Übungsblatt 08

Thomas Samy Dafir, Lex Winandy

Aufgabe 1

XSS beizeichnet das einschleusen von Fremdcode (meist Schadcode) auf eine Website. Eine Site ist durch XSS angreifbar, wenn user-input unvalidiert verwendet wird. Man untercheidet 2 Arten von XSS:

- Nicht persistent: Durch übergebene Argumente, die am Server nicht validiert werden, kann direkt Schaden angerichtet werden, wie etwa Zugriff auf nicht freigegebene Daten oder überschreiben von Daten am Webserver.
- Persistent: Gleiche Ursache: keine Validierung. Hier wird jedoch Schadcode eingeschleust, der bleibt, da er entweder in einer Datenbank abgelegt, oder direkt auf der Website verwendet wird. Standarsbeispiel dafür ist ein Gästebuch, ohne Validierung der User-Einträge.

document.location ermöglicht es, die aktuelle Seite zu wechseln. Dadurch könnte der User z.B. auf eine gleich aussehende, aber bösartige Seite umgeleitet werden. Verhindert werden kann XSS durch Validierung der Eingaben und am Besten durch Verwendung eines Positive-Security-Models: Der Entwickler gibt genau an, welche Eingaben erlaubt sind, anstatt welche nicht erlaubt sind.
Lösung:
Positive-Security-Model: Zuerst wird whitespace am Anfang und Ende der Eingabe entfernt, dann ein Regex Pattern überprüft, das nur matched, wenn die eingabe eine Ganz- oder Kommazahl ist. Abschließend wird ein float-Cast durchgeführt. Matched das Pattern nicht, oder (was nicht mehr vorkommen sollte) wirft der Cast einen ValueError, wird die Eingabe nicht verwendet und der User angewiesen es erneut zu probieren.
Links zur Lösung auf den persönlichen Seiten.

Aufgabe 2

Templates vereinfachen den Entwicklungsprozess, da Code z.B. HTML Code wiederverwendet werden kann, und nicht immer wieder neu geschrieben werden muss. Diese Templates können dynamisch angepasst und an den Anwendungsfall angepasst werden. Templates werden in einem beliebigen Format gespeichert und Platzhalter eingefügt. Diese Templates werden dann in python referenziert und für alle Platzhalter Werte eingefügt. Die Verwendung von Templates trägt auch dazu bei, weniger Fehler zu verursachen. Als Beispiele wurden Files inkludiert, Platzhalter, Inheritance, sowie Schleifen verwendet. Links zu Lösungen auf den persönlichen Seiten.

Aufgabe 3

Für diese Aufgabe verwenden wir die Lösung aus Aufgabe 1, entfernen jedoch die regex-Validierung, damit alle falschen Eingabewerte beim float-Typecast eine ValueError auslösen und wir $try - except$s zeigen können.
Lösung:

- 2 Templates wurden erstellt: Formular und Ergebnisseite.
- Die Templates wurden mit Platzhaltern versehen: Ergebnisse, Fehlermeldung und CSS Klasse für den roten Rahmen.
- Mako wurde in den python code integriert: Template-directory, module-directory und encoding gesetzt.
- Wird die Seite das erste Mal geladen (Formulardaten leer), wird das Formular-Template verwendet, ohne Fehlermeldung und CSS Klasse geladen.
- Ist eine Eingabe vorhanden, aber falsch, wird wiederum das Formular geladen, jedoch der rote Rahmen und eine Fehlermeldung eingefügt.
- Ist die Eingabe in Ordnung, wird die Ergebnisseite geladen, und Werte für die Platzhalter übergeben.

Aufgabe 4

-