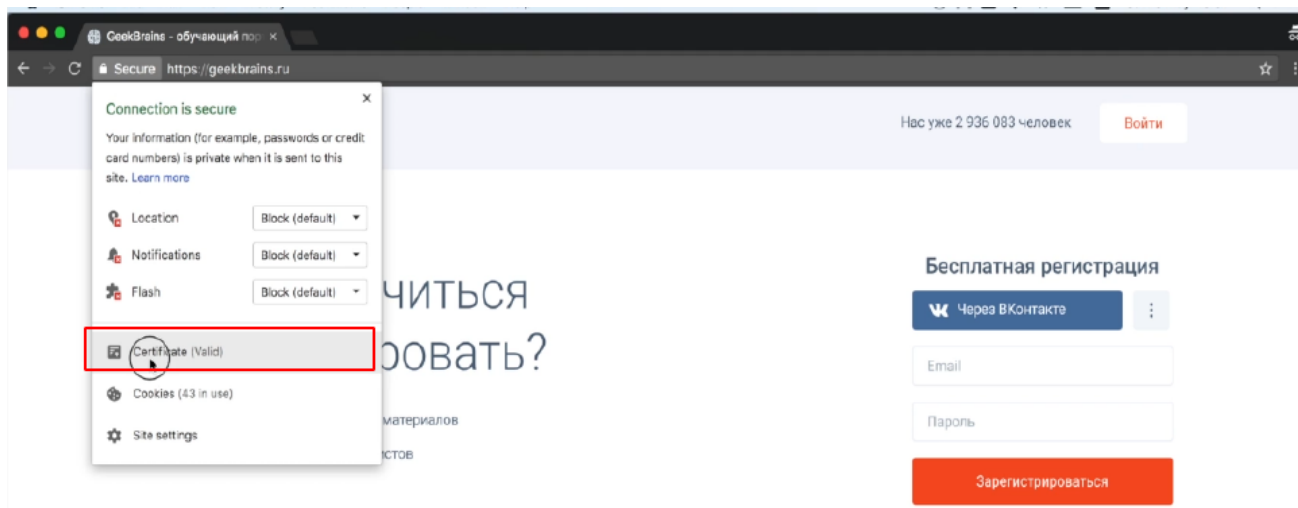
1. `<a href="http://example.com" rel="noreferrer">Example.com</a>`
2. `<a href="http://example.com" referrerpolicy="no-referrer">ReferrerPolicy Attribute</a>`



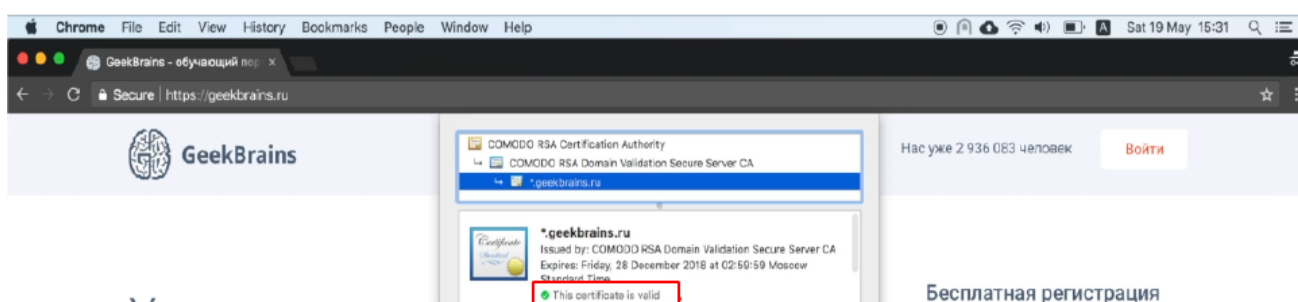
столбце Source=100.100.47.207) на IP-адрес сайта [example.com](http://example.com) Destination=93.184.216.34:

No.	Time	Source	Destination	Protocol	Length	Info
237	33.155330	100.100.47.207	93.184.216.34	TCP	78	57295 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=161877748 TSecr=0
239	33.334566	100.100.47.207	93.184.216.34	TCP	66	57295 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=161877925 TSecr=464598049
240	33.354940	100.100.47.207	93.184.216.34	HTTP	386	GET / HTTP/1.1

Wireshark расшифровывает, что это GET-запрос за главной страницей.

