Stefan Waidele
Ensisheimer Straße 2
79395 Neuenburg am Rhein
Stefan.Waidele@AKAD.de

AKAD University
Immatrikulationsnummer: 102 81 71

Modul SWE02 — Softentwicklung
Assignment

Sicherheit bei Web–Anwendungen

Betreuer: Prof. Paul Kirchberg

2. Mai 2014

AKAD University

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CAPTCHA	Completely Automated Public Tourint Test to Tell Computers and Humans Apart
CSS	Cascading Style Sheets
DoS	Denial of Service
DDoS	Distributed Denial of Service
ERP	Enterprise Resource Planning
HTML	Hypertext Markup Language
HTTP	Hypertext Tranfsfer Protocol
HTTPS	Hypertext Tranfsfer Protocol Secure
OWASP	Open Web Application Security Project
PDO	PHP Data Objects
SCM	Supply Chain Management
SQL	Structured Query Language
WWW	World Wide Web
XSS	Cross Site Scripting

1 Einleitung

1.1 Ziele dieser Arbeit

In dierser Artbeit sollten mögliche Angriffstechniken auf Webanwendungen sowie geeignete Gegenmaßnahmen dargestellt werden.

Auch wenn eine Internetanwendung wie etwa ein Web–Shop nur einem beschränkten Personenkreis zur Verfügung stehen soll, so ist zumindest das Anmeldeformular[1] aufgrund der Struktur des World Wide Web (WWW) doch für Angriffe beliebiger Personen erreichbar. Daher sind schon bei der Implementierung eine Reihe von Sicherheitsaspekten zu beachten, um dieser umfangreichen Bedrohung entgegen zu wirken.

1.2 Aufbau der Arbeit

Zunächst werden im Kapitel 2 erörtert, warum der Sicherheitsaspekt bei der Erstellung eines Web–Shops ein wichtig ist. Es werden Angriffsziele, Szenarien genannt, sowie die möglichen wirtschaftlichen Konsequenzen eines erfolgreichen Angriffs aufgezeigt.

Im Kapitel 3 wird gezeigt, wie Angriffe mit dem Methoden Brute Force, Man in the Middle, SQL–Injection sowie Cross Site Scripting (XSS) durchgeführt und abgewehrt werden können.

1.3 Abgrenzung

Angriffe im Internet können an vielen Punkten der Datenübertragung statt finden. Diese Arbeit beschäftigt sich hauptsächlich mit den Methoden, auf die schon

[1] Zumeist gilt dies für noch mehr Seiten mit Benutzerinteraktion, wie z.B. die Grundfunktionalität des Warenkorbs, Suchfunktione, etc.

bei der Softwareentwicklung geachtet werden sollte. Risiken, die in den Verantwortungsbereich des Serverbetreibers oder WLAN–Anbieters, oder gar des Kunden (Datendiebstahl bzw. Keylogger auf dem Kunden–PC) liegen, werden hier nicht weiter behandelt, auch wenn sie im Grundlagenteil der Vollständigkeit halber erwähnt werden.

Außerdem wird davon ausgegangen, dass der Webshop den Kaufprozess lediglich bis zur Bestellung abbildet. Weitere Schritte, speziell die Bezahlung, werden auf herkömmlichem Weg abgewickelt.

2 Grundlagen

2.1 Schützenswerte Güter

Bei der Sicherheit eines Webshops bestehen die gleichen Grundbedrohunge wie bei der der Informationssicherheit im allgemeinn. Diese sind Vertraulichkeit, Verläßlichkeit, Verbindlichkeit, Verfügbarkeit.[2]

- **Vertraulichkeit:** Die Daten sollen nicht in die Hände von Unbefugten geraten bzw. unberechtigt gelesen werden.
- **Verläßlichkeit** (Integrität): Die Daten sollen nicht verändert werden.
- **Verbindlichkeit** (Authentizität): Die Herkunft der Daten bzw. die Identität des Urheber ist zweifelsfrei zu ermitteln.
- **Verfügbarkeit:** Die Daten bzw. Programmfunktionen sind jederzeit (bzw. immer wenn benötigt) verfügbar.

