Designrichtlinien für Webanwendungen

Secure Web Application Engineering

David Jaeger

Einleitung

Beobachtet man Aktivitäten im Internet, so lässt sich feststellen, dass in der heutigen Zeit die Bedrohungen durch Internetkriminalität immer weiter zunehmen. Dabei werden häufig Webserver zum Angriffsziel. Möglich werden diese Angriffe nur durch Sicherheitslücken in der Software oder in den Webseiten, die auf dem betreffenden Server laufen.
Um Sicherheitslücken vorzubeugen, ist es nötig alle möglichen Risiken zu identifizieren. Häufig bedient man sich dabei dem so genannten „Threat Modelling“. Threat Modelling bezeichnet den Vorgang des Herausstellens und das anschließende Dokumentieren von Bedrohungen und Schwachstellen zum Zweck der Reduzierung der Angriffswahrscheinlichkeit.**[1]** Hat man erst ein solches Modell vorzuliegen, kann mit der Entwicklung des geplanten Softwaresystems begonnen werden. Die Entwickler kennen somit die Schwachstellen und können diese gezielt vermeiden.
Da viele Fragestellungen bei der „Threat Model“-basierten Entwicklung immer wieder auftreten, haben sich bestimmte Designrichtlinien etabliert, die man als Webdesigner einhalten sollte und die einem das Entwickeln erleichtern. Diese Richtlinien sollen in dieser Arbeit vorgestellt werden.
Die Themen die dabei aufgegriffen werden können, richten sich genau nach den Prüfungskriterien für die Erstellung eines Threat Models, die z.B. im „Web Application Security Frame“ aufgeführt sind.
Aus diesem Schema sollen besonders die Punkte *Validation von (Eingabe-)Daten*, *vertrauenswürdige Daten*, *Parametermanipulation* und *Ausnahmebehandlung* näher erläutert werden.**[2]** Die *Validation von Eingabedaten* und die *Ausnahmebehandlung* wird in eigenständigen Abschnitten dieser Arbeit vorgestellt. Die Sicherung vor Zugriffen, also die Vergabe von Rechten, wird in einem weiteren Abschnitt dargestellt, ebenso wie die Darstellung des Entwicklungsprozesses einer Webanwendung (Deployment).

Eingabeüberprüfung

Eingabedaten

Betrachtet man die heute gängigen Technologien im Internet, so kommt man auf die wohl bekannteste Eingabeart, der HTTP-Header mit seinem GET/POST und den Cookies. Des Weiteren werden durch die Popularität des XML-Formats auch immer mehr Eingabedaten über SOAP ausgetauscht. Eine eher marginale Quelle von Eingabedaten sind externe Server. Die einzelnen Eingabearten sollen im Folgenden genauer betrachtet werden.

- **HTTP-Header**: HTTP-Header beinhalten GET/POST und Cookies, bieten jedoch auch andere Arten von Eingabedaten. Wie in Abbildung 1 zu erkennen ist, gibt es auch noch weitere Felder mit Metadaten, wie z.B. die Browserbezeichnung oder die aufrufende Seite (Referer). Obwohl man diese beiden Felder nicht direkt manipulieren kann, so bieten doch manche Browser die Möglichkeit zur Umkonfiguration dieser Daten.
- **GET/POST:** GET und POST sind häufig genutzte Methoden zum Erfragen einer Ressource auf einem Server mittels HTTP. Abbildung 1 zeigt eine typische GET-Anfrage.
- **Cookies:** Der Server überträgt Cookies zum Client, der diese auf Anfrage unverändert zurücksendet. Durch Cookies können Daten über einen Webseitenbesuch hinaus gesichert und auch als Authentifizierungstokens zum Anmelden in einem internen Bereich benutzt werden. Innerhalb des HTTP-Headers ist ein eigener Bereich für Cookies zu finden, der mit dem Label „Cookie:“ eingeleitet wird.
- **XML/SOAP:** Mit SOAP bezeichnet man ein Protokoll zur Übertragung von XML-basierten Daten. Es wird zur Übertragung meistens in HTTP- oder HTTPS-Pakete gekapselt. Das Prinzip von SOAP erinnert an *Remote Procedure Calls*. Damit können also Funktionen auf entfernten Servern aufgerufen werden, die das Ergebnis einer Anfrage ermitteln und dann als XML-kodierte Daten an den Client zurücksenden.**[3]**
- **Externe Server:** Viele Webseiten sind auf externe Ressourcen angewiesen. So werden z.B. SQL-Anfragen an Datenbanken häufig an entfernte Server gestellt. Das Ergebnis wird dann an die Webseite zurückgeschickt, es liegen also Eingabedaten vor. Des Weiteren könnte man sich eine Online-Börse vorstellen, die ihre Kursdaten aus anderen Börsenseiten parst.

```
⊟ Hypertext Transfer Protocol
  ⊞ GET /search?hl=de&q=HPI&btnG=Google-Suche&meta= HTTP/1.1\r\n
    Host: www.google.de\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.9) Gecko/20071025 Firefox/2.0.0.9\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.google.de/\r\n
    Cookie: PREF=ID=ad4cf5d04891d3c8:TM=1195911816:LM=1195911816:S=CfdY1cn2dZ8BKmNm\r\n
    \r\n
```

