Gabriel Ivanica
342C5

TEMA 3 - PW

Access Control Flaws

1. Using an Access Control Matrix
Am selectat pe rand utilizatorii si am verificat ce drepturi au.

2. Bypass a Path Based Access Control Scheme
Am folosit Tamper Data pentru a edita requestul POST si am cerut fisierul tomcat-users.xml folosid o cale relativa :

" ../../../../conf/tomcat-users.xml"

3. LAB: Role Based Access Control

Stage 1: Bypass Business Layer Access Control
M-am conectat cu utilizatorul John ca admin. Am observat ca pot sterge angajati folosind action=DeleteProfile si am folosit acelasi action prin intermediul utilizatorului Tom interceptand requestul de viewProfile

Stage 2: Add Business Layer Access Control
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Stage 3: Bypass Data Layer Access Control
Am interceptat requestul catre viewProfile si am schimbat ID-ul angajatului pentru a vizualiza alt angajat.

Stage 4: Add Data Layer Access Control
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Remote Admin Access
Am adaugat parametru &admin=true in URL

Authentication Flaws

1. **Password Strength**
Nu se putea acesa link-ul din cerinta am folosit datele din Solution

2. **Forgot Password**
Am incercat utilizatorul admin cu raspunsurile red, blue, green (Test stupid, te pune sa incerci random)

3. **Basic Authentication**
Am inteles faza cu COOKIE si BASE64
Am sters COOKIES, m-am relogat cu utilizatorul basic si parola basic

4. **Multi Level Login 2**

Am interceptat requestul de TAN si am modificat hidden_name in Jade pentru a ma loga cu acest utilizator

5. Multi Level Login 1

Am interceptat requestul de TAN si am modificat numarul acestuia in 1 pentru a folosi TAN-ul furat

Parameter Tampering

1. Bypass HTML Field Restrictions

Am editat campul disabled sa nu fie disabled. Am interceptat requestul si am modificat datele.

2. Exploit Hidden Fields

Am interceptat requestul utilizand Tamper Data si am modificat valorea parametrului Price

3. Exploit Unchecked Email

1) Attack the admin
   Deoarce campurile nu sunt validate se poate adauga orice script utilizand
   <script>continut</script>
2) Attack another user
   Interceptare cu Tamper Data si modificat campul email in orice email posibil

4. Bypass Client Side JavaScript Validation

Am modificat butonul de submit in type=”submit” si am sters functia de onclick
Am adaugat tampenii in fiecare field si am trimis requestul.

Buffer Overflows

Am pus parametrul room_no foarte mare ~ 50KB de date, nume si prenume random si am primit ca raspuns informatiile despre VIP’s. Informatiile era in input hidden. Am folosit date din acel raspuns si m-am logat

Denial of Service

Am facut SQL Injection adaugand in campul de password ’ OR ’1’=’1
M-am logat cu 3 utilizatori

Cross-Site Scripting (XSS)

1. Phishing with XSS

Am inspectat codul. Am observat ca e un form.
In campul de search am inchis formularul si am adaugat un script javascript
In script am creat o functie care incarca o imagine avand src catre un alt site si care trimite datele dintr-un formular de locare
Am continuat cu formularul de locare care apeleaza scriptul de sus

2. LAB: Cross Site Scripting

Stage 1: Stored XSS
M-am logat cu un cont de angajat si am adaugat un script intr-unul dintre campurile din profil.
M-am logat cu un cont de admin si in momentul in care am selectat contul de angjat pentru vizualizare s-a executat scriptul adaugat.

Stage 2: Block Stored XSS using Input Validation
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Stage 3: Stored XSS Revisited
M-am conectat cu utilizatorul David si am observat faptul ca utilzatorul Bruce avea profilul atacat.

