# Arbeitsblatt 8: Umsetzen der Sicherheitsmassnahmen in der Webapplikation *Bilderdatenbank*

## Ausgangslage

Sie haben sich mit der Sicherheit von Webanwendungen auseinandergesetzt und jeweils ein Thema präsentiert (Arbeitsauftrag *AB151-06SicherheitWebanwendungen*). Nun geht es darum, die konkreten Massnahmen daraus abzuleiten, unter folgenden Gesichtspunkten:

* Massnahmen, die die Entwickler von Webapplikationen treffen können.
* Massnahmen für das Projekt *Bilderdatenbank*.

Weiter gibt es Vorkehrungen, um die Daten der Benutzer besser zu schützen. Diese haben wir im vorhergehenden Modul 133 zum Teil bereits besprochen (z.B. Formatvorgaben für das Passwort). Solche Vorkehrungen werden in diesem Arbeitsauftrag ebenfalls untersucht.

## Ziele

* Sie leiten konkrete Massnahmen ab, mit der eine Webapplikation vor Bedrohungen geschützt werden kann.
* Die Massnahmen sollen spezifisch für die *Bilderdatenbank* erarbeitet werden. Bedrohungen, die bei dieser Anwendung nicht relevant sind, sollen nicht weiter untersucht werden.

## Unterlagen / Quellen

* Ihnen stehen die Handouts für alle 10 vorgestellten Bedrohungen zur Verfügung, unter:  
  K:/inf2014.4f-151/Sicherheit\_Praesentationen
* Beschränken Sie sich wenn möglich auf die in den Handouts vorgestellten Massnahmen. Eigene Ideen sind natürlich willkommen.
* Bei Bedarf können Sie ergänzende Informationen im Internet holen.

## Vorgehen

* 1. Schritt: Beantworten Sie die Fragen zu Sicherheit und Massnahmen (Tabelle auf Seite 2).
* 2. Schritt: Diskutieren Sie mit Ihrem Pultnachbarn die Resultate und nehmen Sie evtl. Korrekturen vor.
* 3. Schritt: Die Massnahmen werden im Klassenverband besprochen.
* 4. Schritt: Zusätzliche Massnahmen für die Bilderdatenbank werden thematisiert.
* 4. Schritt: Fahren Sie mit der Einzelarbeit *Bilderdatenbank* weiter und setzen Sie die beschlossenen Massnahmen um, soweit das bereits möglich ist. Einige werden Sie erst in den folgenden Wochen in Angriff nehmen können.

## Zeit: 2 Lektionen

## Auftrag

#### Aufgabe 1

Füllen Sie die folgende Tabelle *in Einzelarbeit* mithilfe der Handouts, die Sie elektronisch erhalten haben, aus. Überlegen Sie jeweils, ob die Massnahmen vom Entwickler, dem Serveradministrator, dem Endbenutzer oder von sonst einer Instanz (z.B. den Betreibern von Web-Servern) getroffen werden sollten (Mehrfachnennungen möglich). Überlegen Sie auch, ob eine Gefahr für die Bilderdatenbank-Applikation besteht und wenn ja, wie gross der Schaden sein könnte.

Falls Massnahmen beim Entwickler liegen, nennen Sie die konkreten Vorkehrungen, die Sie zu treffen gedenken – wiederum im Hinblick auf die Bilderdatenbank.

Zeitbedarf: 20 Minuten

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bedrohung** | **Massnahmen durch** | | | | **Gefahr für *Bilder-DB*** | **Konkrete Massnahmen in *Bilderdatenbank* durch Entwickler** |
| **Entw.** | **Admin** | **Benutzer** | **Sonst** |
| SQL-Injection | ❑ | ❑ | ❑ | ❑ | ❑ | PDO Prepared Statements |
| Cross-Site-Scripting | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| Session Fixation/ Session Hijacking | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| Cross-Site- Request-Forgery | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| Directory Traversal | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| E-Mail-Injection | ❑ | ❑ | ❑ | ❑ | ❑ | Keine Email Formulare |
| Man-in-the-Middle/ Man-in-the-Browser | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| Denial of Service DoS / DDoS | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| DNS-Spoofing/ Cache Poisoning | ❑ | ❑ | ❑ | ❑ | ❑ |  |
| Phishing | ❑ | ❑ | ❑ | ❑ | ❑ |  |

#### Aufgabe 2

Diskutieren Sie mit Ihrem Pultnachbarn die Ergebnisse. Wollen Sie dieselben Massnahmen treffen? Falls es Differenzen gibt, argumentieren Sie pro und contra. Kommen Sie schliesslich zu einem Konsens und korrigieren Sie allenfalls Ihre Massnahmen.