Abbildung 1: *HTTP-Header einer GET-Anfrage an Google mit dem Suchwort „HPI“*

Angriffsvektoren

Mit den vorgestellten Eingabedaten bekommt ein Angreifer Mittel in die Hand ein Softwaresystem anzugreifen. Da es meist nur sehr schwer möglich ist alle Eingabedaten zu filtern, können Sicherheitslücken entstehen, die sich kritisch auf das System auswirken. Nachfolgend werden Angriffsvektoren vorgestellt, deren Entstehung durch unzureichende Eingabeüberprüfung ermöglicht wird.**[4]**

Der wohl trivialste Angriffsvektor ist die **Parametermanipulation** in GET- und POST-Anfragen. Hier wird die Anfrage, die z.B. in der Adressleiste eines Browser zu finden ist, verändert. Parametermanipulation ist ein sehr weit gefasster Begriff, der als Obermenge für weitere Angriffsvektoren gilt, wie z.B. die SQL-Injektion und das Cross-Site-Scripting.
Die **SQL-Injektion** ermöglicht das Ausführen von manipulierten SQL-Anweisungen auf eine Datenbank. In ungünstigen Fällen können so genannte *Stored Procedures* aufgerufen werden, die das Ausführen beliebiger Befehle auf der Kommandozeile ermöglichen.
Beim **Cross-Site-Scripting (XSS)** wird HTML- oder Scriptcode in einem Eingabedatum eingebaut, der dann auf der resultierenden Seite ausgeführt wird. Meistens ist es dafür nötig, dem Opfer einen manipulierten URL unterzuschieben, der dann für das Ausführen des eingeschleusten Codes sorgt (nicht persistentes XSS). Gelingt es einem Angreifer den Schadcode z.B. in eine Datenbank einzuschleusen, so wird dieser dauerhaft in eine Webseite integriert (persistentes XSS).
Der letzte Angriffsvektor der hier vorgestellt werden soll, ist der **Buffer Overflow** oder auch Pufferüberlauf. Dieser kann auch zu den Parametermanipulationen gezählt werden. Beim Buffer Overflow wird eine zu geringe Größe eines Eingabepuffers ausgenutzt. Durch das Beschreiben von Speicher über Puffergrenzen hinaus, kann es zur Überschreibung von Funktionsrücksprungadressen kommen. Damit kann man die Rücksprungadresse so verändern, dass das Ausführen beliebigen Codes ermöglicht wird. Interpretierte Sprachen sind relativ immun gegen diese Art von Angriffen, da hier meistens dynamische Puffergrößen verwendet werden. Sobald jedoch auf Bibliotheken zugegriffen wird, die vielleicht in C/C++ implementiert wurden, besteht wieder die Gefahr eines derartigen Angriffs.