2.2 Angriffsziele

Ein Webshop bildet den Verkaufsprozess zwischen Händler und Kunde ab. Auch wenn die Angriffe an diesem Bindeglied stattfinden, können sich die oben genannten Grundbedrohungen auf unterschiedliche Ziele richten:

- **Der Webshop selbst:** Hier wird direkt auf die Verfügbarkeit bzw. Zuverlässigkeit der Shopsoftware gezielt
- **Die Unternehmensdaten:** Auch wenn bei oberflächlicher Betrachtung ein Webshop nur Produktdaten und Preislisten enthält, so sind hier durch die Kaufabwicklung weitere, durchaus sensible Daten wie z.B. Verkaufsstatistiken oder Rabattstaffeln gespeichert. Wird keine Stand–Alone Shoplösung

[2] Aufzählung und Erklärungen vgl. [Dix (Hrsg.), 2008]

sondern eine in das Supply Chain Management (SCM) bzw. Enterprise Resource Planning (ERP) integrierte Lösung verwendet, so können erfolgreiche Angreifer noch weitere Daten des Unternehmens erlangen.

- **Die Kundendaten:** Kundenadressen und vergangene Bestellungen können für gezieltes und effektives Direktmarketing genutzt werden. Kreditkarten– oder Kontodaten, Benutzernamen und Passworte bieten eine Grundlage für weitergehende Angriffe auf die Benutzerkonten des Kunden bei anderen Anbietern und Diensten, bis hin zum Identitätsdiebstahl.[3]

2.3 Angriffsfolgen

Ein erfolgreicher Angriff auf den Webshop kann neben dem direkten wirtschaftlichen Schaden beim Anbieter selbst auch weitreichenden Schaden bei den Kunden haben. Auch kann der Vertrauensverlust bei den Kunden und der Immageschaden der durch mangelnde Sicherheitsmaßnahmen entstehen kann sehr schwer wiegen.

[3] vgl. [Fuest, 2013]

3 Angriffsarten und Gegenmaßnahmen

Je nach Ziel und Absicht des Angreifers kommen unterschiedlichen Angriffsmethoden zum Einsatz. Das Open Web Application Security Project (OWASP) erstellt regelmäßig eine Liste der der wichtigsten Angriffsarten auf Internetapplikationen. Die in dieser Arbeit untersuchten Risiken Injection (Am Beispiel Structured Query Language (SQL)) und XSS[4] belegen auf dieser Liste für 2013 die Plätze eins und drei[5]. Die beiden weiteren hier besprochenen Angriffsmethoden Denial of Service (DoS) bzw. Distributed Denial of Service (DDoS), Man in the Middle und Brute Force können als zumindest als Teilsspekte der ebenfalls in der Liste vorkommenden Punkte „Broken Authentication and Session Management“ (Platz 2) und „Security Misconfiguration“ (Platz 5) betrachtet werden.

3.1 Denial of Service

Unter einem DoS versteht man einen Angriff auf die Verfügbarkeit eines Angebots. Im Kontext dieser Arbeit verfolgt der Angreifer somit das Ziel, den Webshop für die Kunden unbenutzbar zu machen. Dies kann aufgrund der öffentlichen Erreichbarkeit mit legitim erscheinenden Anfragen an den Webserver geschehen. Aufgrund der Leistungsfähigkeit moderner Webserver ist dies aber i.d.R. nicht mit einzelnen Rechnern zu leisten. Daher werden meist große Netzwerke von Rechnern[6] für einen gemeinsamen, dann DDoS gennanten Angriff verwendet.[7]

Oft sind DoS–Angriffe nicht zweifelsfrei nachweisen, da Server auch durch eine entsprechende Menge von legitimen Zugriffen[8] überlastet werden können. Mögliche Gegenmaßnahmen sind die Nutzung einer skalierbaren Infrastruktur, die

[4] X für „Cross“, um Verwechslungen mit Cascading Style Sheets (CSS) zu vermeiden.
[5] vgl. [OWASP.org (Hrsg.), 2013]
[6] Die Besitzer dieser Rechner wissen i.d.R. nichts von diesen Angriffen, da die Schadsoftware unbemerkt über Computerviren eingeschleust wurde.
[7] vgl. [Carr, 2011], Kapitel 5
[8] z.B. durch Berichterstattung in populären Medien bzw. auf reichweitenstarken Webseiten („Slashdot–Effekt“)

allgeimein effektive Implementierung der Shopsoftware sowie eine entsprechende Konfiguration der Serversoftware (z.B. Caching).

