

# Application Security (apsi)

Lecture at FHNW

Lecture 9, 2021

Arno Wagner, Michael Schläpfer, Rolf Wagner

<arno@wagner.name>, <{michael.schlaepfer, rolf.wagner}@fort-it.ch>

# Agenda V09

▶ 12:15 – 14:00: Self-Work on WebGoat

▶ 14:15 – 15:00: Exercise Session

- Q&A WebGoat
- Walk through sample solutions

Vorlesung startet um 14:15 Uhr.  
Fragen können bereits davor auch  
im Chat deponiert werden



# A7 Cross Side Scripting

(A7) Cross-Site Scripting (XSS) >

Cross Site Scripting

## Theory

- A7.2: What is XSS?
- A7.3: Most common locations (XSS attacks)
- A7.4: XSS attacks may result in ...
- A7.5: Types of XSS (Stored, Reflected, DOM-based)
- A7.6: Reflected XSS scenario (in detail)
- A7.8: Self XSS or reflected XSS (example is a Self XSS)
- A7.9: Reflected and DOM-based XSS (what are the differences?)
- A7.12: QUIZ ;-)
  - 1:4; 2:3; 3:1; 4:2 ; 5:4




# A7 Cross Side Scripting

(A7) Cross-Site Scripting (XSS) >

Cross Site Scripting

## Exercises

- A7.2  
Die Antwort ist “yes”: Mit XSS kann innerhalb der Gleichen Origin auf alle Cookies zugegriffen werden
- A7.7  
Das Feld “Card Number” ist anfällig auf XSS. Fügen Sie z.B. “<script>alert("test");</script>” ein.
- A7.10  
Die “Base Route” ist start.mvc#test/. Alle nachfolgenden Zeichen sind Parameter welche vom JavaScript Handler bearbeitet werden.
- A7.11  
[http://localhost:8080/WebGoat/start.mvc#test/%3Cscript%3Ealert\(%22test%22\);%3C%2Fscript%3E](http://localhost:8080/WebGoat/start.mvc#test/%3Cscript%3Ealert(%22test%22);%3C%2Fscript%3E) zeigt, dass Scripts ausgeführt werden können.  
Mit z.B. Burp kann die Response von [http://localhost:8080/WebGoat/start.mvc#test/%3Cscript%3Ewebgoat.customjs.phoneHome\(\);%3C%2Fscript%3E](http://localhost:8080/WebGoat/start.mvc#test/%3Cscript%3Ewebgoat.customjs.phoneHome();%3C%2Fscript%3E) angeschaut werden  1088045964

(Achtung: URL Encoding nicht vergessen)

```
Response
Raw Headers Hex
Pretty Raw Render In Actions v
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Mon, 02 Nov 2020 09:46:18 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback":"Congratulations. You have successfully completed the assignment.",
12   "output":"phoneHome Response is 1088045964",
13   "assignment":"DOMCrossSiteScripting",
14   "attemptWasMade":true
15 }
```



# A1 Injection

(A1) Injection



SQL Injection (intro)

SQL Injection (advanced)

## Theory (Intro)

- A1.2 – A1.5: SQL Basics
- A1.6: What is SQL injection?
- A1.7: Consequences of SQL injection
- A1.8: Severity of SQL injection

## Exercises (Intro)

- A1.2 – A1.5  
Führe ein paar grundlegende SQL-Befehle durch (=Grundlage für spätere SQL Injections)
- A1.9  
WHERE first\_name = 'John' and last\_name = 'Smith' or '1' = '1' führt zum Resultat, da 1=1 immer True ergibt.
- A1.10  
SELECT \* FROM user\_data WHERE login\_count = 1 AND userid = 1 OR 1 = 1
- A1.11  
SELECT \* FROM employees WHERE last\_name = 'Smith' AND auth\_tan = " OR '1' = '1'
- A1.12  
SELECT \* FROM employees WHERE last\_name = 'Smith' AND auth\_tan = "; UPDATE employees SET salary = 100000 WHERE auth\_tan = '3SL99A';
- A1.13  
SELECT \* FROM access\_log WHERE action LIKE %'; DROP TABLE access\_log --%;  
(--& ist auskommentiert)



# A1 Injection

(A1) Injection



SQL Injection (intro)

SQL Injection (advanced)

## Theory (Advanced)

- A1.4: Blind SQL Injection ☐ Resultat / Verhalten des Injected Query gibt Aufschluss über (Ja, Nein) Fragestellungen
- A1.6: QUIZ ;-)
  - 1:4; 2:3; 3:2; 4:3 ; 5:4



# A1 Injection

(A1) Injection



SQL Injection (intro)

SQL Injection (advanced)

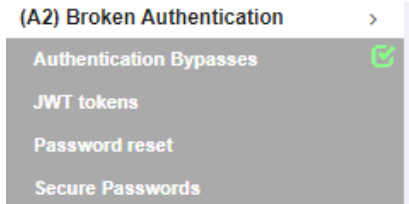
## Exercises (Advanced)

- A1.3 (zum Beispiel)  
`' ; SELECT * FROM user_system_data WHERE '1' = '1`
- A1.5  
Register.Username ist anfällig auf SQL Injektion... Try yourself ;-)