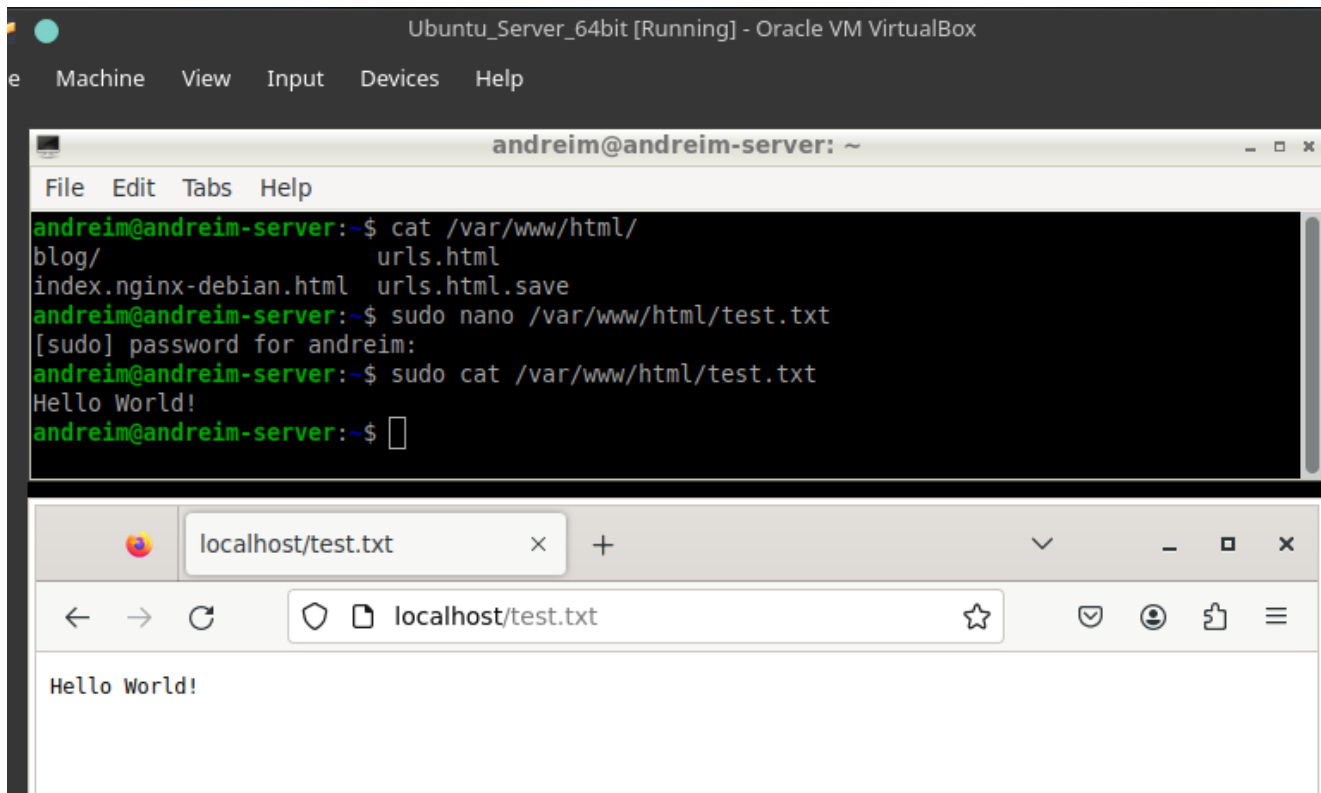


Мы получаем информацию о сертификате, можно посмотреть больше информации:



