**Week-02 A2-Broken Authentication, A5-Broken Access Control**

1. **Explain about OWASP A2, and explain a number of the problems that would leave your application vulnerable to A2-problems**

Nogle af de ting som A2 indebærer er hvorvidt en applikation er sårbar når det gælder authentication af brugers identiteter. På OWASP ratingen er den rated således:

|  |  |  |
| --- | --- | --- |
| Threat Agents | Sårbarhed - (Security Weakness) | Pårvikning - (Impact) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| App. Specific | Udnyttelse: 3 | Prævalens: 2 | Opdagelse: 2 | Teknisk: 3 | Business ? |

|  |  |  |
| --- | --- | --- |
| Angribere har adgang til mange mange passwords, der ligger offentligt. (Offentligt pga. leaks etc.) | Session håndtering er meget vigtig at håndtere korrekt. | Angribere skal kun have adgang til få, hvis ikke kun en enkelt, admin bruger for at overtage systemet. |

Problemer der vil gøre en applikation sårbar over for A2-problemer ville for eksempel være:

* [Credential Stuffing](https://owasp.org/www-community/attacks/Credential_stuffing)
* [Brute Force Attack](https://www.techopedia.com/definition/18091/brute-force-attack)
* Default, weak or well-known passwords
* Brug af svag glemt/mistet password funktion
* Manglende eller ineffektiv to-faktor godkendelse
* Blottet session-ID

1. **For each of the problems explained above, explain how this problem could have been prevented**

Disse problemer kunne løses på følgende måder:

* Crediential Stuffing og Brute Force Attack
* Kan løses ved implementation af multi faktor login.
* Brute Force Attack kan også løses ved et maksimalt antal af forsøg man kan foretage i løbet af en given periode.
* Default, weak, WK pws
* Kan løses ved ikke aldrig at gøre applikationen offentlig/udgive den før at alle testbrugere/admins er slettet, da de typisk har et default password.
* Man kan implementere et password-check som sammenligner med et dataset af kendte/ofte brugt passwords.
* To-faktor login
* Implementering af et anerkendt to-faktor system.
* Session-ID
* Session-ID’er må aldrig blive blottet, da det er nøglen for dig til din identitet på applikationen - den må derfor ikke så i URL’en for eksempel. Derudover bør der være automatisk log ud, hvis en session er inaktiv, samt id’et skal invalideres efter. Alt dette bør gøres server-side.

1. **Explain the idea behind the term Credential Stuffing**

Credential Stuffing er et automatiseret angreb der gør brug af en liste bestående af ofte brugte/kendte bruger og password kombinationer. Ideen er at disse kombinationer er så normale, at der er en chance for at applikationen også har en sådan bruger med en af disse kombinationer.

1. **Demonstrate how you, in practice, have, or could, remove some of the vulnerabilities listed above**

Se **DEMONSTRATION 1.py** for et password strength validation eksempel

1. **Explain about OWASP A5, and explain a number of the problems that would leave your application vulnerable to A5-problems**

Access control tvinger en politik ind i systemet om at brugere ikke kan agere uden for deres tilladte muligheder. Hvis der er fejl i denne access control kan brugere, uden adgang, agerer som bruger med tilladt adgang. Nogle af disse problemer ville specifikt være:

* Adgang forbi access control ved modifikationer til URL, HTML siden eller ved brug af et API attack tool.
* Tilladelse af ændring af primary key til en anden person, som ville give mulighed for at se og redigere en andens bruger.
* Forhøjning af privilegium, at kunne agerer som en bruger uden login og kunne agere som en admin men være logget ind som en normal bruger.
* Metadata manipulation, som en bruger der redigerer en JSON Web Token / Cookie / andet field til at give dem en højere brugerstatus eller mere adgang.

1. **For each of the problems explained above, explain how this problem could have been prevented**

* Implementer access control mekanismer én gang, og genbrug dem gennem applikationen, minimerer også CORS brug.
* Log access control fejl og giv admins besked når det giver mening, f.eks. ved gentagende fejl.
* JWT token skal invalideres efter server logout, implementer funktionelle test til at dette system fungerer.

1. **Demonstrate how you, in practice, have removed some of the vulnerabilities listed above**
2. **Demonstrate at least one practical example of a A5 Vulnerability, and show how you have solved the problem (removed the vulnerability)**