Da DoS–Angriffe nicht nur per Hypertext Tranfsfer Protocol (HTTP) auf die Shopsoftware selbst sondern auch auf Firewalls oder Router zielen können[9] sind die Abwehrmöglichkeiten für den Entwickler der Shopsoftware sehr beschränkt.

Außerdem besteht ein Zusammenhang zwischen Brute Force Angriffen, den Abwehrmaßnahmen auf ebendiese sowie eventueller Nichtverfügbarkeit des Dienstes. Näheres dazu im Abschnit 3.3 Brute Force.

3.2 Man in the Middle

Beim Man in the Middle Angriff versucht der Angreifer sich unbemerkt als Mittelsmann in die Kommunikation zwischen Kunde und Webshop einzuklinken. Obwohl der Kunde denkt, direkt mit der Anwendung zu kommunizieren, kann jeglicher Datenfluss vom Angreifer mitgelesen, protokolliert und in schlimsten Fall manipuliert werden. Der Angreifer benötigt daher Zugriff auf die Infrastruktur. Dieser kann im tatsächlichen Zwischenschalten von Abhörhardware[10] oder im virtuellen Zwischenschalten auf den unteren Protokollebenen der Netzwerkkommunikation erfolgen.[11]

Da der Applikationsentwickler keinen Einfluss auf die Infrastruktur des Internets hat, bleibt hier als Gegenmaßnahme lediglich die Wahl von verschlüsselten Kommunikaitonswegen, z.B. per Hypertext Tranfsfer Protocol Secure (HTTPS).

[9] vgl. [Amoroso, 2011], S. 60ff
[10] vgl. [Stoll, 1998], S. 28
[11] vgl. [Pritchett et al., 2013], Kapitel 7.8

3.3 Brute Force

Bei einem Brute Force Angriff setzt der Angreifer nicht wie es der Name vermuten lässt auf rohe Gewalt, sondern benötigt Geduld und Ausdauer. Durch Ausprobieren aller möglichen Kombinationen von Benutzername und Passwort versucht der Angreifer Zugriff auf die geschützten Bereiche des Webshops zu erlangen. Es werden oft Listen mit beliebten Passwortern verwendet, um die Effizienz des Angriffs zu optimieren.

Hierbei ist beachtenswert, dass durch die wiederholten Anmeldeversuche der Server überlastet werden kann. Somit kann ein nicht erfolgreicher Brute Force Angriff durchaus zu einem unbeabsichtigten, aber erfolgreichen DoS–Angriff entwickeln.[12]

Als Gegenmaßnahmen kann die Anzahl der Anmeldeversuche pro Benutzer oder von einer IP–Adresse aus beschränkt werden. Da der Angreifer jedoch meist nicht nur einen, sondern mehrere Nutzernamen durchprobiert, kann hier recht schnell für mehrere Nutzer der Zugriff gesperrt werden. Die regulären Kunden können dann den Webshop nicht erreichen. Die Auswirkungen entsprechen wieder DoS.

Der Angreifer kann durch starke Passworte aufgehalten werden. Somit sollte beim Anwendungsentwurf darauf geachtet werden, dass lange Passworte möglich sind, die sowohl Buchstaben und Zahlen als auch Sonderzeichen enthalten können. Je ungewöhnlicher ein Passwort ist, desto wahrscheinlicher ist es, dass es nicht in den Passwortlisten der Angreifer auftaucht, was einen Angriff langsamer, teurer und somit unattraktiver macht. Allerdings sind starke Passworte nicht benutzerfreundlich. Kryptographisch ausreichende Passworte benötigen 20 bis 32 Zeichen und sind somit im Alltag nicht gebräuchlich.[13]

Somit ist eine Erweiterung des Passwortbegriffs notwendig, um Brute Force Attacken wirksam bekämpfen zu können. Die Benutzerauthentifikation kann auf zwei oder mehr Stufen erweitert werden. Hierzu wird bei der Benutzeranmeldung nicht

[12] vgl. [Bartl (Hrsg.), 2014]
[13] vgl. [Kaufman et al., 2002], Kapitel 10.4

nur das abgefragt, was der Nutzer *weiß* (also ein Passwort oder eine Passphrase), sondern auch etwas, was er *besitzt*, was er *ist*[14] oder was er *kann*.