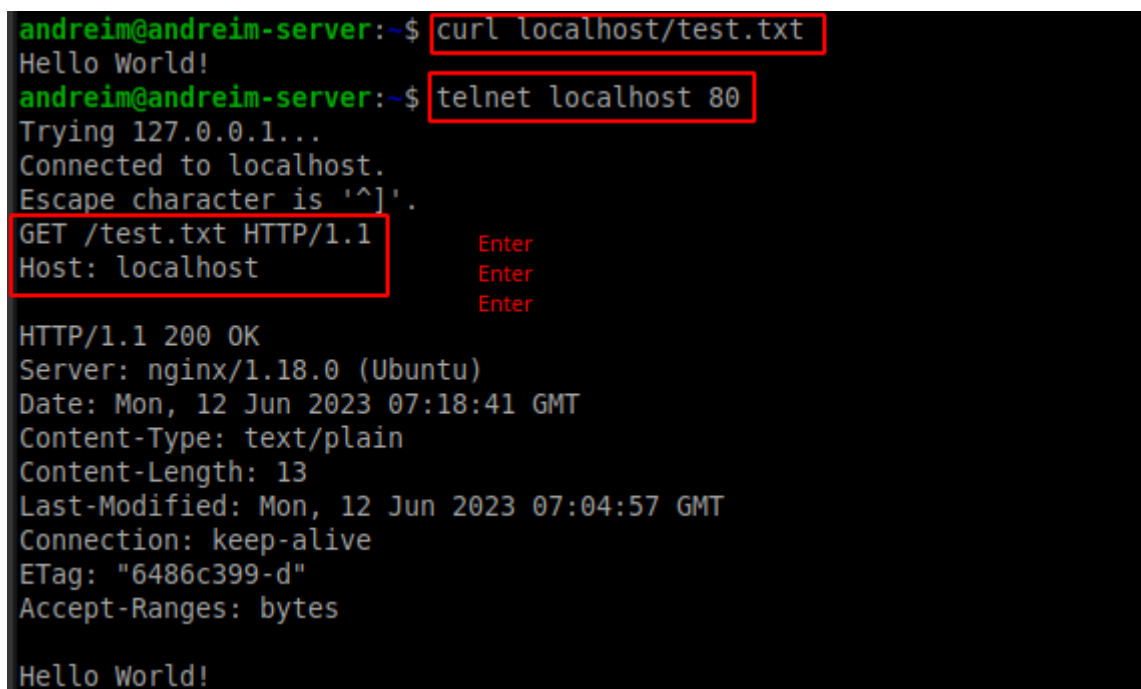
## Задание

1. Создать файл test.txt в корневом каталоге сервера. Получить этот файл через браузер.
- Установить в терминале программу curl, получить тот же файл с помощью этой программы.
  - Установить telnet или netcat, получить тот же файл с помощью одной из этих программ.



```
sudo apt install telnet curl netcat
curl -i https://geekbrains.ru | less

curl localhost/test.txt
telnet localhost 80    GET /test.txt HTTP/1.1    Host: localhost
netcat localhost 80    GET /test.txt    ...
```



```
andreim@andreim-server:~$ netcat localhost 80
GET /test.txt
Hello World!
^C
andreim@andreim-server:~$ netcat localhost 80
GET /test.txt HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 12 Jun 2023 07:27:10 GMT
Content-Type: text/plain
Content-Length: 13
Last-Modified: Mon, 12 Jun 2023 07:04:57 GMT
Connection: keep-alive
ETag: "6486c399-d"
Accept-Ranges: bytes

Hello World!
```

2. Создать на сервере файл `sensitive_info.txt`. Добавить базовую HTTP авторизацию для этого файла.