# A2 Broken Authentication

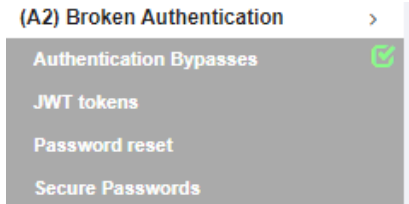


## Theory (Authentication Bypasses)

- A2.1: Possibilities of Authentication Bypasses



# A2 Broken Authentication

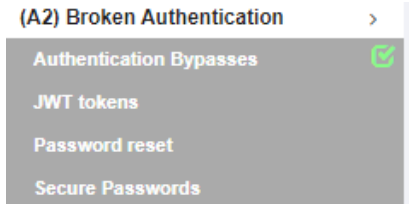


## Exercises (Authentication Bypasses)

- A2.2  
Mit Burp können die Parameter abgefangen und geändert werden: SecurityQuestions auf 8 und 9 setzen.



# A2 Broken Authentication

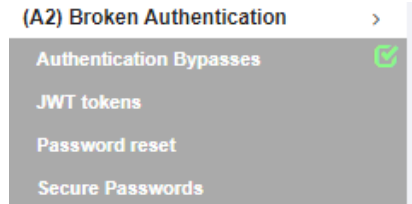


## Theory (JWT Tokens)

- A2.2: Structure of a JWT Token
- A2.3: Authentication and getting a JWT token
- A2.6: Refreshing a token



# A2 Broken Authentication

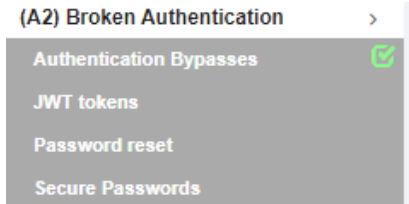


## Exercises (JWT tokens)

- A2.4  
Umschalten auf TOM und anschliessend den Refresh z.B. Burp abfangen. Das JWT ändern auf «admin = ja» und Algorithmus auf NONE (separates De-/Encoding nötig).
- A2.5  
Finden sie z.B. mit HashCat das Passwort heraus. Anschliessend können sie z.B. <https://www.jsonwebtoken.io/> das Token neu schreiben und auch signieren.
- A2.7 & A2.8  
Dieser Übung ist etwas komplizierter und die Lösung ist wiederum nicht offensichtlich, sondern man muss diese «Finden». Eine schöne Anleitung dazu ist hier zu finden: <https://pvxs.medium.com/webgoat-password-reset-2-4-b3ef02eda826>



# A2 Broken Authentication

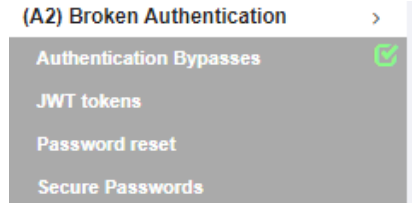


## Theory (Password reset)

- A2.3: Find out if account exists...
- A2.5: The Problem with Security Questions
- A2.6: How to create a secure password reset link
- A2.7: How to prevent abusing the password reset function



# A2 Broken Authentication

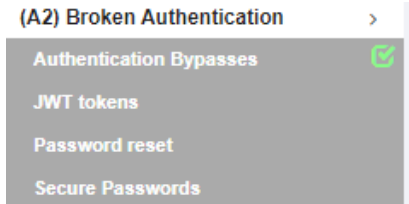


## Exercises (Password reset)

- A2.2  
Forgot your email `?[username]@[irgendwas.org]`
- A2.4  
Einfach ein paar Farben für Tom (oder Admin, oder Larry) durchprobieren. Bei Tom ist die Lieblingsfarbe z.B. «purple». Da es keinen Lock-out-Mechanismus gibt, können sie die drei Accounts auch mit einer «Cluster Bomb» von Burp automatisiert angreifen.
- A2.5  
Schauen sie sich 2-3 Security-Fragen an inkl. der Antwort, wieso das keine gute Idee ist.  
Ein interessanter Vorschlag von WebGoat ist: «If you have to pick a security question, we recommend not answering them truthfully.»
- A2.6  
Wenn man sich selber ein Password-Reset-Email sendet, findet man die Link-Notation heraus:  
`http://localhost:8080/WebGoat/PasswordReset/reset/reset-password/72e7a0eb-88f9-40a5-aecf-3fd984deb5d9`.  
Wir brauchen also den letzten String.  
Vorgehen: «Forgot Password» an Tom senden und intercepten (z.B. Burp). URL von localhost:8080 auf localhost:9090 (WebWolf) ändern.  
Wenn nun Tom auf den Email-Link klickt, erhält man in WebWolf den Link um das Passwort von Tom neu zu setzen.



# A2 Broken Authentication



## Theory (Secure Passwords)

- A2.3: NIST password standard
- A2.5: Improve security of your account
- A2.6: Storing passwords (we had that in the previous lesson «cryptography»)

(keine Übungen – ausser ein sicheres Passwort eingeben)