M183
Applikationssicherheit Implementieren

Projekt 2017

Wertigkeit: 20% zu Schlussnote
Maximum in Zweiergruppen
Abgabe: .NET Projektdatei (Solution mit allen Ressourcen) muss vollständig auf Github versioniert und vorhanden sein (public repo).

Allerspäteste Abgabe: Samstag 20. Januar 2018, 12.00

Frühzeitige Abgaben bitte an juerg.nietlispach@gibz.ch melden mit Angabe des Git Repos.

URL Git-Repo:

Klasse

Teammember A

Name:

Vorname:

Teammember B

Name:

Vorname:

Anforderungen an die Applikation „Mini-Blog-Engine“ (30p)

Es soll eine funktionierende MVC .NET Webapplikation „from scratch“ erstellt werden, welche auf den Inhalten und den Tutorials des Moduls 183 basiert.

Wichtig:

- Jeder Arbeitsschritt muss in einem eigenen Commit auf Github versioniert werden! Der Code muss mit individuellen und nachvollziehbaren Kommentaren versehen sein.
- Fragen bzw. Begründungen müssen umfangreich erklärt bzw. aufgezeigt werden – wo möglich mit Beispielen ergänzt.

Setup

#	Anforderung / Aufgabe	Pkte	Pkte err. / Bem.
1	.NET MVC Applikation erstellen	0.5	
2	Filebasierte Datenbank erstellen. Tabellen: - User (z.B. id, vorname, nachname, mobile phonenumber, username (als hash), password (als hash), role, status) - Token (z.B. id, user_id, tokenstring, expiry, deleted) - Post (z.B. id, user_id, title, description, content, createdon, modifiedon, deletedon) - Comment (z.B. id, user_id, post_id, comment, createdon) - Userlogin (user_id, user_ipaddress, sessionid, createdon, modifiedon, deletedon) - Userlog (user_id, action)		Wird als .mdf-Datei und als SQL-Dump zur Verfügung gestellt.
3	Nexmo SMS API für 2-Factor-Auth bereitstellen	0.5	

Login & Logout

#	Anforderung / Aufgabe	Pkte	Pkte err. / Bem.
1	Login Formular für Eingabe von Benutzernamen, Passwort unter der URL /login erstellen.	1	
2	Beim Absenden des Login-Formulars soll geprüft werden, ob ein Benutzer mit den angegebenen Credentials überhaupt existiert.	1	
3	Existiert ein User, soll ein jeweils zufälliges und für den Vertriebskanal SMS passendes Token generiert werden.	1	
4	Dieses Token soll zusammen mit der User_Id in die Token-Tabelle gespeichert werden. Das Token soll 5‘ gültig sein.	1	
5	Das Token soll nun via Nexmo API an den einzuloggenden	1	

	Benutzer versendet werden.		
6	Die Applikation soll nun eine zweite Login-Maske anzeigen, wo das per SMS versendete Token eingegeben werden kann (Benutzername und Passwort sollen als hidden-input Felder ins Formular mitgegeben werden).	1	
7	Beim Absenden des zweiten Formulars sollen die Credentials des Users zusammen mit dem Token geprüft werden.	1	
8	Stimmen die Credentials und das Token, soll eine SESSION erzeugt werden und der Login in die User-Log Tabelle sowie in die Userlogintabelle geschrieben werden. Das gebrauchte Token muss als gelöscht markiert werden.	1	
9	Zusätzlich soll der Rolle des authentifizierten Benutzers entsprechend auf das Dashboard weitergeleitet werden. Es sollen ein User und ein Admin Dashboard vorgesehen werden. Also /user/dashboard bzw. /admin/dashboard	1	
10	Sollte einer dieser oberen Schritte aus technischen Gründen fehlschlagen, soll dies dem Benutzer entsprechend angezeigt werden (Fehlermeldung) und in der User-Log-Tabelle eingetragen werden.	1	
11	Hat ein Benutzer mehr als 3 mal hintereinander das Passwort bzw. das Token falsch eingegeben, soll der Benutzer Blockiert werden (z.B. status = blocked)	1	
12	Bei einem Logout sollen die SESSION Informationen gelöscht- und auf die Login-Form weitergeleitet werden.	1	
13	Zusätzlich soll der Eintrag aus der Userlogin-Tabelle mit der aktuellen SESSION-ID als gelöscht markiert werden und ein Eintrag in der User-Log Tabelle gemacht werden.	1	