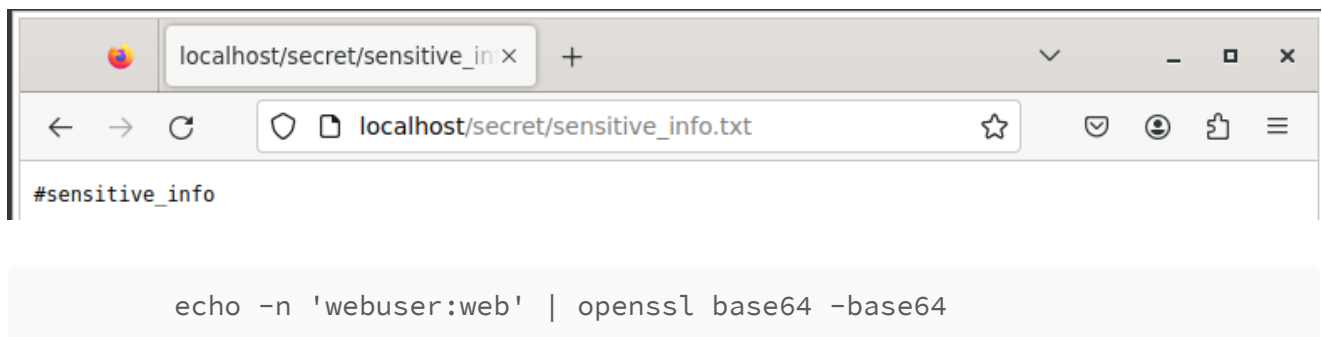
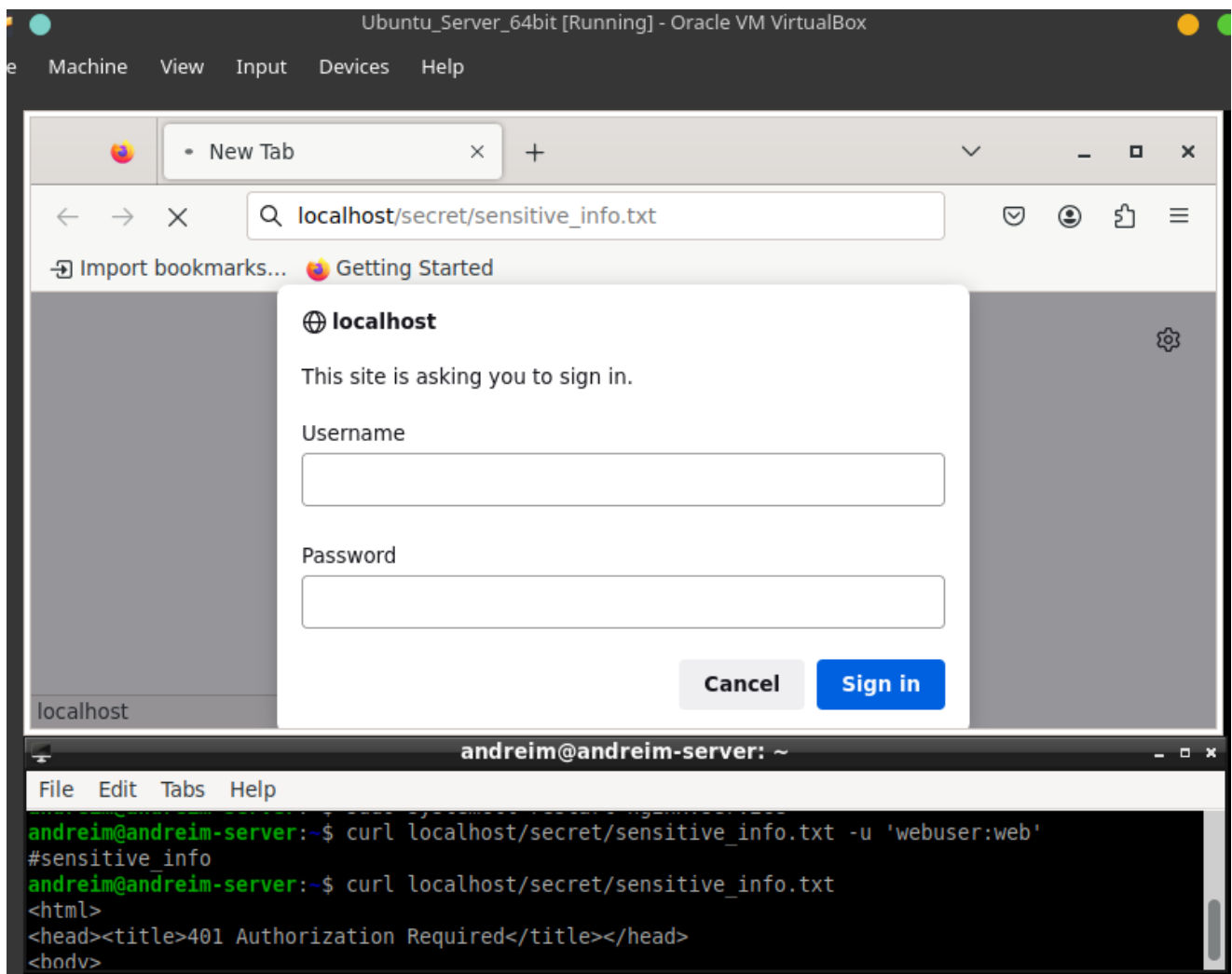
- Получить этот файл через браузер.
- Получить тот же файл с помощью `curl` и `telnet` или `netcat`.

```
sudo apt install apache2-utils
sudo htpasswd -c /var/www/pass webuser
sudo nano /etc/nginx/sites-available/default
sudo systemctl restart nginx.service
```



```
andreim@andreim-server: ~
File Edit Tabs Help
andreim@an... x andreim@an... x
GNU nano 6.2 /etc/nginx/sites-available/default *

# new add
location /secret/ {
    auth_basic "Very sensitive";
    auth_basic_user_file /var/www/pass;
}
```



```

andreim@andreim-server:~$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /secret/sensitive_info.txt HTTP/1.1
Host: localhost
Authorization: Basic d2VidXNlcjp3ZWl=

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 12 Jun 2023 09:49:53 GMT
Content-Type: text/plain
Content-Length: 16
Last-Modified: Mon, 12 Jun 2023 08:17:50 GMT
Connection: keep-alive
ETag: "6486d4ae-10"
Accept-Ranges: bytes

#sensitive_info

```

```
netcat localhost 80
```

```
...
```

- Открыть инструменты разработчика, вкладку Сеть (Network). Зайти на сайт <https://geekbrains.ru>. Проанализировать куки каждого запроса за HTML и картинками. Какие запросы уходят с куками, а какие без кук? Почему в каждом из случаев происходит именно такое поведение?

The screenshot shows a web browser window with the address bar displaying `https://gb.ru/login`. The page contains a login form with fields for "E-mail" and "Пароль", a checkbox for "Запомнить меня", a "Войти по смс" link, and a green "Войти" button. Below the form is a purple chat icon.

The Network developer tools panel is open, showing a list of requests. The "All" tab is selected, and the "Filter URLs" field is empty. The table below shows the first four requests:

Stat...	Met...	Domain	File	Initiator	Type	Transferred	Size	0 ms	1.37 min
204	POST	region1...	collect?v=2&tid=G-D11RM3RGCC&gt;	beacon	plain	418 B	0 B	28 ms	
200	POST	mc.yand...	40414440?wmode=0&wv-part=1&	xhr	gif	500 B	43 B	578 ms	
200	POST	mc.yand...	40414440?wmode=0&wv-part=1&wv-	xhr	gif	500 B	43 B	56 ms	
403	GET	frontend...	GBFont-Regular.woff	font	xml	411 B	145 B	60 ms	

Firefox -> gb.ru -> Inspect(Q) -> network / storage  
Burp Suite -> Proxy -> HTTP history -> Browser: gb.ru

**Burp Suite Community Edition v2023.5.2 - Temporary Project**

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Settings  
Comparer Logger Organizer Extensions Learn  
Intercept **HTTP history** WebSockets history Proxy settings

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension
1	https://gb.ru	GET	/			401	603	HTML	
2	https://gb.ru	GET	/__qrator/qauth_utm_v2.js			200	214211	script	js
4	https://gb.ru	POST	/__qrator/validate?pow=22&nonc...	✓		403	178		
5	https://gb.ru	GET	/qerror/403.html			403	1306	HTML	html
6	https://gb.ru	GET	/login			401	603	HTML	
7	https://gb.ru	GET	/__qrator/qauth_utm_v2.js			304	233	script	js
8	https://gb.ru	POST	/__qrator/validate?pow=107&nonc...	✓		403	178		
9	https://gb.ru	GET	/qerror/403.html			403	1306	HTML	html
10	https://gb.ru	GET	/log			401	603	HTML	
11	https://gb.ru	GET	/__qrator/qauth_utm_v2.js			304	233	script	js
12	https://gb.ru	POST	/__qrator/validate?pow=74&nonc...	✓		403	178		
13	https://gb.ru	GET	/qerror/403.html			403	1306	HTML	html
14	https://gb.ru	GET	/			401	603	HTML	