- **Können:** Eine Abwehrmethode sind CAPTCHAs[15], also für Menschen einfache Aufgaben, die für Computerprogramme sehr schwer zu lösen sind.[16] CAPTCHAs sind schon weit verbreitet, sind aber oft nicht benutzerfreundlich.
- **Besitzen:** Schlüssel, Magnetkarten oder Transponder können zur Benutzeranmeldung eingesetzt werden. Desweiteren besteht die Möglichkeit, dass ein Codegenerator[17] einen Code erzeugt, der nur eine bestimmte, kurze Zeit gültig ist. Somit sind ausgespäte Codes für einen Angreifer wertlos. Nur der Besitzer des Generators kann diese Zeitnah erzeugen und sich damit authentifizieren.
- **Sein:** Über biometrische Merkmale wird die Identität des Benutzers festgestellt. Hierzu können unter anderem Fingerabdruckscanner, Irisscanner, Gesichtserkennung oder auch das Timing beim Unterschreiben genutzt werden.

Auch wenn jede dieser Methoden Nachteile hat und für sich alleine genommen nicht zum Aufbau eines Authentifizierungssystems ausreicht, so können diese Methoden zu einer sogenannten Anmeldung in zwei oder mehr Schritten kombiniert werden, und somit eine wirkungsvolle Maßnahme gegen Brute Force Attacken sowie einer Reihe anderer Risiken rund um Passworte darstellen.

[14] vgl. [Kaufman et al., 2002], Kapitel 10
[15] Completely Automated Public Tourint Test to Tell Computers and Humans Apart (CAPTCHA)
[16] Diese können aus umgangssprachlich Formulierten Rechenaufgaben auf Erstklässlerniveau oder aus dem Erkennen einer Zeichenfolge in einer Grafik mit Hintergrundrauschen bestehen.
[17] z.B. Google–Authenticator als Smartphone–App oder spezielle Hardwaregeneratoren

3.4 SQL–Injection

Die Daten moderner Internet–Applikationen werden meist in Datenbanken gespeichert, auf die mithilfe von SQL zugegriffen wird. Die Parameter für die Datenbankbefehle werden oft vom Nutzer in HTML–Formularen eingegeben. Hier ergibt sich eine Möglichkeit, Schadcode in die generierte SQL–Anweisung einzubringen.

Eine einfach gehaltene Nutzerverwaltung könnte Nutzername (z.B. „john“) und Passwort (z.B. „geheim“) einlesen. Der vom Formular aufgerufene PHP–Code (siehe Abbildung 1) erzeugt daraus die in Abbildung 2 gezeigte SQL–Abfrage.

```
$query = "SELECT id FROM Users ".
         "WHERE user = '$_GET["user"]' ".
         "AND password = '$_GET["password"]'".
```

Abbildung 1: Quellcode: Dynamische Erzeugung einer SQL–Abfrage

```
SELECT id FROM Users WHERE user = 'john'
   AND password = 'geheim';
```

Abbildung 2: SQL: Erwartete Abfrage

Problematisch wird es, wenn ein Angreifer diese durchaus übliche Funktionsweise errät. Dann kann er über gezielt gestaltete Eingabedaten die SQL–Abfrage so manipulieren, dass er auch ohne korrekte Anmeldedaten Zugriff erlangen kann. Hierzu muss nur als Passwort die Zeichenkette `keines'OR '1'='1` eingegeben werden. Nun entsteht die in Abbildung 3 gezeigte Abfrage, welche aufgrund der immer wahren Bedingung `'1'='1'` eine Liste mit allen Benutzer–IDs zurückgibt. Das Programm schließt dann später aus der nicht leeren Rückgabeliste, dass die Anmeldung rechtens ist, und gewährt dem Angreifer Zugriff.[18]

```
SELECT id FROM Users WHERE user = 'john'
   AND password = 'keines' OR '1'='1';
```

Abbildung 3: SQL: Manipulierte Abfrage

[18] vgl. [Clarke, 2012], Kapitel 1.3

Je nach genutztem Datenbanksystem sind nach diesem Schema eine Vielzahl von Angriffen auf die Daten selbst, oder auch auf die Betriebssystemebene des Datenbankservers möglich.