User und Admin Dashboards

#	Anforderung / Aufgabe	Pkte	Pkte err. / Bem.
1	Das User-Dashboard soll alle Beiträge (Post-Datenbank) des aktuell eingeloggten Benutzers anzeigen.	1	
14	Das Admin-Dashboard soll alle Beiträge (Post-Datenbank) aller Benutzer anzeigen (inkl. den gelöschten)	1	
15	Das Admin-Dashboard soll mit einer Suchfunktion erweitert werden (über die Felder Title, Description und	1	

	Content – sie können LIKE ‚%...%' verwenden) und vor SQL-Injections gesichert werden. Die gefundenen Resultate sollen angezeigt werden.		
16	Die passwortgeschützten Bereiche sollen vor einer URL-Tampering Attacke (auf der Rolle basierend) gesichert werden.	1	

Public Dashboard / Startseite

#	Anforderung / Aufgabe	Pkte	Pkte err. / Bem.
1	Auf der Startseite sollen alle veröffentlichten Posts (status published) angezeigt werden.	1	
2	Für jeden Post soll eine Detail-Ansicht erstellt werden.	1	
3	Ebenfalls sollen vorhandene Kommentare in der Detailansicht angezeigt werden.	1	
4	Es soll ein Formular für Kommentare in die Post-Detailansicht eingefügt werden.	1	
5	Das Absenden des Formulars soll den Kommentar in die Kommentar-Tabelle speichern, nach dem der Kommentar auf Länge geprüft wurde (200 Zeichen). Der Kommentar muss so „sanitized" werden, dass nur Plaintext in die Datenbank gespeichert wird (Achtung, auch bei Plaintext SQL-Injections beachten)	1	
6	Die Applikation soll so angepasst werden, dass es via URL-Tampering nicht möglich ist, ‚verborgene' Posts anzusehen (also Posts, welche nicht den status published haben bzw. gelöscht wurden). Ebenfalls soll es nicht möglich sein, die ID der Posts zu erraten (URL-Tampering).	1	

API für Posts

#	Anforderung / Aufgabe	Pkte	Pkte err. / Bem.
1	Unter /api soll ein JSON-API der Blogposts erstellt werden.	1	
2	Zugriff auf das API soll durch ein API Token (als URL – Parameter: apitoken=TOKEN) oder durch Digest Authentication realisiert werden (Annahme: die Kommunikation erfolgt über SSL – dies ist hier jedoch	1	

	keine Anforderung an die Applikation).		
3	Ein API-Token kann als Passwort (gehashed) in der User-Tabelle auf einem API-User abgelegt werden.	--	
4	Unter /api/posts sollen alle veröffentlichten Posts im json-Format retourniert werden.	1	
5	Unter /api/posts/{id} soll der Post (title, description, content) im JSON-Format retourniert werden inkl. zugehörige Kommentare.	1	

Fragen (Sollen als Kommentar im HomeController.cs beanwortet werden) (2P)

1) Warum haben Sie sich für gerade für den Hash Algorithmus (Usernamen & Passwort) entschieden?
2) In der User-Login-Tabelle ist noch ein Feld für die IP-Adresse Reserviert. Welche Attacke lässt sich dadurch verhindern?
3) Erklären Sie, wie diese Attacke genau funktioniert und inwiefern die Gegenmassnahmen die Attacke vereitelt?