**Request** Pretty Raw Hex

1 POST /\_\_qrator/validate?pow=107&nonce=1686608759.533.g7hwb9G5EhaYuCTs&qsession=

**Response** Pretty Raw

1 HTTP/2 403 Forbidden  
2 Server: QRATOR  
3 Date: Mon, 12 Jun 2023 22:26:00 GMT

**Inspector**

Request attributes 2

Request query parameters 3

Request cookies 2

Вход | GeekBrains - обрабо

https://gb.ru/login 110%

E-mail

Пароль

☐ Запомнить меня **Войти по смс**

**Войти**

Inspector Console Debugger Network Style Editor **Storage** 34

**Cache Storage**

**Cookies**

- https://api.flocktory.com
- https://gb.ru**
- https://webchat.yc.fstrk.io
- https://www.recaptcha.net

**Indexed DB**

**Local Storage**

Filter Items

Name	Value	Domain	Path
__event...	1hqctm3g8h	.gb.ru	/
<b>_app_se...</b>	<b>038b6a64f28ab</b>	<b>.gb.ru</b>	<b>/</b>
_dc_gt...	1	.gb.ru	/
_ga_D1...	GS1.1.16866...	.gb.ru	/
_gasess...	Mon, 12 Jun ...	.gb.ru	/
_gasess...	20230612 03...	.gb.ru	/
qa	GA1.2.50228...	.gb.ru	/

Filter values

**Data**

**\_app\_session:** "a12fe742346...8b6a64f28ab"

Created: "Mon, 12 Jun 2023 22:30:42 GMT"

Domain: ".gb.ru"

Expires / Max-Age: "Session"

HostOnly: false

HttpOnly: true

Last Accessed: "Mon, 12 Jun ...22:30:42 GMT"

Path: "/"

Куки в браузере выглядят как пары "ключ-значение" и похожи на GET-параметры, используются в качестве идентификатора сессии, сохраняются данные на стороне клиента и его браузера.

Например, после запроса на логин-пароль одна из Куки, которую выставил сервер - сессионная (app\_session). Заменив логин и пароль, получем сессионную куки и доступ к аккаунту пользователя. Если знаем сессионную куки, то возможен доступ к аккаунту пользователя без знания логина и пароля ...

Куки имеют метки с жизненным циклом сессии:

Expires – время жизни куки

Domain – на какой домен проставляется куки

Secure – отправлять куки через защищенное соединение

4. • Для выполнения этого задания вам потребуется:
- 1) Настроить домены `attacker.com`, `sub.attacker.com`, `sub.sub.attacker.com`, `victim.com`. Каждый из этих доменов должен указывать на `127.0.0.1`
  - 2) Настроить установку кук для доменов. Добавьте следующий конфигурационный файл `nginx` (изменив `root` сервера на свой):

```
$ cat /etc/nginx/sites-available/cookie-research.conf
server {
    listen 80;
    server_name attacker.com;
    root /var/www/html;
```

```
location / {
    add_header "Set-Cookie" "test1=attacker-com_sub-attacker-com;
Domain=sub.attacker.com";
    add_header "Set-Cookie" "test2=attacker-com_victim-com;
Domain=victim.com";
    try_files $uri $uri/ =404;
}
```

```
}
```

```
server {
    listen 80;
    server_name sub.attacker.com;
```



```
root /var/www/html;
```

```
location / {  
    add_header "Set-Cookie" "test3=sub-attacker-com_attacker-com;  
Domain=attacker.com";  
    try_files $uri $uri/ =404;  
}
```

```
}
```

```
server {  
    listen 80;  
    server_name sub.sub.attacker.com;  
    root /var/www/html;
```

```
location / {  
    add_header "Set-Cookie" "test4=sub-sub-attacker-com_attacker-com;  
Domain=attacker.com";  
    try_files $uri $uri/ =404;  
}
```

```
}
```

``` Проведите исследование механизма проставления кук, для этого попробуйте установить следующие куки: 1. С домена`attacker.com`на домен`sub.attacker.com` 2. С домена`attacker.com`на домен`victim.com` 3. С домена`sub.attacker.com`на домен`attacker.com` 4. С домена`sub.sub.attacker.com`на домен`attacker.com`

По каждому пункту ответьте на вопросы:

1. Куда установились куки?
2. Если не установились, то почему?

Обобщите полученные знания и напишите вывод в формате: "Домен может проставлять куки для себя, для ... и ..., но не может проставлять куки для ..., ... и ...".

[ ](<https://gb.ru/lessons/330026/homework>)

5. (\*) Сгенерировать самоподписанный сертификат и разместить его на своем сервере.