Stage 4: Block Stored XSS using Output Encoding
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Stage 5: Reflected XSS
Se poate adauga script in cadrul inputului de SearchStaff

Stage 6: Block Reflected XSS
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

3. Stored XSS Attacks
Am adaugat un <script> cu un alert in inputul de mesaj

4. Reflected XSS Attacks
Am adaugat un <script> cu un alert in inputul de card code

5. Cross Site Request Forgery (CSRF)
Am creat o imagine cu src catre pagina curenta in care am mai trimis inca un parametru

6. CSRF Prompt By-Pass
Am creat o imagine care face request catre pagina locala la care mai adauga un parametru transferFounds=4000. Apoi am mai adaugat inca o imagine cu parametru transferFounds=CONFIRM

7. CSRF Token By-Pass
Am creat 2 iframes. Primul face request catre pagina ca sa obtina tokenul, iar imediat dupa va seta SRC-ul de la al 2-lea iframe pentru a trimite tokenul catre alt site.

8. HTTPOnly Test
Am urmat instructiunile si am inteles scopul optiunii HTTPOnly

9. Cross Site Tracing (XST) Attacks
Am facut un request HTTP Trace si am printat raspunsul

Gabriel Ivanica
342C5

Session Management Flaws

1. Hijack a Session

Am inspectat COOKIES si am gasit WEEKID-ul. Am facut multe requesturi pana am descorperit WEEKID-ul lipsa cu 10904 in fata. Apoi am descarcat toolul J-Baah si am facut multe requesturi cu timestamp diferit pana am reusit sa obtin informatiile corecte adica sesiunea.

2. Spoof an Authentication Cookie

M-am logat cu ambii utilizatori si am inspectat COOKIES. Pentru ca era absolut normal sa ma prind ca valorile dupa din cookie content erau de forma „65432**numetranspus**” (logic ca nu, am observat asta in solution)
Am modificat cookie-ul in ce ar fi trebui sa fie pentru alice si a functionat.

3. Session Fixation

Am adaugat un SID in link-ul din mesaj.
Jane (adica tot io) a apasat pe link apoi s-a logat.
Hakerul (tot io) a modificat dupa SID-ul din NOVALIDSESSION in SID-ul setat precedent si am avut acess la informatiile lui Jane.

Injection Flaws

1. Command Injection

Interceptat cererea si adaugat o comanda de sistem gen ls:

BasicAuthentication.help" & ls

2. Numeric SQL Injection

SQL injection simplu. Am adaugat un ‚OR true’

3. Log Spoofing

Am adaugat text si formatare cat sa arate cum trebuie cerut

4. XPATH Injection

Am folosit un query cu dublu OR de genu (interceptat requestul si modificat)

ceva' OR 1=1 OR ’ 1’ = 1’

5. String SQL Injection

Injection: **Name' OR '1' = 1 --**

6. LAB: SQL Injection

<u>Stage 1: String SQL Injection</u>

Again: **' OR '1'=1' --**

Stage 2: Parameterized Query #1
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Stage 3: Numeric SQL Injection
Am adaugat: **OR 1=1 ORDER BY salary DESC** pentru a luat informatiile despre manager, presupunand ca el primeste cei mai multi bani

Stage 4: Parameterized Query #2
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

7. Modify Data with SQL Injection
Am inchis ghilimelele pentru a termina cererea de salariu si am continuat cu un query de UPDATE dupa ID-ul jsmith:

'; UPDATE salaries SET salary=123 WHERE userid='jsmith

8. Add Data with SQL Injection
Acelasi procedeu ca si sus dar cu un query de INSERT urmat de ingorarea tuturor informatilor ulterioare:

asd'; INSERT INTO salaries VALUES ('usernou',123); --

9. Database Backdoors
Am facut un SQL injection urmat de un query de update:

Update 101'; UPDATE employee SET salary=123123

Am creat un nou eveniment folosind codul (dat si in cerinta):

'; CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com'WHERE userid = NEW.userid

10. Blind Numeric SQL Injection
Am facut o cerere dupa ID-ul 102 si am observat ca exista in baza de date
Am interceptat requestul si am modificat cerere pentru a cauta dupa CC_number:

102 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 0);

Am incercat continuat sa incerc pana sa imi dau seama ca pinul este **2364**

11. Blind String SQL Injection
Eu ma gandisem sa incerc cu LIKE pana imi vine rau. Dupa am observat ca se poate testa fiecare litera cu SUBSTRACT si mai mare sau mai mic.
101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 1, 1) > 'J');