Um eine Web–Applikation gegen SQL–Injection Angriffe zu sichern, dürfen keine Eingaben aus Formularfeldern ungeprüft in SQL–Anweisungen übernommen werden. Sämtliche Zeichen, die im SQL–Syntax eine Bedeutung haben müssen herausgefiltert oder so kodiert werden, dass diese Bedeutung nicht mehr vorhanden ist[19] Hierbei ist jedoch zu beachten, dass wirklich alle relevanten Zeichen berücksichtigt werden und dass das Escaping wirklich jedes mal für alle Daten durchgeführt wird.[20] Um hier Fehler zu vermeiden, empfielt sich der Einsatz entsprechender Module für den Zugriff auf SQL–Datenbanken, die selbst und automatisch für das Escaping sorgen. Als Beispiel seien hier die PHP Data Objects (PDO) genannt.

Einen Schritt weiter geht die Methode, dem SQL–Server zunächst die SQL–Anweisungen mit entsprechenden Platzhaltern für die Parameter zu übergeben. Der SQL–Server bereitet nun die Abfrage vor und optimiert diese. Erst danach werden die eigentlichen Werte der Parameter übergeben und die Abfrage ausgeführt.[21] Hierduch ist dem Server unabhängig von der Zeichenkette der Abfrage bekannt, was zu seinen Anweisungen gehört und was die zu verarbeitenden Daten sind. SQL–Injection Angriffe sind auf diese Weise wirkungsvoll zu verhindern.[22]

3.5 Cross Site Scripting

Bei einem XSS–Angriff handelt es sich ebenfalls um eine Technik, bei der Schadcode in die Anwendung eingebracht wird. Im Gegensatz zur SQL–Injection wird

[19] Zumeist durch voranstellen eines Backslashes „\“. Dieser Vorgang wird „Escaping“ genannt.
[20] z.B. durch den Aufruf der PHP–Funktionen magic_quotes(), add_slashes() oder mysql_real_escape_string()
[21] Die Schritte „Parameterübergabe“ und „Ausführung“ können in einer Schleife mehrfach hintereinander mit unterschiedlichen Daten ausgeführt werden. Da die Vorbereitung und Optimierung nur ein mal (vor der Schleife) durchgeführt wird, ergibt sich hier auch ein Geschwindigkeitsvoreil.
[22] vgl. [Clarke, 2012], Kapitel 8.3

dieser jedoch nicht auf dem Server, sondern im Browser des Anwenders aktiv. Voraussetzung ist, dass der Webserver die an den Browser gesendete Seite dynamisch erzeugt und hierbei Nutzereingaben verarbeitet werden. Sollte es dem Angreifer gelingen dass hier schadhafte Inhalte, z.B. in Form von JavaScript–Code eingebracht werden können, haben weder Server noch Browser[23] genügend Informationen dies zu erkennen und sich gegen den Angriff zu schützen.[24]

Gefährdet sind also alle Websites, die Benutzereinganben entgegennehmen und ohne entsprechende Prüfungen wieder zurückgeben. Dies kann z.B. über ein Kommentar– oder Gästebuchfeld des Webshops der Fall sein. Der Angreifer übergibt dem Kommentarfeld seinen JavaScript–Code. Der Administrator bekommt eine Nachricht, dass ein neuer Kommentar vorliegt und lässt sich diesen anzeigen. Schon bei der Anzeige, wird dann der Schadcode im Browser des Administrators mit dessen Zugangsberechtigungen ausgeführt. Das Script hat auch Zugriff auf die HTML–Cookies der angegriffenen Website, also auch auf das Session–Cookie, welches dem Administrator umfangreichen Zugriff auf den Webshop gewährt. Solange diese Session gültig ist[25], kann der Angreifer die Privilegien des angegriffenen Administrators übernehmen.