Zeitbedarf: 10 Minuten

#### Aufgabe 3

Es ist Pflicht des Entwicklers, die Daten der Benutzer applikatorisch so gut wie möglich zu schützen, damit Daten nicht von unberechtigten Personen eingesehen werden können. Etliche wichtige Massnahmen haben wir bereits angeschaut, damit sind die Daten in der Datenbank gut geschützt. Wie sieht es aber mit den Bildern aus, die im Dateiverzeichnis des Servers abgelegt werden? Wenn wir die Bilder unterhalb des Webroots in Verzeichnissen ohne weiteren Schutz speichern, kann jeder Benutzer (auch ohne Login!) alle Bilder via URL/Webbrowser anzeigen – vorausgesetzt, er kennt die Verzeichnis- und Dateinamen. Wenn wir die Bilder andererseits in Verzeichnissen ausserhalb des Webroots ablegen, können die Bilder nur noch via PHP-Skript angezeigt werden. Der PHP-Web-Benutzer braucht die entsprechenden Berechtigungen, was ein Sicherheitsproblem darstellen kann (Stichwort *Directory Traversal*).

Überlegen Sie sich Möglichkeiten, wie Sie die Bilder effektiv vor unberechtigtem Zugriff schützen können. Einerseits müssen Sie verhindern, dass Bilder direkt via URL/Browser angezeigt werden können. Andererseits darf jeder Benutzer nur seine eigenen Bilder anzeigen können.

Nehmen Sie sich 10 Minuten Zeit, um mit Ihrem Pultnachbarn mögliche Lösungen zu diskutieren, danach werden diese im Plenum besprochen. Tragen Sie die Ergebnisse in dieses Dokument ein.

*Hinweis*: Via HTML/Webbrowser kann nur auf Dateien zugegriffen werden, die unterhalb des Webroots liegen. Der PHP-Web-Benutzer jedoch hat normalerweise Lese- und Schreibberechtigung auch auf Verzeichnisse oberhalb des Webroots. Mit andern Worten: Die Berechtigungen von HTML und PHP sind unterschiedlich und können sich beim Schutz der Bilder zunutze gemacht werden!

#### Zusatzaufgabe 4

Wir alle kennen das Konzept von Registration, Login, Sessions, Member-Bereich und Logout und haben dieses in andern Projekten bereits angewendet. Wir kennen es natürlich auch von unzähligen Webanwendungen, die darauf aufbauen. Haben Sie sich schon mal überlegt, warum es beim Erstellen von Passwörtern Vorgaben bezüglich des Formats gibt? Warum Passwörter verschlüsselt in die Datenbank abgelegt werden? Welches der beste Algorithmus zum Verschlüsseln bzw. Verschleiern des Passwortes ist?

Beantworten Sie dazu die folgenden Fragen. Die Lösungen werden zu einem späteren Zeitpunkt in der Klasse diskutiert bzw. vorgestellt.

1. Was kann passieren, wenn der Benutzer ein sehr unsicheres Passwort erstellt? (Z.B. ein sehr kurzes Passwort oder ein Wort aus dem Wörterbuch.)  
   Dictionary Attack.
2. Was kann passieren, wenn die Passwörter unverschlüsselt bzw. mit einem schwachen Algorithmus verschlüsselt in der Datenbank abgelegt sind? (Auf die Begriffe *Verschlüsselung* bzw. *Verschleierung* gehen wir in Punkt 4 ein.)  
   Wenn DB gestoheln -> alle UAC compromised. Viele User verwenden Passwörter auf mehreren Services
3. Wie wir unter Punkt 1 gesehen haben, sollten Passwörter sicher sein. Der Programmierer kann dies mind. teilweise sicherstellen, indem er Mindestanforderungen bezüglich des Formates festlegt. Wie sollten diese Anforderungen aussehen?  
   Min. 7 Charakters, nicht Username, nicht 1000 häufigste Wörter.

Source: <http://security.stackexchange.com/questions/93611/password-rules-should-i-disallow-leetspeak-dictionary-passwords-like-xkcds-t/93618#93618>

1. Wie wir unter Punkt 2 gesehen haben, sollten Passwörter immer verschlüsselt in die Datenbank gespeichert werden. Wobei *verschlüsselt* nicht ganz korrekt ist: Es handelt sich um eine *Verschleierung* mittels Einweg-Hash-Algorithmus. Der Ursprungswert kann also nicht mehr wiederhergestellt werden, aber die Eingabe des Passwortes kann wiederum verschleiert und mit dem Wert in der Datenbank verglichen werden.  
   In PHP gibt es die vier Hash-Funktionen *md5*, *sha1*, *password\_hash* und *crypt* (*crypt* lassen wir hier mal weg). Was sind die Unterschiede von *md5* und *sha1* verglichen mit *password\_hash*? Welchen Algorithmus sollte man aktuell verwenden?

Informationen dazu finden Sie u.a unter:  
<http://php.net/manual/de/faq.passwords.php> und  
<http://www.a-coding-project.de/ratgeber/php/verschluesseln-und-entschluesseln>

Man sollte password\_hash verwenden, da es am sichersten ist. Md5 und SHA1 können beide zu einfach gpu parallelisiert werden und sind zu schnell, sind dh. Einfach wie offline-attack crackbar. Password\_hash mit so vielen rounds wie zulässig sollte verwendet werden (2^15 -> 32768 rounds in meinem Fall. Geht ca. eine Sekunde.). Bcrypt basiert auf einem modifizierten Blowfish Verschlüsselungsmechanismus.