Mögliche Angriffsziele

Mit den oben beschriebenen Methoden ist es Angreifern möglich viel Schaden anzurichten. Im Folgenden sind die Angriffsziele auf die Eingabeüberprüfung und die dazugehörigen Angriffsvektoren aufgelistet.**[5]**

- **Code Injektion:** Einschleusen von Code auf einen Server bzw. in eine Webseite (SQL-Injektion, XSS, Buffer Overflow)
- **Denial of Service:** Server kann seinen Dienst nicht weiter fortführen (Buffer Overflow)
- **Sitzungsübernahme:** Nutzen einer Sitzung eines anderen Benutzers (Parametermanipulation)
- **Phishing:** Nutzerdaten werden auf gefälschten Seiten ausspioniert (XSS)
- **Informationsoffenlegung:** Anzeigen von Nutzer- oder Geschäftsdaten, die nicht für jemanden bestimmt sind (Parametermanipulation, SQL-Injektion, Buffer Overflow)

Angriffsvorbeugung

Grundsätzlich lassen sich Eingaben nach verschiedenen Kriterien prüfen. So können die **Typen der übergebenen Daten** geprüft werden, z.B. möchte man sicher gehen, dass bei einer erwarteten Ganzzahl kein String übergeben wurde. Des Weiteren kann die **Eingabelänge** eines Strings überprüft werden, um Buffer Overflows zu reduzieren. Erwartet man Daten in einem bestimmten Format, wie eine E-Mail-Adresse, kann das **Datenformat** z.B. mit regulären Ausdrücken auf Gültigkeit überprüft werden. Um Pufferüberläufe durch zu große Zahlenbereiche einzuschränken kann man **Bereichsüberprüfungen** von Zahlen vornehmen. Damit könnten z.B. Zahlen gefiltert werden, die über den Bereich eines Integers hinausgehen.
Im nächsten Abschnitt sollen Hilfsmittel beschrieben werden, um die genannten Prüfungen durchzuführen können.**[5]**

Hilfsmittel zur Überprüfung von Eingabedaten

Grundsätzlich gibt es verschiedene Klassen von Hilfsmitteln. Einige sind direkt in der Programmiersprache eingebaut, andere werden durch die Laufzeitumgebung bereitgestellt und wieder andere sind Bestandteil des benutzten Webservers. Das wohl bekannteste und leistungsstärkste Mittel auf der programmiertechnischen Ebene ist der reguläre Ausdruck. Mit ihm können Eingaben auf beliebige Muster getestet werden. Scriptsprachen bieten zumeist spezielle Methoden zum *Escapen* von Zeichenketten an, also die Umwandlung von Steuerzeichen in normale Ausgaben. Die Laufzeitumgebung von ASP.NET bietet bereits ein automatisches Escapen sämtlicher Eingabedaten. Als Webservererweiterung für die Internet Information Services (IIS) kann auch URLScan benutzt werden.**[5]** Es sorgt dafür, dass alle Eingaben über HTTP/HTTPS *Escaped* bzw. gefiltert werden.

Ausnahmebehandlung (Exception Handling)

Wenn die Prüfung der Eingabe durchgeführt und fehlerhafte oder gefährliche Ausdrücke entdeckt wurden, kann mit dem Fehlerreport begonnen werden. Der Nutzer soll auf seine fehlerhafte Eingabe aufmerksam gemacht werden. Dabei sind bestimmte Dinge zu beachten. Näheres soll in diesem Abschnitt beschrieben werden.

Begriffsdefinition

Ausnahmen werden für gewöhnlich bei unerwartetem Verhalten geworfen. Durch sie können Inkonsistenzen vermieden werden, die durch unerwartete Programmabläufe entstehen könnten. Des Weiteren bekommt ein Nutzer Informationen zu einem fehlerhaften Vorgang angezeigt und kann dementsprechend handeln. Ausnahmen stellen auch ein wichtiges Mittel in der Softwareentwicklung dar. Sie sind sehr gut für Fehlererkennung und –beseitigung (Debugging) geeignet. Üblicherweise werden Ausnahmen in Programmiersprachen mit *try/catch*- oder *try/except*-Kontrukten abgefangen.

Arten von Ausnahmen

Grundsätzlich unterscheidet man systembedingte und nutzerbedingte Ausnahmen.**[6]**

- **Systembedingte Ausnahmen**: Sie treten dann auf, wenn z.B. Ressourcen des Servers aufgebraucht sind, Verbindungsprobleme vorliegen, nicht genügend Rechte für einen Zugriff vorliegen oder wenn der Interpreter einen Syntaxfehler meldet.
- **Nutzerbedingte Ausnahmen:** Sie können auftreten, wenn der Benutzer eine ungültige Eingabe vornimmt oder wenn er versucht auf nicht mehr gültige Teile einer Webseite zuzugreifen, sei es weil die Sitzung abgelaufen ist.

Fehlerreport

Wenn die Ausnahme abgefangen wurde, wird sie für gewöhnlich dem Benutzer gezeigt, damit dieser über ein fehlerhaftes Verhalten informiert wird. Eine wohl bekannte Methode ist die Weiterleitung auf eine Fehlerseite, wie z.B. „Fehler 404 – Seite wurde nicht gefunden“. Der Nachteil daran ist, dass der neue Seitenaufruf den Kontext des Fehlers zerstört. Damit ist schlecht nachzuvollziehen, was genau den Fehler ausgelöst hat.
Eine Methode, bei der auch der Kontext erhalten bleibt, wäre die direkte Anzeige einer Warnung oder eines Fehlers auf dem Seitenteil, wo die Ausnahme aufgetreten ist. Das schließt auch den Standardfehlerreport sämtlicher Laufzeitumgebungen ein, der jedoch möglichst im Live-System vermieden werden sollte.
Manchmal ist ein Fehlerreport an den Nutzer nicht erwünscht, da dadurch sensible Informationen preisgegeben werden könnten oder einem Angreifer soll nicht gezeigt werden, dass seine Aktionen registriert werden. Für diese Zwecke können Ausnahmen auch nur in eine Logdatei auf dem Server gespeichert werden, sodass die Meldungen nur für Administratoren zugänglich sind.

Was sollte man bei der Ausnahmebehandlung beachten?

Beim Anzeigen von Ausnahmen können viele Dinge zur Bedrohung werden. Man könnte zu viele Informationen preisgeben, die gegebenenfalls einem Angreifer behilflich sein könnten oder es werden zu wenig Informationen gegeben, sodass der Nutzer nichts mit dem Fehler anzufangen weiß. Daher sollte man Fehlermeldungen für Anwender gut verständlich formulieren und nicht nur kryptischen Fehlercodes verwenden.**[7]** Fehlercodes könnten für Debugging- oder Reportzwecke benutzt werden. Wenn kritische Situationen im Programm entstehen, sollten Ausnahmen geworfen werden, um die Konsistenz des Programms zu wahren. Bei der Ausgabe von Fehlern sollten keine sensiblen Daten frei werden, wie z.B. Stack-Traces oder SQL-Anfragen. Ausnahmen selbst sollten hierarchisch aufgebaut werden. Meistens bietet es sich an, Ausnahmen an einem zentralen Punkt zu behandeln bzw. nach oben zu reichen, um dort von detaillierten Fehlermeldungen abstrahieren zu können. Generell sollte Ausnahmebehandlung an allen möglichen fehlererzeugenden Stellen im Quellcode durchgeführt werden.**[8]**

Berechtigungen

Nachdem beschrieben wurde, wie man sich vor manipulierten Eingabedaten schützt, wird nachfolgend gezeigt, wie man als weiteren Schutz Dateisystem und Server vor unberechtigtem Zugriff schützen kann. Normalerweise greifen Webservices auf das Dateisystem des Servers zu. Damit nicht uneingeschränkter Zugriff möglich wird, müssen die Ressourcen des Servers geschützt werden. Die wohl wertvollste Ressource wäre das Dateisystem, in dem viele sensible Daten abgelegt sein könnten. Man benötigt also Mittel, um Ressourcen durch Berechtigungen zu schützen.
Das Vergeben von Rechten kann auf zweierlei Arten realisiert werden, über das Dateisystem oder durch den Webserver, der einen Dienst anbietet.

Auswirkungen falsch gesetzter Berechtigungen

Durch Berechtigungen lassen sich viele Ressourcen durch ungewollten Zugriff schützen, doch was passiert, wenn diese unvorteilhaft gesetzt werden? Dieser Abschnitt soll einzelne Szenarios des ungewünschten Zugriffs darstellen. Schränkt man den Zugriff z.B. nicht auf das Wurzelverzeichnis des HTTP-Webservers ein, so ist ein Zugriff auf das gesamte Dateisystem möglich. Es könnten nun wertvolle Kundendaten oder die /etc/passwd oder /etc/shadow gestohlen werden. Sind zudem Schreibrechte gegeben, kann Schadcode in das System eingeführt werden und die Webseite lässt sich nach belieben ändern. Sind auch Leserechte für Scripte gegeben, lassen sich Quelltexte auslesen, die eventuell hartkodierte Passwörter zur Datenbank enthalten. Sind nun die Rechte der Datenbank auch unzureichend festgelegt, können Tabellen vollständig ausgelesen oder gelöscht werden. Die Dateiberechtigungen sollen nun genauer betrachtet werden.

Arten von Dateiberechtigungen

Generell gibt es zwei Wege die Berechtigungen für bestimmte Dateien zu setzen. Der erste Weg ist die Berechtigung, die durch das Dateisystem gegeben wird. Gerade Nutzern von UNIX-basierten Systemen sollte dieses Prinzip bekannt sein. Eine weitere Methode ist die Einschränkung der Rechte durch den Webserver.

Dateisystemberechtigungen

Dateisystemberechtigungen regeln den physischen Zugriff auf eine Datei im Dateisystem. Für das Verwalten von Berechtigungen ist das NTFS-Dateisystem unter Windows von den üblichen UNIX-Dateisystemen (ext3, ReiserFS) abzugrenzen.

Unter Windows verwendet man für gewöhnlich so genannte *Access Control Lists* (*ACL*). ACLs bieten die Möglichkeit Berechtigungen von Objekten für einzelne Personen, aber auch für Personengruppen festzulegen. Die typischen Attribute, wie Lese-, Schreib- und Ausführungsrechte, können in *Access Control Entries* (*ACE*) innerhalb der ACLs für ein Objekt festgelegt werden.

Die typischen UNIX-Rechte, wie sie in ReiserFS oder ext3 verwendet werden, bieten nicht so flexible Berechtigungen wie die ACLs unter Windows[1]. Eine Datei hat hier lediglich einen Besitzer und eine Gruppe. Berechtigungen können nun für den Besitzer, die Gruppe und alle anderen Personen gesetzt werden. Auch hier sind wieder die üblichen Attribute wie Lese-, Schreib- und Ausführungsrechte setzbar.**[9]**

Webserverberechtigungen

Webserverberechtigungen regeln den Zugriff für alle Nutzer eines Webservers. Sie dienen dazu, die Rechte des Dateisystems weiter zu verschärfen. Die Rechte des Dateisystems lassen sich jedoch nicht heruntersetzen, sondern dienen eher als Basis für weitere Rechte. Der Webserver greift üblicherweise mit den Rechten seines Benutzerkontos auf das Dateisystem zu.**[9]** Zumeist wird für jede verwaltete Webseite ein eigenes Benutzerkonto verwendet.**[10]** Beispiele für zwei HTTP-Server mit einer Vergabe von Berechtigungen ist der *HTTP Server* von *Apache* und die *Internet Information Services* (*IIS*) von *Microsoft*.

Was sollte man bei der Vergabe von Berechtigungen beachten?

Für die Vergabe von Berechtigungen sollen hier noch einmal die wichtigsten Aspekte zusammengefasst werden, die man für eine sichere Konfiguration benötigt.**[11]**

- Zugriffsbereiche und Zugriffsberechtigungen sollten auf das Wesentliche beschränkt werden (z.B. nur das Stammverzeichnis htdocs oder wwwroot)
- Benutzerkonten von Webservern sollten möglichst wenig Rechte bekommen
- Verwendung von Sicherheitsvorlagen
- Vergabe der Rechte möglichst an Gruppen anstatt an Benutzern
- Rechte sollten so hoch wie möglich in der Dateistruktur gesetzt werden
- Statische Dokumente, wie HTML-Dokumente, sollten nur lesbar sein
- Schreibberechtigungen sollten für sämtliche Dateien vermieden werden, insbesondere für ausführbare Dateien
- Ausführungsrechte für ausführbare Dateien nur wenn nötig
- Code Access (z.B. in IIS) sollte ausgeschaltet werden

Deployment von Webanwendungen

Nachdem bisher nur beschrieben wurde was bei der Implementierung von Webservices zu beachten ist bzw. wie Dateien auf dem Server geschützt werden können, soll nun das Deployment von Webanwendungen betrachtet werden, also wie der Entwicklungsprozess bei Webanwendungen durchgeführt wird. Dieser Punkt hängt nahe mit dem Software-Life-Cycle zusammen.

[1] ReiserFS uns ext3 bieten auch *Access Control Lists*, es wird jedoch nur selten verwendet

Begriffsdefinition

„Software deployment is all of the activities that make a software system available for use" **[12]**

Software deployment bezeichnet also die Instandsetzung und das Aufsetzen eines Softwaresystems. Es wird meistens in mehrere Schritte unterteilt, die bis zur kompletten Aufsetzung des Systems führen. Dazu gehören Release, Installation, Aktivierung, Adaption, Update und Erstellung. Diese Schritte werden zumeist vom Softwarehersteller, aber auch vom Kunden durchgeführt. Der genaue Prozess hängt jedoch vom konkreten Softwareprojekt ab.**[12]**

Deployment in der Praxis

Deployment kann auf verschiedene Arten in der Praxis durchgeführt werden. Die wohl einfachste, aber auch umständlichste Methode ist die direkte Manipulation am Live-System. Dieses Vorgehen mag für interpretierte Sprachen gut funktionieren, jedoch muss der Webserver bei kompilierten Sprachen heruntergefahren werden. Eine etwas elegantere Methode ist das Kopieren des Live-Systems auf einen Mirror-Server, an dem nun Änderungen vorgenommen werden. Das Kopieren der Dateien zwischen den beiden Umgebungen kann durch Scripte gewährleistet werden. Die wohl komfortabelste Variante ist die Nutzung von eingebauten Deployments-Tools in der Entwicklungsumgebung. Hier bietet sich z.B. das *Microsoft Visual Studio 2005 Web Deployment Project* oder *Capistrano* für *Ruby on Rails* an. Grundsätzlich sollte man sich für das Deployment von größeren Projekten an folgende Punkte halten:**[13]**

- Quellcodes sollten in Subversion-Repositories abgelegt werden
- Bereitstellen von automatisierten Fehlertests
- Benutzung einer Staging Environment
- Automatisierung durch Sripte oder eingebaute Funktionen in Entwicklungsumgebungen
- Fehler sollten so schnell wie möglich behoben werden
- Durchführung regelmäßiger Backups

Gefahren durch falsches Deployment

Beim Deployment sollte man einige Dinge beachten, insbesondere wenn man mit Tools zur Automatisierung arbeitet. Grundsätzlich sollten keine Scripte im Webverzeichnis liegen, da damit das Auslesen von Quellcodes ermöglicht werden könnte. Wenn möglich sollten also Scripte vorkompiliert werden. Beim automatisierten Deployment werden teilweise Logdateien erzeugt, die empfindliche Informationen über den Server preisgeben könnten. Es sollte daher genau darauf geachtet werden, welche Dateien auf das Live-System kopiert werden.

Umgebungen beim Deployment

Beim Deployment großer Softwareprojekte werden meist verschiedene Hard- und Softwareumgebungen verwendet.**[14][15]**

- **Development Environment:** Hier benutzt jeder Entwickler seine eigene Hard- und Softwareumgebung. Man arbeitet sich in das Thema ein und entwickelt die ersten Teile des späteren Systems.

- **Integration Environment:** Hier wird Software im Team entwickelt, jeder Entwickler hat jedoch noch seine Umgebung. Gemeinsames Arbeiten wird über Subversion-Repositories gewährleistet.
- **Test Environment:** Das Testsystem hat eine ähnliche Konfiguration wie das Live-System, jedoch sollte schon die gleiche Version des Webservers benutzt werden. In der Test Environment werden die letzten Vorbereitungen für die Staging Environment getroffen.
- **Staging Environment:** Die Umgebung sollte identisch zum späteren Live-System sein, das heißt, gleiche Hard- und Software einschließlich gleiche Updates. Die Entwicklung der Software sollte in dieser Phase des Deployments abgeschlossen sein.**[16]**
- **Production Environment:** Die Production Environment stellt das eigentliche Live-System dar. In dieser Umgebung wird die entwickelte Software eingesetzt.

Entwicklungsumgebungen und Web-Deployment

Für das automatisierte Deployment bieten sich einige Werkzeuge an, von denen hier zwei vorgestellt werden sollen.
Microsoft Visual Studio 2005 Web Deployment Projects bezeichnet ein Plugin für Microsoft Visual Studio 2005, mit dem das Deployment und Redeployment direkt in der Entwicklungsumgebung unterstützt wird. Es bietet Features wie Vorkompilierung, Vorkompilierung in eine einzelne Datei, Management von Serverkonfigurationen und der Ausschluss bestimmter Dateien vom Export/Import.**[17]**
Capistrano bezeichnet ein Deployment Tool, das hauptsächlich für Web Anwendungen, die auf *Ruby on Rails* basieren, gedacht ist, aber z.B. auch für PHP benutzt werden kann. Capistrano kann auf mehreren Servern parallel laufen und übernimmt somit Umkonfigurationen auf allen Servern gleichzeitig. Den Servern können verschiedene Rollen zugewiesen werden, wie Application Server oder Database Server, jeder mit einer eigenen Konfiguration. Die Beschreibung der Sachverhalte wird in einem so genannten Deployment Recipe vorgenommen, das an eine Art Shell-Script erinnert.**[18]**

Fazit

Es existieren viele Angriffsvektoren die man bei der Entwicklung von Webanwendungen beachten sollte. In dieser Arbeit wurden einige Ansätze zur Abschwächung oder Eliminierung derselben vorgestellt. Die hier genannten Maßnahmen reichen jedoch nicht aus, denn nahezu täglich werden neue Sicherheitslücken bekannt, die eine Webanwendung und deren Integrität bedrohen. Es reicht daher nicht aus bestehende Sicherheitslücken und Designrichtlinien zu kennen, sondern man muss kontinuierlich über neue Entwicklungen informiert sein.

Quellenverzeichnis

[1] **Design and Deploy Secure Web Apps with ASP.NET 2.0 and IIS 6.0 – Designing Secure Web Applications**
http://msdn.microsoft.com/msdnmag/issues/05/11/securewebapps/

[2] **Cheat Sheet: Web Application Security Frame**
http://msdn2.microsoft.com/en-us/library/ms978518.aspx

[3] **What is SOAP?**
http://searchwebservices.techtarget.com/searchWebServices/downloads/what_is_soap.swf

[4] **Detection of SQL Injection and Cross-site Scripting Attacks**
http://www.securityfocus.com/infocus/1768

[5] **Erstellen sicherer ASP.NET-Seiten und -Steuerelemente**
http://www.microsoft.com/germany/msdn/library/security/ErhoehenDerSicherheitVonWebanwendungen/secmod83.mspx?mfr=true

[6] **Exception Handling in Workflow-Driven Web Applications**
http://www2005.org/cdrom/docs/p170.pdf

[7] **JavaScript Exception Handling - Anatomy Of An Exception**
http://www.devshed.com/c/a/JavaScript/JavaScript-Exception-Handling/1/

[8] **Excpetion Handling in Web Services**
http://www.developer.com/net/csharp/article.php/10918_3088231_1

[9] **Dateien und Berechtigungen**
http://help.websitebaker.org/pages/de/knowledge-base/dateien-und-berechtigungen.php

[10] **Setting Appropriate Permissions on Web Applications**
http://www.windowsitpro.com/Articles/ArticleID/23580/23580.html

[11] **Empfehlungen für Berechtigungen und Benutzerrechte**
http://www.microsoft.com/technet/prodtechnol/windowsserver2003/de/library/ServerHelp/69780581-04f2-4321-ad10-24ab86c3eec6.mspx?mfr=true

[12] **Software Deployment**
http://en.wikipedia.org/wiki/Software_deployment

[13] **Good Web Application Development Practices**
http://www.jponrails.com/blog/articles/tag/deploying

[14] **Deployment Considerations For Your Project**
http://blogs.ittoolbox.com/c/programming/archives/deployment-considerations-for-your-project-9041

[15] **Traditional Development/Integration/Staging/Production Practice for Software Development**
http://dltj.org/2006/12/software-development-practice/

[16] **Staging site**
http://en.wikipedia.org/wiki/Staging_%28websites%29

[17] **Using Web Deployment Projects with Visual Studio 2005**
http://msdn2.microsoft.com/en-us/library/aa479568.asp

[18] **Capistrano: Automating Application Deployment**
http://manuals.rubyonrails.com/read/book/17