Die Gegenmaßnahmen, mit denen man sich vor einem XSS–Angriff schützen kann, sind direkt aus der Beschreibung der Bedrohung: Über HTML–Formulare eingegebene Benutzerdaten müssen so gefiltert werden, dass Zeichen mit spezieller Bedeutung in Hypertext Markup Language (HTML) bzw. JavaScript nicht unverändert in dynamisch generierte Webseiten einfließen können. Besonders hervorzuheben sind die die Bregrenzer für HTML–Tags „<“ bzw. „>“ sowie einfache und doppelte Hochkomma, das Prozentzeichen „%“ und das „kaufmännische–Und“ „&“. Diese sind durch die jeweiligen HTML–Entities[26] zu ersetzen.

[23] Also auch weder Shopbetreiber noch Kunde
[24] vgl. [Ross et al., 2000], Abschnitt 1
[25] Sessions können durch Ablauf einer vom Entwickler bestimten Zeitspanne oder durch ausdrückliches Abmelden ungültig werden.
[26] Also < bzw. > usw.

Man kann auch für Formularfelder komplett die Eingabe von HTML–Code verbieten, also alle HTML–Tags herausfiltern. Dies ist im betrachteten Fall eines Webshops ohne Einschränkungen für den Nutzer möglich. Sollten doch Textformatierungen erwünscht sein, so kann wie dies durch die Nutzung von entsprechendem Markup–Syntax wie „Markdown“[27] oder „BBCode“[28] erlaubt werden.[29]

3.6 Weitere Schutzmaßnahmen

Als generelle Schutzmaßnahmen bestehen eine Reihe von Techniken, die zwar keinen spezifische Angriffsart verhindern, aber bei einem erfolgreichen Angriff dessen Schaden begrenzen können. Hierzu zählen unter anderem:

- Die gleichzeitige Nutzung des Webshops mit identischen Anmeldedaten aber unterschiedlichen IP—Adressen sollte unterbinden werden.
- Der Administrators darf sich nur von bestimmten IP–Adressen (Innerhald bes Firmennetzwerks) einloggen.
- Bei der Anmeldung wird Zeit und Ort der letzten Anmeldung angezeigt.
- Regelmäßige Backups
- Benutzerpasswörter verschlüsselt speichern.
- Plausibilitätsprüfungen, z.B. auf auffällige Bestellmengen, plötzlich häufige Bestellungen oder Lieferung an „fremde“ Adressen.

[27] siehe http://markdown.de/
[28] siehe http://www.bbcode.org/
[29] Hierbei werden die jeweiligen Formatierungsbefehle in die entsprechenden HTML–Auszeichnungen umgewandelt. Somit hat die Applikation trotz dynamischer Generierung der Webseiten doch die volle Kontrolle über den ausgelieferten HTML–Code.

4 Fazit & Ausblick

4.1 Fazit

Bei Betrachtung der Angriffsmöglichkeiten und der Risiken wird offensichtlich, dass es nur mit beachtlichem Aufwand möglich ist, einen Webshop vollständig vor Angreifern zu schützen. Speziell aufgrund der Tatsache, dass hier nur eine Auswahl der Bedrohungen besprochen wurde.

Andererseits ist auch zu erkennen, dass viele Angriffsszenarien durch die Auswahl entsprechender Komponenten und Best–Practices wirkungsvoll abgewehrt werden können. Wenn Sicherheitsaspekte schon von Beginn an in den Softwareentwurf eingehen, ist die resultierende Applikation zumindest vor den meisten Angriffen geschüzt. Zumindest zufälligen Angreifern, die im Internet nach offensichtlichen Sicherheitslücken suchen, wird somit kein Ziel geboten.

4.2 Ausblick

Auf dem Weg zu einer tatsächlichen Implementierung ist es notwendig, weitere Angriffsmethoden zu betrachten, damit das fertige Produkt zumindest gegen die von der OWASP erstellten Top–10 Liste bestehen kann. Die jeweils notwendigen Gegenmaßnahmen sollten als Richtlinien für die Codierung festgelegt und überprüft werden.

Falls der Internetshop um eine Bezahlfunktion erweitert werden soll, sind noch weitere Angriffsziele zu beachten, da in diesem Fall nicht nur Daten, sondern auch Geld[30] sowie Waren[31] erbeutet werden können. Dieses Szenario würde aber den Umfang dieser Arbeit überschreiten und müsste somit seperat erörtert werden.

[30] Das Geld unserer Kunden
[31] Die angebotenen Waren ohne Bezahlung

Für weitere Untersuchungen würde sich die Fragestellungen anbieten, in wie weit die am Markt erhältliche Shopsoftware gegen Angriffe geschützt ist bzw. wie groß der Entwicklungsaufwand ist, den am Markt üblichen Grad an Sicherheit bei einer selbst erstellten Anwendung zu erreichen. Hierdurch könnte eine Make-or-Buy Entscheidung unterstützt bzw. begründet werden.

Literatur

[Amoroso, 2011] Amoroso (2011). *Cyber Attacks – Protecting National Infrastructure*. Butterworth-Heinemann, Oxford, UK.

[Bartl (Hrsg.), 2014] Bartl (Hrsg.) (2014). Brute Force Attack erfolgreich abwehren und blocken, Abruf am 01.05.2014. `http://www.defense.at/securitypaper/brute-force-attack.html`.

[Carr, 2011] Carr (2011). *Inside Cyber Warfare*. O'Reilly Media, Sebastopol, CA, USA, 2nd. edition.

[Clarke, 2012] Clarke (2012). *SQL Injection Attacks and Defense*. Syngress, New York, NY, USA, 2nd edition.

[Dix (Hrsg.), 2008] Dix (Hrsg.) (2008). Begriffsbestimmung: Verfügbarkeit, Integrität, Vertraulichkeit, Authentizität, Abruf am 01.05.2014. `http://www.datenschutz-berlin.de/content/technik/begriffsbestimmungen/verfuegbarkeit-integritaet-vertraulichkeit-authentizitaet`.

[Fuest, 2013] Fuest (2013). Fall Vodafone zeigt die wahren Sicherheitslücken, Abruf am 01.05.2014. `http://www.welt.de/wirtschaft/webwelt/article119967954/Fall-Vodafone-zeigt-die-wahren-Sicherheitsluecken.html`.

[Kaufman et al., 2002] Kaufman et al. (2002). *Network Security: Private Communication in a Public World*. Pretice Hall, Upper Saddle River, NJ, USA, 2nd edition.

[OWASP.org (Hrsg.), 2013] OWASP.org (Hrsg.) (2013). Top 10 – 2013, The Most Critical Web Application Security Risks, Abruf am 01.05.2014. `http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf`.

[Pritchett et al., 2013] Pritchett et al. (2013). *Kali Linux Cookbook*. Packt Publishing, Birmingham, UK.

[Ross et al., 2000] Ross et al. (2000). Cross-site Scripting Overview, Abruf am 02.05.2014. `http://ha.ckers.org/cross-site-scripting.html`.

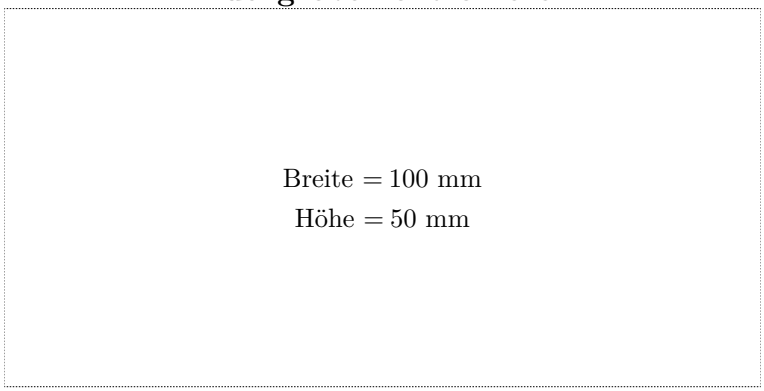
[Stoll, 1998] Stoll (1998). *Das Kuckucksei*. Fischer Taschenbuch Verlag, Frankfurt am Main.

Eidesstattliche Erklärung

Ich versichere, dass ich das beiliegende Assignment selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.

(Datum, Ort) (Unterschrift)

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —