



Creative Commons License Deed

Namensnennung-Keine Bearbeitung 2.0 Deutschland

Sie dürfen:



das Werk vervielfältigen, verbreiten und öffentlich zugänglich machen

Zu den folgenden Bedingungen:



Namensnennung. Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen (wodurch aber nicht der Eindruck entstehen darf, Sie oder die Nutzung des Werkes durch Sie würden entlohnt).



Keine Bearbeitung. Dieses Werk darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter welche dieses Werk fällt, mitteilen. Am Einfachsten ist es, einen Link auf diese Seite einzubinden.
- Jede der vorgenannten Bedingungen kann aufgehoben werden, sofern Sie die Einwilligung des Rechteinhabers dazu erhalten.
- Diese Lizenz lässt die Urheberpersönlichkeitsrechte unberührt.

Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.

Die Commons Deed ist eine Zusammenfassung des [Lizenzvertrags](#) in allgemeinverständlicher Sprache.



Namensnennung – Keine Bearbeitung 2.0

CREATIVE COMMONS IST KEINE RECHTSANWALTSGESELLSCHAFT UND LEISTET KEINE RECHTSBERATUNG. DIE WEITERGABE DIESER LIZENZENTWURFES FÜHRT ZU KEINEM MANDATSVERHÄLTNIS. CREATIVE COMMONS ERBRINGT DIESE INFORMATIONEN OHNE GEWÄHR. CREATIVE COMMONS ÜBERNIMMT KEINE GEWÄHRLEISTUNG FÜR DIE GELIEFERTEN INFORMATIONEN UND SCHLIEßT DIE HAFTUNG FÜR SCHÄDEN AUS, DIE SICH AUS IHREM GEBRAUCH ERGEBEN.

Lizenzvertrag

DAS URHEBERRECHTLICH GESCHÜTZTE WERK ODER DER SONSTIGE SCHUTZGEGENSTAND (WIE UNTEN BESCHRIEBEN) WIRD UNTER DEN BEDINGUNGEN DIESER CREATIVE COMMONS PUBLIC LICENSE („CCPL“ ODER „LIZENZVERTRAG“) ZUR VERFÜGUNG GESTELLT. DER SCHUTZGEGENSTAND IST DURCH DAS URHEBERRECHT UND/ODER EINSCHLÄGIGE GESETZE GESCHÜTZT.

DURCH DIE AUSÜBUNG EINES DURCH DIESEN LIZENZVERTRAG GEWÄHRTEN RECHTS AN DEM SCHUTZGEGENSTAND ERKLÄREN SIE SICH MIT DEN LIZENZBEDINGUNGEN RECHTSVERBINDLICH EINVERSTANDEN. DER LIZENZGEBER RÄUMT IHNEN DIE HIER BESCHRIEBENEN RECHTE UNTER DER VORAUSSETZUNGEIN, DASS SIE SICH MIT DIESEN VERTRAGSBEDINGUNGEN EINVERSTANDEN ERKLÄREN.

1. Definitionen

- a. Unter einer **„Bearbeitung“** wird eine Übersetzung oder andere Bearbeitung des Werkes verstanden, die Ihre persönliche geistige Schöpfung ist. Eine freie Benutzung des Werkes wird nicht als Bearbeitung angesehen.
- b. Unter den **„Lizenzelementen“** werden die folgenden Lizenzcharakteristika verstanden, die vom Lizenzgeber ausgewählt und in der Bezeichnung der Lizenz genannt werden: „Namensnennung“, „Nicht-kommerziell“, „Weitergabe unter gleichen Bedingungen“.
- c. Unter dem **„Lizenzgeber“** wird die natürliche oder juristische Person verstanden, die den Schutzgegenstand unter den Bedingungen dieser Lizenz anbietet.
- d. Unter einem **„Sammelwerk“** wird eine Sammlung von Werken, Daten oder anderen unabhängigen Elementen verstanden, die aufgrund der Auswahl oder Anordnung der Elemente eine persönliche geistige Schöpfung ist. Darunter fallen auch solche Sammelwerke, deren Elemente systematisch oder methodisch angeordnet und einzeln mit Hilfe elektronischer Mittel oder auf andere Weise zugänglich sind (Datenbankwerke). Ein Sammelwerk wird im Zusammenhang mit dieser Lizenz nicht als Bearbeitung (wie oben beschrieben) angesehen.
- e. Mit **„SIE“** und **„Ihnen“** ist die natürliche oder juristische Person gemeint, die die durch diese Lizenz gewährten Nutzungsrechte ausübt und die zuvor die Bedingungen dieser Lizenz im Hinblick auf das Werk nicht verletzt hat, oder die die ausdrückliche Erlaubnis des Lizenzgebers erhalten hat, die durch diese Lizenz gewährten Nutzungsrechte trotz einer vorherigen Verletzung auszuüben.

- f. Unter dem „**Schutzgegenstand**“ wird das Werk oder Sammelwerk oder das Schutzobjekt eines verwandten Schutzrechts, das Ihnen unter den Bedingungen dieser Lizenz angeboten wird, verstanden
- g. Unter dem „**Urheber**“ wird die natürliche Person verstanden, die das Werk geschaffen hat.
- h. Unter einem „**verwandten Schutzrecht**“ wird das Recht an einem anderen urheberrechtlichen Schutzgegenstand als einem Werk verstanden, zum Beispiel einer wissenschaftlichen Ausgabe, einem nachgelassenen Werk, einem Lichtbild, einer Datenbank, einem Tonträger, einer Funksendung, einem Laufbild oder einer Darbietung eines ausübenden Künstlers.
- i. Unter dem „**Werk**“ wird eine persönliche geistige Schöpfung verstanden, die Ihnen unter den Bedingungen dieser Lizenz angeboten wird.

2. Schranken des Urheberrechts. Diese Lizenz lässt sämtliche Befugnisse unberührt, die sich aus den Schranken des Urheberrechts, aus dem Erschöpfungsgrundsatz oder anderen Beschränkungen der Ausschließlichkeitsrechte des Rechtsinhabers ergeben.

3. Lizenzierung. Unter den Bedingungen dieses Lizenzvertrages räumt Ihnen der Lizenzgeber ein lizenzgebührenfreies, räumlich und zeitlich (für die Dauer des Urheberrechts oder verwandten Schutzrechts) unbeschränktes einfaches Nutzungsrecht ein, den Schutzgegenstand in der folgenden Art und Weise zu nutzen:

- a. den Schutzgegenstand in körperlicher Form zu verwerten, insbesondere zu vervielfältigen, zu verbreiten und auszustellen;
- b. den Schutzgegenstand in unkörperlicher Form öffentlich wiederzugeben, insbesondere vorzutragen, aufzuführen und vorzuführen, öffentlich zugänglich zu machen, zu senden, durch Bild- und Tonträger wiederzugeben sowie Funksendungen und öffentliche Zugänglichmachungen wiederzugeben;
- c. den Schutzgegenstand auf Bild- oder Tonträger aufzunehmen, Lichtbilder davon herzustellen, weiterzusenden und in dem in a. und b. genannten Umfang zu verwerten;

Die genannten Nutzungsrechte können für alle bekannten Nutzungsarten ausgeübt werden. Die genannten Nutzungsrechte beinhalten das Recht, solche Veränderungen an dem Werk vorzunehmen, die technisch erforderlich sind, um die Nutzungsrechte für alle Nutzungsarten wahrzunehmen. Insbesondere sind davon die Anpassung an andere Medien und auf andere Dateiformate umfasst.

4. Beschränkungen. Die Einräumung der Nutzungsrechte gemäß Ziffer 3 erfolgt ausdrücklich nur unter den folgenden Bedingungen:

- a. Sie dürfen den Schutzgegenstand ausschließlich unter den Bedingungen dieser Lizenz vervielfältigen, verbreiten oder öffentlich wiedergeben, und Sie müssen stets eine Kopie oder die vollständige Internetadresse in Form des Uniform-Resource-Identifier (URI) dieser Lizenz beifügen, wenn Sie den Schutzgegenstand vervielfältigen, verbreiten oder öffentlich wiedergeben. Sie dürfen keine Vertragsbedingungen anbieten oder fordern, die die Bedingungen dieser Lizenz oder die durch sie gewährten Rechte ändern oder beschränken. Sie dürfen den Schutzgegenstand nicht unterlizenzieren. Sie müssen alle Hinweise unverändert lassen, die auf diese Lizenz und den Haftungsausschluss hinweisen. Sie dürfen den Schutzgegenstand mit keinen technischen Schutzmaßnahmen versehen, die den Zugang oder den Gebrauch des Schutzgegenstandes in einer Weise kontrollieren, die mit den Bedingungen dieser Lizenz im Widerspruch stehen. Die genannten Beschränkungen gelten auch für den Fall, dass der Schutzgegenstand einen Bestandteil eines Sammelwerkes bildet; sie verlangen aber nicht, dass das Sammelwerk insgesamt zum Gegenstand dieser Lizenz gemacht wird. Wenn Sie ein Sammelwerk erstellen, müssen Sie - soweit dies praktikabel ist - auf die Mitteilung eines Lizenzgebers oder Urhebers hin aus dem Sammelwerk jeglichen Hinweis auf diesen Lizenzgeber oder diesen Urheber entfernen. Wenn Sie den Schutzgegenstand bearbeiten, müssen Sie - soweit dies praktikabel ist - auf die Aufforderung eines Rechtsinhabers hin von der Bearbeitung

jeglichen Hinweis auf diesen Rechtsinhaber entfernen.

- b. Wenn Sie den Schutzgegenstand oder ein Sammelwerk vervielfältigen, verbreiten oder öffentlich wiedergeben, müssen Sie alle Urhebervermerke für den Schutzgegenstand unverändert lassen und die Urheberschaft oder Rechtsinhaberschaft in einer der von Ihnen vorgenommenen Nutzung angemessenen Form anerkennen, indem Sie den Namen (oder das Pseudonym, falls ein solches verwendet wird) des Urhebers oder Rechteinhabers nennen, wenn dieser angegeben ist. Dies gilt auch für den Titel des Schutzgegenstandes, wenn dieser angegeben ist, sowie - in einem vernünftigerweise durchführbaren Umfang - für die mit dem Schutzgegenstand zu verbindende Internetadresse in Form des Uniform-Resource-Identifier (URI), wie sie der Lizenzgeber angegeben hat, sofern dies geschehen ist, es sei denn, diese Internetadresse verweist nicht auf den Urhebervermerk oder die Lizenzinformationen zu dem Schutzgegenstand. Ein solcher Hinweis kann in jeder angemessenen Weise erfolgen, wobei jedoch bei einer Datenbank oder einem Sammelwerk der Hinweis zumindest an gleicher Stelle und in ebenso auffälliger Weise zu erfolgen hat wie vergleichbare Hinweise auf andere Rechtsinhaber.
- c. Obwohl die gemäß Ziffer 3 gewährten Nutzungsrechte in umfassender Weise ausgeübt werden dürfen, findet diese Erlaubnis ihre gesetzliche Grenze in den Persönlichkeitsrechten der Urheber und ausübenden Künstler, deren berechnigte geistige und persönliche Interessen bzw. deren Ansehen oder Ruf nicht dadurch gefährdet werden dürfen, dass ein Schutzgegenstand über das gesetzlich zulässige Maß hinaus beeinträchtigt wird.

5. Gewährleistung. Sofern dies von den Vertragsparteien nicht anderweitig schriftlich vereinbart,, bietet der Lizenzgeber keine Gewährleistung für die erteilten Rechte, außer für den Fall, dass Mängel arglistig verschwiegen wurden. Für Mängel anderer Art, insbesondere bei der mangelhaften Lieferung von Verkörperungen des Schutzgegenstandes, richtet sich die Gewährleistung nach der Regelung, die die Person, die Ihnen den Schutzgegenstand zur Verfügung stellt, mit Ihnen außerhalb dieser Lizenz vereinbart, oder - wenn eine solche Regelung nicht getroffen wurde - nach den gesetzlichen Vorschriften.

6. Haftung. Über die in Ziffer 5 genannte Gewährleistung hinaus haftet Ihnen der Lizenzgeber nur für Vorsatz und grobe Fahrlässigkeit.

7. Vertragsende

- a. Dieser Lizenzvertrag und die durch ihn eingeräumten Nutzungsrechte enden automatisch bei jeder Verletzung der Vertragsbedingungen durch Sie. Für natürliche und juristische Personen, die von Ihnen eine Datenbank oder ein Sammelwerk unter diesen Lizenzbedingungen erhalten haben, gilt die Lizenz jedoch weiter, vorausgesetzt, diese natürlichen oder juristischen Personen erfüllen sämtliche Vertragsbedingungen. Die Ziffern 1, 2, 5, 6, 7 und 8 gelten bei einer Vertragsbeendigung fort.
- b. Unter den oben genannten Bedingungen erfolgt die Lizenz auf unbegrenzte Zeit (für die Dauer des Schutzrechts). Dennoch behält sich der Lizenzgeber das Recht vor, den Schutzgegenstand unter anderen Lizenzbedingungen zu nutzen oder die eigene Weitergabe des Schutzgegenstandes jederzeit zu beenden, vorausgesetzt, dass solche Handlungen nicht dem Widerruf dieser Lizenz dienen (oder jeder anderen Lizenzierung, die auf Grundlage dieser Lizenz erfolgt ist oder erfolgen muss) und diese Lizenz wirksam bleibt, bis Sie unter den oben genannten Voraussetzungen endet.

8. Schlussbestimmungen

- a. Jedes Mal, wenn Sie den Schutzgegenstand vervielfältigen, verbreiten oder öffentlich wiedergeben, bietet der Lizenzgeber dem Erwerber eine Lizenz für den Schutzgegenstand unter denselben Vertragsbedingungen an, unter denen er Ihnen die Lizenz eingeräumt hat.

- b. Sollte eine Bestimmung dieses Lizenzvertrages unwirksam sein, so wird die Wirksamkeit der übrigen Lizenzbestimmungen dadurch nicht berührt, und an die Stelle der unwirksamen Bestimmung tritt eine Ersatzregelung, die dem mit der unwirksamen Bestimmung angestrebten Zweck am nächsten kommt.
- c. Nichts soll dahingehend ausgelegt werden, dass auf eine Bestimmung dieses Lizenzvertrages verzichtet oder einer Vertragsverletzung zugestimmt wird, so lange ein solcher Verzicht oder eine solche Zustimmung nicht schriftlich vorliegen und von der verzichtenden oder zustimmenden Vertragspartei unterschrieben sind
- d. Dieser Lizenzvertrag stellt die vollständige Vereinbarung zwischen den Vertragsparteien hinsichtlich des Schutzgegenstandes dar. Es gibt keine weiteren ergänzenden Vereinbarungen oder mündlichen Abreden im Hinblick auf den Schutzgegenstand. Der Lizenzgeber ist an keine zusätzlichen Abreden gebunden, die aus irgendeiner Absprache mit Ihnen entstehen könnten. Der Lizenzvertrag kann nicht ohne eine übereinstimmende schriftliche Vereinbarung zwischen dem Lizenzgeber und Ihnen abgeändert werden.
- e. Auf diesen Lizenzvertrag findet das Recht der Bundesrepublik Deutschland Anwendung.

CREATIVE COMMONS IST KEINE VERTRAGSPARTEI DIESES LIZENZVERTRAGES UND ÜBERNIMMT KEINERLEI GEWÄHRLEISTUNG FÜR DAS WERK. CREATIVE COMMONS IST IHNEN ODER DRITTEN GEGENÜBER NICHT HAFTBAR FÜR SCHÄDEN JEDWEDER ART. UNGEACHTET DER VORSTEHENDEN ZWEI (2) SÄTZE HAT CREATIVE COMMONS ALL RECHTE UND PFLICHTEN EINES LIZENSGEBERS, WENN SICH CREATIVE COMMONS AUSDRÜCKLICH ALS LIZENZGEBER BEZEICHNET.

AUSSER FÜR DEN BESCHRÄNKTEN ZWECK EINES HINWEISES AN DIE ÖFFENTLICHKEIT, DASS DAS WERK UNTER DER CCPL LIZENSIERT WIRD, DARF KEINE VERTRAGSPARTEI DIE MARKE "CREATIVE COMMONS" ODER EINE ÄHNLICHE MARKE ODER DAS LOGO VON CREATIVE COMMONS OHNE VORHERIGE GENEHMIGUNG VON CREATIVE COMMONS NUTZEN. JEDE GESTATTETE NUTZUNG HAT IN ÜBEREINSTIMMUNG MIT DEN JEWEILS GÜLTIGEN NUTZUNGSBEDINGUNGEN FÜR MARKEN VON CREATIVE COMMONS ZU ERFOLGEN, WIE SIE AUF DER WEBSITE ODER IN ANDERER WEISE AUF ANFRAGE VON ZEIT ZU ZEIT ZUGÄNGLICH GEMACHT WERDEN.

CREATIVE COMMONS KANN UNTER <http://creativecommons.org> KONTAKTIERT WERDEN.

[« Zurück zu Commons Deed](#)

Abspaltung (Softwareentwicklung)

aus Wikipedia, der freien Enzyklopädie

Eine **Abspaltung**, auch (engl.) **Fork** (fork = Gabel, üblicherweise als Maskulinum verwendet) ist in der Softwareentwicklung ein (Neben-) **Entwicklungszweig** nach der Aufspaltung eines Projektes in zwei oder mehr Folgeprojekte, wobei Teile des Quellcodes kopiert werden und dann unabhängig von dem ursprünglichen Projekt weiterentwickelt werden.

Inhaltsverzeichnis

- 1 Details
- 2 Projekt-Beispiele
- 3 Übertragung des Begriffs
- 4 Siehe auch

Details

Gründe für einen Fork können verschiedene Ziele für das Projekt, Uneinigkeiten in der technischen Ausführung oder persönliche Unstimmigkeiten zwischen den Entwicklern sein.

Typischerweise erhält nur die größere Gruppe oder die, in der der ursprüngliche Chefentwickler verbleibt, den Originalnamen des Projektes und das damit verbundene soziale Kapital.

In der Regel führt ein Fork zu zwei verschiedenen, untereinander inkompatiblen Produkten. Manchmal werden die beiden Projekte später wieder zusammengeführt, manchmal werden gute Ideen ausgetauscht, manchmal besteht nach dem Fork keine Verbindung mehr zwischen den Projekten. Ein Fork kann sein Ursprungsprojekt überdauern und an Popularität übersteigen.

Forks finden überwiegend in freien Software-Projekten statt, da bei diesen jeder das Recht zur Weiterentwicklung und Veränderung besitzt. Manche betrachten die Aufteilung in mehrere Projekte als Schwäche der freien Software, da die einen Fork üblicherweise begleitenden Diskussionen viel menschliche Arbeitszeit und Nerven kosten, die somit nicht mehr dem eigentlichen Projekt gewidmet werden können. Es kann auch bei *Closed-Source*-Projekten zu Forks kommen, wenn mehrere Firmen zusammenarbeiten und sich die Rechte an dem Produkt teilen.

Projekt-Beispiele

(Beispiele für Projekte, die aus Aufspaltungen entstanden sind)

- Die spanische Version der Wikipedia kapselte sich ab zur Enciclopedia Libre
- Der Editor XEmacs, entstanden aus Emacs.
- Der Compiler EGCS entstand aus GCC.
- Der Internetbrowser Mozilla hat viele Tochterprojekte, einige davon sind Mozilla Firefox und

- Galeon, der wiederum aufgeteilt wurde: Epiphany.
- Das freie OpenOffice.org entstand aus StarOffice, das seitdem kommerziell weiterentwickelt wird.
- Der Multimediaplayer MPlayerXP ist aus MPlayer hervorgegangen.
- Das NTFS-Dateisystem basiert auf HPFS, dem Dateisystem von OS/2.
- Die Filesharing-Software aMule (eDonkey-Netzwerk) ging aus xMule hervor.
- Die Peer-to-Peer-Software FrostWire ging aus LimeWire hervor.
- Das BSD-Derivat OpenBSD ging aus NetBSD hervor und legt höheren Wert auf Sicherheit als das Mutterprojekt.
- X.Org ging aus XFree86 hervor, dessen Entwickler mit Version 4.4 eine andere Lizenz eingeführt haben.
- Der Composite Window Manager *Beryl* ist nach Unstimmigkeiten in den eigenen Reihen aus Compiz entstanden. Beide Projekte sind zwischenzeitlich unter dem Namen Compiz Fusion wieder vereint
- Das Web-Content-Management-System Joomla! entstand aus Mambo.

Übertragung des Begriffs

Manchmal wird der Begriff auch auf Open-Content-Projekte übertragen. Prominentes Beispiel ist dafür der Ableger von Wikipedia, Wikinfo.

Siehe auch

- Derivat (Software)
- Schisma


Von „http://de.wikipedia.org/wiki/Abspaltung_%28Softwareentwicklung%29“

Kategorie: Projektmanagement

- Diese Seite wurde zuletzt am 9. März 2008 um 17:49 Uhr geändert.
- Ihr Text steht unter der GNU-Lizenz für freie Dokumentation.
Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.

Cross-Site Scripting

aus Wikipedia, der freien Enzyklopädie

 Dieser Artikel oder Abschnitt bedarf einer Überarbeitung. Näheres ist auf der Diskussionsseite angegeben. Hilf bitte mit, ihn zu verbessern, und entferne anschließend diese Markierung.

Cross-Site Scripting (XSS) bezeichnet das Ausnutzen einer Computersicherheitslücke in Webanwendungen, indem Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext eingefügt werden, in dem sie als vertrauenswürdig eingestuft sind. Aus diesem vertrauenswürdigen Kontext kann dann ein Angriff gestartet werden.

Ziel ist es meist, an sensible Daten des Benutzers wie Cookies zu gelangen, um beispielsweise Benutzerkonten zu übernehmen (Identitätsdiebstahl).

Inhaltsverzeichnis

- 1 Terminologie
- 2 Funktionsweise
- 3 Angriffsarten
 - 3.1 Typ 0
 - 3.1.1 Beispiel
 - 3.2 Typ 1
 - 3.2.1 Beispiel
 - 3.3 Typ 2
 - 3.3.1 Beispiel
- 4 Formen des Cross-Site Scripting
- 5 Schutz
 - 5.1 Schutzmaßnahmen für Webseitenbetreiber
 - 5.2 Schutzmaßnahmen für Webseitennutzer
- 6 Siehe auch
- 7 Quellen
- 8 Weblinks

Terminologie

Die Bezeichnung „Cross-Site“ leitet sich von der Art ab, wie dieser Angriff Website-übergreifend ausgeführt wird.

Cross-Site Scripting wird manchmal auch „CSS“ abgekürzt, hat jedoch nichts mit der Cascading-Style-Sheet-Technologie zu tun, die weit häufiger CSS genannt wird. Um Verwechslungen zu vermeiden, sollte daher die Abkürzung XSS benutzt werden. (X steht im IT-Bereich des öfteren für ein Kreuz = Cross)

Funktionsweise

Beim *Cross-Site Scripting* wird Schadcode auf Seiten des Clients ausgeführt, etwa dem Webbrowser oder E-Mail-Programm. Dazu schiebt der Angreifer dem Opfer ein von ihm präpariertes HTML-Dokument unter, das beispielsweise per E-Mail verschickt wird. Oder der Angreifer lässt dem Opfer einen Hyperlink zukommen, der auf eine vom Angreifer präparierte Webseite weist oder selbst den Schadcode enthält. Häufig werden dazu auch URL-Spoofing-Techniken und andere Kodierungsverfahren eingesetzt, um den Link unauffällig oder vertrauenswürdig erscheinen zu lassen.

Ein klassisches Beispiel für *Cross-Site Scripting* ist die Übergabe von Parametern an ein CGI-Skript einer Website. So ist es unter Umständen möglich, manipulierte Daten an den Benutzer zu senden. Diese Daten sind oft Code einer clientseitigen Skriptsprache (meist JavaScript), die als Parameter an eine Website übergeben werden.

Gefährlich wird dies, wenn die Webseite, auf der der Schadcode untergebracht wurde, im lokalen Browser mit besonderen Sicherheitsrechten (Privilegien) ausgestattet ist. Der Schadcode kann dann in Abhängigkeit von der Mächtigkeit der Skriptsprache verschiedene Dinge tun, die mit den Rechten des lokalen Benutzers möglich sind.

Da aus Bequemlichkeitsgründen auf Microsoft-Windows-Systemen der lokale Benutzer häufig mit Administrator-Rechten ausgestattet ist, ist dies bereits eine potenziell sehr gefährliche Konstellation. Aber auch ohne Administrator-Rechte kann der Angreifer versuchen, durch Ausnutzung von Sicherheitslücken bei der Ausführung der betreffenden Skriptsprache diese Rechte zu erlangen.

Angriffsarten

Es gibt drei grundlegende Arten von Cross-Site-Scripting-Angriffen. (Diese werden in diesem Dokument allein der Einfachheit halber mit *Typ 0*, *Typ 1* und *Typ 2* bezeichnet.)

In den Beispielen wird zur Anschauung der einfache JavaScript-Code

```
alert ("XSS")
```

verwendet, der mithilfe der `script`-Tags in ein HTML-Dokument eingebunden wird:

```
<script>alert ("XSS") </script>
```

Dieser JavaScript-Code beschreibt zwar nur einen harmlosen Warnhinweis-Dialog mit dem Text „XSS“. Doch über dieses Schema kann auch jeder andere JavaScript-Code eingeschleust werden.

Typ 0

Diese Art der Sicherheitslücke wird *DOM-basiertes* oder *lokales* Cross-Site Scripting bezeichnet. Hierbei existiert die Sicherheitslücke in einem clientseitigen Skriptcode selbst. Dies kann beispielsweise ein JavaScript-Code sein, der einen URL-Argumentwert zur Ausgabe von HTML verwendet ohne diesen ausreichend zu prüfen. Ein solcher Angriff bedarf des gezielten Aufrufs einer kompromittierten URL.

Neuerdings werden auch Webspider missbraucht, um XSS- und SQL-Injection-Attacken auszuführen. Hierzu wird ein präparierter Link auf einer Webseite veröffentlicht. Sobald ein Webspider diesem Link folgt, löst er die Attacke aus. Dadurch taucht die IP-Adresse des Spiders und nicht die des eigentlichen Angreifers in den Protokollen des angegriffenen Systems auf. Der Angreifer kann somit anonym agieren.

Beispiel

Eine clientseitige Skriptsprache wie etwa JavaScript liest ein Argumentwert aus URL mit folgendem Schema aus:

```
http://example.com/foobar.html?arg=Argumentwert
```

Und fügt diesen nach folgendem Schema ungeprüft in das HTML-Dokument ein:

```
<p>Sie haben an die URL diesen String angehängt: Argumentwert<p>
```

Wird nun die aufrufende URL folgendermaßen manipuliert:

```
http://example.com/foobar.html?arg=<script>alert ("XSS")</script>
```

würde der durch das serverseitige Skript erzeugte HTML-Code wie folgt aussehen:

```
<p>Sie haben an die URL diesen String angehängt: <script>alert ("XSS")</script><p>
```

Somit würde obiges Skript im Kontext der aufrufenden Seite ausgeführt werden.

Typ 1

Diese Art wird als *nicht-persistentes* oder *reflektives* Cross-Site Scripting bezeichnet. Anders als beim Typ 0 liegt das anfällige Programm hier auf dem Server; ansonsten ist das Funktionsprinzip sehr ähnlich.

Hierbei wird der Umstand ausgenutzt, dass dynamische generierte Webseiten ihren Inhalt oft über URL (HTTP-GET-Methode) oder Formulare (HTTP-GET- oder HTTP-POST-Methode) übergebene Eingabewerte anpassen. *Nicht-persistent* heißt dieser Typ, da der Schadcode nur temporär bei der einzelnen Generierung der Webseite eingeschleust wird. Ruft man die Seite danach ohne die manipulierte URL oder das manipulierte Formular auf, ist der Schadcode nicht mehr enthalten.

Beispiel

Eine Suchfunktion soll das Durchsuchen sämtlicher Dokumente einer Website ermöglichen. Dabei wird auf der Ergebnisseite der Suchbegriff nochmals gesondert angezeigt. Der Suchfunktion kann der Suchbegriff entweder per URL-Argument oder über ein Formular übergeben werden, was in Anfragen nach folgendem Schema resultiert:

```
http://example.com/?suche=Suchbegriff
```

Die übergebenen Suchbegriffe werden nun von einer serverseitigen Skript- oder Programmiersprache ungeprüft auf der Ergebnisseite wieder ausgegeben:

```
<p>Sie suchten nach: Suchbegriff</p>
```

Würde nun hier, wie im Beispiel von Typ 0, folgender Suchbegriff verwendet:

```
<script>alert("XSS")</script>
```

würde dann folgender HTML-Code erzeugt:

```
<p>Sie suchten nach: <script>alert("XSS")</script></p>
```

Nach dem gleichen Prinzip kann auch über manipulierte Formulare oder Formulareingaben Schadcode eingeschleust werden. Dies ist schwieriger, da hierbei das Opfer zuerst auf eine Webseite mit manipuliertem Formular gelockt werden muss, das das Opfer dann auch nutzen muss.

Typ 2

Diese Art wird als *persistentes* Cross-Site Scripting bezeichnet. Hierbei wird der Schadcode auf dem Webserver gespeichert, wodurch er bei jeder Anfrage ausgeliefert wird. Dies ist theoretisch bei jeder Webanwendung möglich, die Benutzereingaben serverseitig speichert und diese später wieder ausliefert.

Beispiel

Eine Webseite bietet ein Gästebuch, in dem Besucher Nachrichten oder Grüße hinterlassen können. Die eingetragenen Daten werden serverseitig in einer Datenbank gespeichert und bei jedem Aufruf wieder ausgegeben.

Wird nun hier als Nachricht Folgendes eingegeben:

```
Eine wirklich sehr informative Website!<script>alert("XSS")</script>
```

Würde diese bei jedem Aufruf ausgeliefert.

Formen des Cross-Site Scripting

Es gibt viele Variationen, mit denen versucht wird, sensible Daten des Benutzers zu erlangen. Darunter beispielsweise:

- Cross-Site Authentication

- Cross-Site Cooking
- Cross-Site Tracing

Cross-Site Request Forgery (CSRF) ist *keine* Form von Cross-Site-Scripting.

Schutz

Schutzmaßnahmen für Webseitenbetreiber

Um einer Webanwendung keine Basis für XSS-Angriff zu bieten, müssen alle eingehenden Eingabewerte als unsicher betrachtet und vor der weiteren Verarbeitung auf der Serverseite geprüft werden. Dabei sollte man sich nicht darauf verlassen, „böse“ Eingaben zu definieren (Schwarze Liste), um diese herauszufiltern, da man nicht genau wissen kann, welche Angriffsmethoden^[1] es gibt. Besser ist es daher, die „guten“ Eingaben exakt zu definieren (Weiße Liste) und nur solche Eingaben zuzulassen.

Hier treffen die Zitate eines Microsoft-Mitarbeiters zu, der ein Buch *Never trust the client* und sein nächstes bereits *All incoming data is EVIL* nannte. Jegliche Daten, die einmal beim Client, d.h. beim Besucher der Website, waren, sind demnach potenziell verseucht.

Als Schutz bei einer HTML-Ausgabe (Typ 1, Typ2, teilweise Typ 0) ist es nötig, die HTML-Metazeichen (insbesondere „<“, „>“, „&“, sowie Anführungszeichen „““, „'“ in Attributwerten) durch entsprechende Zeichenreferenzen zu ersetzen, damit sie als normale Zeichen und nicht als Metazeichen behandelt werden. Erfolgt die Ausgabe in URLs („src“- und „href“-Attribut), dann muß URL-Kodierung für die Zeichen verwendet werden. Bei der Ausgabe in JavaScript-Code müssen die Zeichen mit „\“ "escaped" werden. Dabei ist zu beachten, dass an diesen 3 Stellen (HTML, URL, Skript) unterschiedliche Zeichen eine Metafunktion haben (z. B. ist „\“ für HTML kein Metazeichen), es muss also immer eine an das Ausgabemedium angepasste Kodierung verwendet werden.

Die meisten Programmier- sowie Skriptsprachen verfügen über vordefinierte Methoden zur Ersetzung der Metazeichen durch normale Zeichen.

Beispiel in PHP:

```
<?php
$boesercode = "<script>alert('XSS');</script>";
echo "Sie haben ".htmlentities($boesercode, ENT_QUOTES) . " eingegeben";

//Ausgabe: Sie haben &lt;script&gt;alert(&#039;XSS&#039;);&lt;/script&gt; eingegeben
?>
```

Beispiel in Perl:

```
#!/usr/bin/perl
use HTML::Entities;

my $boesercode = "<script>alert('XSS');</script>";
print "Sie haben " . HTML::Entities::encode($boesercode) . " eingegeben";

#Ausgabe: Sie haben &lt;script&gt;alert(&#039;XSS&#039;);&lt;/script&gt; eingegeben
```

Zusätzlich können durch Einsatz von Web Application Firewalls (WAF) zumindest in der Theorie einfache (primitive) XSS-Attacken verhindert werden. Praktisch sind sichere Anwendungen jeder WAF vorzuziehen.

Schutzmaßnahmen für Webseitennutzer

Durch Ausschalten der JavaScript-Unterstützung (Active Scripting) im Browser kann man sich gegen clientseitige XSS-Angriffe schützen. Allerdings bieten einige Browser weitere Angriffsvektoren. Dies gilt allerdings nur für "echtes" XSS, also solches, welches mit JavaScript arbeitet. Wenn nur HTML-Injection (z. B. mit <IFRAME .../>) verwendet wird, dann hilft abschalten von Scripting im Browser nicht. Gleiches gilt wenn XSS in Styles (CSS) verwendet wird.

Für einige Browser gibt es auch Erweiterungen, mit denen gezielt nur mögliche XSS-Angriffe erkannt und verhindert werden.

Siehe auch

- SQL-Injektion
- Header-Injection
- Computersicherheit
- Webanwendung

Quellen

- ↑ ha.ckers.org: XSS (Cross Site Scripting) Cheat Sheet (<http://ha.ckers.org/xss.html>) (englisch)

Weblinks

- Bundesamt für Sicherheit in der Informationstechnik (BSI): Maßnahmenkatalog und Best Practices für die Sicherheit von Webanwendungen (<http://www.bsi.de/literat/studien/websec/WebSec.pdf>)
- heise.de – Sicherheit von Webanwendungen (<http://www.heise.de/security/artikel/84149>)
- XSS-Anwendungs-/Wissenstest (<http://blogged-on.de/xss/>)
- XSS-Tutorial (http://su2.info/uni/sosi/xss_paper/node5.html)
- Maui Security Scanner – kommerzieller Web-Anwendungs-Scanner (<http://www.elanize.com>)
- Springenwerk Open-Source Cross-Site-Scripting-Scanner (<http://springenwerk.org>)
- Charset-De/Encoder für XSS-Tests (<http://h4k.in/encoding>)
- Die Funktion htmlentities auf PHP.net (<http://de.php.net/manual/de/function.htmlentities.php>)
- Michael Suttons Blog – *How Prevalent Are XSS Vulnerabilities?* (http://portal.spidynamics.com/blogs/msutton/archive/2007/01/31/How-Prevalent-Are-XSS-Vulnerabilities_3F00_.asp; (englisch))
- Informationen der Apache Foundation zu CSS-Attacken (<http://httpd.apache.org/info/css-security/>) (englisch)
- *The Cross-site Scripting FAQ* (<http://www.cgisecurity.com/articles/xss-faq.shtml>) (englisch)
- *The Cross Site Scripting Threat* (http://www.virtualforge.de/cross_site_scripting_threat.php) (englisch)
- *The impact of Cross Site Scripting on your business* (http://www.virtualforge.de/cross_site_scripting_impact.php) (englisch)
- Cross Site Scripting – Animierter Kurzfilm (<http://www.virtualforge.de/vmovie.php>) (englisch)
- Web Security Threat Classification (deutsch) (http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.de.pdf) PDF 432kb
- RSnake: *XSS Cheat Sheet, esp. for filter evasion* (<http://ha.ckers.org/xss.html>) (englisch)

Von „http://de.wikipedia.org/wiki/Cross-Site_Scripting“

Kategorien: World Wide Web | Sicherheitslücke

Wartungskategorie: Wikipedia:Überarbeiten

- Diese Seite wurde zuletzt am 5. März 2008 um 20:13 Uhr geändert.
- Ihr Text steht unter der GNU-Lizenz für freie Dokumentation.

Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.

Denial of Service

aus Wikipedia, der freien Enzyklopädie

Als **Denial of Service** (**DoS**, zu Deutsch etwa: *Dienstverweigerung*) bezeichnet man einen Angriff auf einen Host (Server) oder sonstigen Rechner in einem Datennetz mit dem Ziel, einen oder mehrere seiner Dienste arbeitsunfähig zu machen. In der Regel geschieht dies durch Überlastung. Erfolgt der Angriff koordiniert von einer größeren Anzahl anderer Systeme aus, so spricht man von **Verteilter Dienstblockade** bzw. **DDoS** (**Distributed Denial of Service**). Normalerweise werden solche Angriffe nicht per Hand, sondern mit Backdoor-Programmen oder Ähnlichem durchgeführt, welche sich von alleine auf anderen Rechnern im Netzwerk verbreiten und dem Angreifer durch solche Botnetze weitere Wirte zum Ausführen seiner Angriffe bringen.

Inhaltsverzeichnis

- 1 Funktionsweise
- 2 Beispiele
 - 2.1 Chronologie
 - 2.2 Zur Veranschaulichung
- 3 Dienstverweigerung bei herkömmlicher Überlastung
- 4 Quellen
- 5 Weblinks

Funktionsweise

Primitive DoS-Angriffe wie SYN-Flooding, PIH-Flooding oder die Smurf-Attacke belasten die Dienste eines Servers, beispielsweise HTTP, mit einer größeren Anzahl Anfragen, als dieser in der Lage ist zu bearbeiten, woraufhin er eingestellt wird oder reguläre Anfragen so langsam beantwortet, dass diese abgebrochen werden. Wesentlich effizienter ist es jedoch, wie bei WinNuke, der Land-Attacke, der Teardrop-Attacke oder dem Ping of Death Programmfehler auszunutzen, um eine Fehlerfunktion (wie einen Absturz) der Serversoftware auszulösen, worauf diese ebenso auf Anfragen nicht mehr reagiert.

Eine besondere Form stellt die **DRDoS** (**Distributed Reflected Denial of Service**)-Attacke dar. Hierbei adressiert der Angreifer seine Datenpakete nicht direkt an das Opfer, sondern an regulär arbeitende Internetdienste, trägt jedoch als Absenderadresse die des Opfers ein (IP-Spoofing). Die Antworten auf diese Anfragen stellen dann für das Opfer den eigentlichen DoS-Angriff dar. Der Ursprung des Angriffs ist für den Angegriffenen durch diese Vorgehensweise praktisch nicht mehr ermittelbar.

Im Unterschied zu anderen Angriffen will der Angreifer hier normalerweise nicht in den Computer eindringen und benötigt deshalb keine Passwörter oder Ähnliches. Jedoch kann ein DoS-Angriff Bestandteil eines Angriffs auf ein System sein, zum Beispiel bei folgenden Szenarien:

- Um vom eigentlichen Angriff auf ein System abzulenken, wird ein anderes System durch einen DoS lahmgelegt. Dies soll dafür sorgen, dass das mit der Administration betraute Personal vom eigentlichen Ort des Geschehens abgelenkt ist, bzw. die Angriffsversuche im durch den DoS

erhöhten Datenaufkommen untergehen.

- Werden Antworten eines regulären Systems verzögert, können Anfragen an dieses durch eigene, gefälschte Antworten kompromittiert werden. Beispiel hierfür ist das Hijacking fremder Domainnamen durch Liefern gefälschter DNS-Antworten.
- Als Form des Protests sind DoS-Attacken in letzter Zeit populär geworden. Zum Eigenschutz der Protestierenden werden Angriffe dieser Art im Allgemeinen von Würmern durchgeführt, die sich selbstständig auf fremden Systemen verbreiten. Entsprechend handelt es sich bei Protestaktionen dieser Art um DDoS-Attacken.

Beispiele

Chronologie

- Im Februar 2000 wurden verschiedene, große Internet-Dienste (zum Beispiel Yahoo, CNN, amazon.de, eBay) durch DDoS-Attacken lahmgelegt. Hierbei hatten sich die Angreifer Zugang zu hunderten von Computern im Internet verschafft, um die Wirksamkeit ihrer Attacken durch die Vielzahl der gleichzeitig angreifenden Rechner stark zu erhöhen.
- Mai 2001: drei Tage andauernde DDoS-Attacke gegen das CERT/CC (Computer Emergency Response Team/Coordination Center).
- August 2003: Der E-Mail-Wurm Lovsan / W32.Blaster soll die Update-Site der Firma Microsoft unerreichbar machen, wird jedoch durch Deaktivierung des Domainnamens ins Leere geführt.
- Februar 2004: Der E-Mail-Wurm Mydoom bringt die Website der Firma SCO zum Erliegen.
- Januar/Februar 2005: Ein aktiv gesteuerter DoS-Angriff legt in mehreren Angriffswellen das Online-Angebot des Heinz-Heise-Verlags für zwei Tage teilweise lahm. Der Zeitschriftenverlag stellt Strafanzeige und setzt eine Belohnung in Höhe von 10.000 Euro für sachdienliche Hinweise aus, die zur Ergreifung des Täters führen.
- April 2005: Das Online-Spielenetzwerk *PlayOnline* der Firma Square Enix, auf dem unter anderem das Spiel Final Fantasy XI läuft, ist Ziel eines DDoS-Angriffs.
- August 2005: Wegen mehreren DDoS wird das Bundeskriminalamt eingeschaltet, zum Beispiel bei der FLUXX AG, die Ziel eines gescheiterten Erpressungsversuchs um 40.000 Euro war.
- Dezember 2005: DDoS gegen die Server von dialerschutz.de, computerbetrug.de, gulli.com und antispam.de; Ermittlungsbehörden eingeschaltet.
- Februar 2007: Das Onlinespiel Ragnarok Online (euRO) der Burda Holding ist Ziel eines DoS-Angriffs.
- März 2007: DDoS gegen den Service von DynDNS.org. Es wurde vor allem der Client-Server angegriffen welcher die IP-Adressen der dynamischen Hostnamen annimmt.
- April / Mai 2007: Die Server der estländischen Regierung und von Unternehmen in Estland werden in mehreren Wellen angegriffen und brechen zeitweilig zusammen. Es soll sich um den schwersten DDoS-Angriff gegen ein Land gehandelt haben.
- November 2007: DDoS-Angriff gegen antispam.de und aa419.org, die Betreiber gehen von einem Zusammenhang aus.^[1]
- Dezember 2007: Das beliebte Browserspiel OGame wird Ziel eines DDoS-Angriffes.^[2]
- Februar 2008: Das Browserspiel OGame wird wieder zum Ziel einer DDoS Attacke.^[3]
- März 2008: Vampirefreaks.com, eine beliebte Seite für die Gothic/Industrial-Subkultur wird Opfer einer DDoS Attacke

Die beobachteten Angriffe basierten auf zwei wesentlichen Schwachstellen: Zum einen konnten die Absenderadressen der „angreifenden“ Datenpakete gefälscht werden (IP-Spoofing), zum anderen konnte vor dem eigentlichen Angriff auf einer großen Anzahl dritter – nur unzureichend geschützter – Internet-Rechner unberechtigt Software installiert werden, die dann ferngesteuert durch massenhaft versendete Datenpakete den eigentlichen Angriff ausführten. Das Besondere an diesen DDoS-Angriffen ist, dass diese auch diejenigen treffen können, die sich ansonsten optimal vor Eindringlingen aus dem Internet geschützt haben. Insofern sind Rechner, auf denen nicht einmal so genannte Grundschutzmaßnahmen umgesetzt sind, nicht nur für den jeweiligen Betreiber eine Gefahr, sondern auch für alle anderen Computer im Internet.

Zur Veranschaulichung

Werden in einer HTML-Datei bestimmte aktive Inhalte in einem Browser ausgeführt, indem beispielsweise mit der Maus über einen Hyperlink gefahren wird (Klicken ist dabei nicht zwingend notwendig), versucht der Browser bei entsprechend niedrigen Sicherheitseinstellungen zum Beispiel durch das Aneinanderhängen von Zeichenketten sehr viel längere Zeichenketten zu erzeugen, die nach und nach den Arbeitsspeicher des Rechners füllen. Dieser Speicher steht anderen Anwendungen dann nicht mehr zur Verfügung und muss gegebenenfalls sogar auf die Festplatte ausgelagert werden. Dadurch wird der Rechner stark ausgelastet und für andere Aufgaben unter Umständen blockiert. Der Arbeitsspeicher wird erst wieder freigegeben, wenn das Browserfenster geschlossen wurde.

Dienstverweigerung bei herkömmlicher Überlastung

Führt der sprunghafte Anstieg von Anfragen an eine bisher nur gering frequentierte Website aufgrund der Berichterstattung in einem publikumswirksamen Medium (wie dem IT-Online-Magazin Slashdot) zu deren Überlastung und damit zur Dienstverweigerung, wird das zumeist Slashdot-Effekt genannt und gelegentlich scherzhaft mit einem DDoS-Angriff verglichen.

Quellen

- ↑ Zitat von der temporären Startseite von antispam.de: „Die Betreiber von Antispam.de gehen aufgrund der zeitlichen Überschneidung von einem Zusammenhang mit der DDoS auf die Anti-Scam-Seite aa419.org (Artists Against 419) aus, die sich mit dem als 'Nigeria-Mails' bekannten Vorschussbetrug befasst und diesen bekämpft.“
- ↑ *DDOS Attacke auf OGame* im OGame-Forum (<http://board.ogame.de/thread.php?threadid=563304>)
- ↑ [<http://board.ogame.de/thread.php?threadid=592901> *DDOS auf die Startseite* im OGame-Forum

Weblinks

- Funktionsweise des DoS (Bundesamt für Sicherheit in der Informationstechnik) (<http://www.bsi.de/fachthem/sinet/gefahr/toolsana.htm>)
- Eine Liste freier deutschsprachiger Dokumente zum Thema (<http://www.compute.ch/download.php?list.7>)
- Ausführliche Beschreibung des Denial of Service (<http://www.highgames.com/?set=hardwareview&view=8>)
- Detaillierte Beschreibung eines Denial of Service Angriffs, anhand von grafischen Mitteln (<http://www.irc-mania.de/DDOS.php>)

Von „http://de.wikipedia.org/wiki/Denial_of_Service“

Kategorie: Sicherheitslücke

- Diese Seite wurde zuletzt am 6. März 2008 um 23:46 Uhr geändert.
- Ihr Text steht unter der GNU-Lizenz für freie Dokumentation.
Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.

HTTP-Authentifizierung

Stellt der Webserver fest, dass für eine angeforderte Datei Benutzername oder Passwort nötig sind, meldet er das dem Browser mit dem Statuscode *401 Unauthorized* und dem Header *WWW-Authenticate*. Größere Webauftritte verwenden dieses standardisierte Verfahren jedoch nur selten, da sich die Eingabefelder für Benutzername und Passwort nur mit Javascript in die Webseite einbetten lassen. Dieser Javascript-Anmeldung sollte keine Barriere darstellen, wenn ohne Javascript die gewohnte Passwort-Abfrage des Browsers erscheint. Auf kleinen Homepages ist HTTP-Authentifizierung aber oft zu finden, da viele Webspaceanbieter eine simple Möglichkeit zur Konfiguration bieten.

Es gibt mehrere Möglichkeiten, Benutzer (Clients) zu authentifizieren. Verbreitet sind:

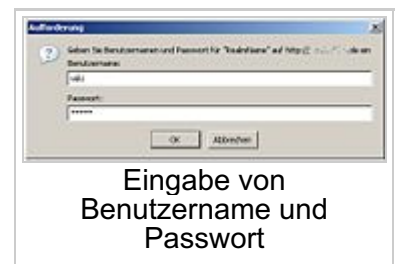
Basic Authentication

Die Basic Authentication nach RFC 2617 ist die häufigste Art der HTTP-Authentifizierung. Der Webserver fordert mit

```
WWW-Authenticate: Basic realm="RealmName"
```

eine Authentifizierung an, wobei RealmName eine Beschreibung des geschützten Bereiches darstellt. Der Browser sucht daraufhin nach Benutzername/Passwort für diese Datei und fragt gegebenenfalls den Benutzer. Anschließend sendet er die Authentifizierung mit dem Authorization-Header in der Form Benutzername:Passwort Base64-codiert an den Server. Beispiel:

```
Authorization: Basic d2lraTpwZWRRpYQ==
```



d2lraTpwZWRRpYQ== ist die Base64-Codierung von *wiki:pedia* und steht damit für Benutzername *wiki*, Passwort *pedia*. Ein Nachteil dieses Verfahrens ist, dass Benutzername und Passwort nur aus technischen Gründen codiert, jedoch nicht verschlüsselt werden. Bei einer Verschlüsselung mit SSL/TLS bei HTTPS wird bereits vor der Übermittlung des Passwortes eine verschlüsselte Verbindung aufgebaut, so dass auch bei Basic Authentication das Passwort nicht abhörbar ist.

max execution time [integer](#)

Legt die maximale Zeit in Sekunden fest, die ein Skript laufen darf, bevor der Parser die Ausführung stoppt. Diese Einstellung hilft zu verhindern, dass schlampig geschriebene Skripte Ihren Server lahmlegen. Der Standardwert für diese Einstellung ist 30 Sekunden.

Die maximale Ausführungszeit beinhaltet keine Systemaufrufe, Stream Operationen, usw. Weitere Details finden Sie bei der [set_time_limit\(\)](#) Funktion.

Bei aktiviertem [Safe Mode](#) können Sie diese Einstellung nicht mit [ini_set\(\)](#) verändern. Die einzige Möglichkeit diese Beschränkung zu umgehen besteht darin, entweder den Safe Mode abzustellen, oder das Zeitlimit in der *php.ini* zu verändern.

Ihr Webbrowser kann andere timeout Beschränkungen haben. Apache z.B. kennt die *Timeout* Anweisung, IIS hat eine CGI timeout Funktion. Beide sind im Standard auf 300 Sekunden eingestellt. Für die Bedeutung dieser timeouts schauen Sie bitte in die Dokumentation Ihres Webrowsers.

mysql_real_escape_string

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_real_escape_string — Maskiert spezielle Zeichen innerhalb eines Strings für die Verwendung in einer SQL-Anweisung

Beschreibung

```
string mysql_real_escape_string ( string $unescaped_string [, resource $link_identifier ] )
```

Maskiert spezielle Zeichen im *unescaped_string* unter Berücksichtigung des aktuellen Zeichensatzes der Verbindung, so dass das Ergebnis ohne Probleme in [mysql_query\(\)](#) verwendet werden kann. Wenn Sie Binärdaten einfügen wollen, müssen Sie die Funktion auf jeden Fall verwenden.

mysql_real_escape_string() ruft die Funktion `mysql_real_escape_string` der MySQL-Bibliothek auf, die folgende Zeichen mit einem Backslash (`\`) versieht: `\x00`, `\n`, `\r`, `\`, `'`, `"` und `\x1a`.

Die Funktion muss immer (mit wenigen Ausnahmen) verwendet werden, um Daten abzusichern, bevor sie per Query an MySQL übermittelt werden.

Parameter Liste

unescaped_string

Der zu maskierende String.

Verbindungs-Kennung

Die MySQL-Verbindung. Wird die Verbindungskennung nicht angegeben, wird die letzte durch [mysql_connect\(\)](#) geöffnete Verbindung angenommen. Falls keine solche Verbindung gefunden wird, wird versucht, eine Verbindung aufzubauen, wie es beim Aufruf von [mysql_connect\(\)](#) ohne Angabe von Argumenten der Fall wäre. Falls zufällig keine Verbindung gefunden oder aufgebaut werden kann, wird eine Warnung der Stufe `E_WARNING` erzeugt.

Rückgabewerte

Gibt einen maskierten String oder im Fehlerfall **FALSE** zurück.

Beispiele

Example#1 Einfaches mysql_real_escape_string()-Beispiel

```
<?php
// Verbindung herstellen
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    OR die(mysql_error());

// Anfrage erstellen
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%s'",
    mysql_real_escape_string($user),
    mysql_real_escape_string($password));

?>
```

Example#2 Ein beispielhafter SQL Injection Angriff

```
<?php
// Datenbankabfrage zur Ueberpruefung der Logindaten
$query = "SELECT * FROM users WHERE user='{$_POST['username']}' AND password='{$_POST['password']}'";
mysql_query($query);

// Wir haben $_POST['password'] nicht geprueft, es koennte also alles darin
// stehen, was der User will. Zum Beispiel:
$_POST['username'] = 'aidan';
```

```
$_POST['password'] = '' OR '=';

// Das bedeutet, der an MySQL gesendete Query wuerde sein:
echo $query;
?>
```

Die Abfrage, die an MySQL übermittelt wird:

```
SELECT * FROM users WHERE user='aidan' AND password='' OR '='
```

Dies würde jedermann erlauben, sich ohne valides Passwort einzuloggen.

Example#3 Optimale Vorgehensweise zur Querybehandlung

Die Verwendung von **mysql_real_escape_string()** bei jeder Variablen beugt SQL Injection Angriffen vor. Das Beispiel demonstriert ein optimales Verfahren für Datenbankabfragen, das unabhängig vom für [Magic Quotes](#) gesetzten Wert funktioniert.

```
<?php

if (isset($_POST['product_name'])
    && isset($_POST['product_description'])
    && isset($_POST['user_id'])) {

    // Verbinden mit der Datenbank
    $link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')

    if(!is_resource($link)) {

        echo "Verbindung zum Server fehlgeschlagen\n";
        // ... den Fehler loggen

    } else {

        // Die Auswirkungen von magic_quotes_gpc/magic_quotes_sybase zurücksetzen,
        // sofern die Option auf ON gesetzt ist

        if(get_magic_quotes_gpc()) {
            $product_name      = stripslashes($_POST['product_name']);
            $product_description = stripslashes($_POST['product_description']);
        } else {
            $product_name      = $_POST['product_name'];
            $product_description = $_POST['product_description'];
        }

        // einen sicheren Query zusammenstellen
        $query = sprintf("INSERT INTO products (`name`, `description`, `user_id`) VALUES ('%s', '%s', %d)"
            . mysql_real_escape_string($product_name, $link),
            mysql_real_escape_string($product_description, $link),
            $_POST['user_id']);

        mysql_query($query, $link);

        if (mysql_affected_rows($link) > 0) {
            echo "Produkt eingefuegt\n";
        }
    }
} else {
    echo "Fuellen Sie das Formular korrekt aus.\n";
}
?>
```

Die Anfrage wird jetzt korrekt ausgeführt und SQL Injection Angriffe funktionieren nicht mehr.

Anmerkungen

Hinweis: Sie müssen eine Verbindung zu MySQL geöffnet haben, bevor Sie **mysql_real_escape_string()** verwenden, ansonsten erhalten Sie einen Fehler vom Typ *E_WARNING* und der Rückgabewert wird zu **FALSE**. Ist *link_identifier* nicht angegeben, wird die letzte MySQL-Verbindung verwendet.

Hinweis: Ist [magic_quotes_gpc](#) aktiviert, wenden Sie zuerst [stripslashes\(\)](#) auf die Daten an. Das Bearbeiten bereits in irgend einer Form maskierter Daten durch `mysql_real_escape_string` führt ansonsten dazu, dass bereits Maskiertes doppelt maskiert wird.

Hinweis: Wenn die Funktion nicht verwendet wird, um die Daten zu maskieren, ist der Query anfällig für [SQL Injection Angriffe](#).

Hinweis: `mysql_real_escape_string()` maskiert weder `%` noch `_`. Diese Zeichen werden in MySQL als Platzhalter interpretiert, wenn sie mit *LIKE*, *GRANT* oder *REVOKE* kombiniert werden.

Siehe auch

- [mysql_client_encoding\(\)](#)
- [addslashes\(\)](#)
- [stripslashes\(\)](#)
- Die [magic_quotes_gpc](#)-Direktive
- Die [magic_quotes_runtime](#)-Direktive

pg_escape_string

(PHP 4 >= 4.2.0, PHP 5)

pg_escape_string — Maskiert einen String zum Einfügen in Felder mit text/char Datentypen

Beschreibung

```
string pg_escape_string ( string $data )
```

pg_escape_string() Maskiert einen String zum Einfügen in Felder mit text/char Datentypen von PostgreSQL. Der Rückgabewert ist der maskierte String. Diese Funktion sollte anstelle von [addslashes\(\)](#) verwendet werden. Falls der Datentyp der Spalte bytea ist, müssen Sie stattdessen [pg_escape_bytea\(\)](#) verwenden.

Hinweis: Diese Funktion setzt PostgreSQL 7.2 oder höher voraus.

Parameter Liste

data

Ein [string](#) mit den Daten, die maskiert werden müssen.

Rückgabewerte

Ein [string](#) mit den maskierten Daten.

Beispiele

Example#1 pg_escape_string() Beispiel

```
<?php
// Datenbankverbindung öffnen
$dbconn = pg_connect('dbname=foo');

// Eine Textdatei (mit Hochkommas und Backslashes) auslesen
$data = file_get_contents('letter.txt');

// Die Textdaten maskieren
$escaped = pg_escape_string($data);

// und in die Datenbank einfügen
pg_query("INSERT INTO correspondence (name, data) VALUES ('My letter', '{$escaped}')");
?>
```

Siehe auch

- [pg_escape_bytea\(\)](#)

Search Documentation: Text Size: [Normal](#) / [Large](#)[Home](#) → [Documentation](#) → [Manuals](#) → [PostgreSQL 8.2](#)**PostgreSQL 8.2.6 Documentation**[Prev](#)[Fast
Backward](#)

Chapter 8. Data Types

[Fast
Forward](#)[Next](#)

8.3. Character Types

Table 8-4. Character Types

Name	Description
<code>character varying(n)</code> , <code>varchar(n)</code>	variable-length with limit
<code>character(n)</code> , <code>char(n)</code>	fixed-length, blank padded
<code>text</code>	variable unlimited length

[Table 8-4](#) shows the general-purpose character types available in PostgreSQL.

SQL defines two primary character types: `character varying(n)` and `character(n)`, where n is a positive integer. Both of these types can store strings up to n characters in length. An attempt to store a longer string into a column of these types will result in an error, unless the excess characters are all spaces, in which case the string will be truncated to the maximum length. (This somewhat bizarre exception is required by the SQL standard.) If the string to be stored is shorter than the declared length, values of type `character` will be space-padded; values of type `character varying` will simply store the shorter string.

If one explicitly casts a value to `character varying(n)` or `character(n)`, then an over-length value will be truncated to n characters without raising an error. (This too is required by the SQL standard.)

The notations `varchar(n)` and `char(n)` are aliases for `character varying(n)` and `character(n)`, respectively. `character` without length specifier is equivalent to `character(1)`. If `character varying` is used without length specifier, the type accepts strings of any size. The latter is a PostgreSQL extension.

In addition, PostgreSQL provides the `text` type, which stores strings of any length. Although the type `text` is not in the SQL standard, several other SQL database management systems have it as well.

Values of type `character` are physically padded with spaces to the specified width n , and are stored and displayed that way. However, the padding spaces are treated as semantically insignificant. Trailing spaces are disregarded when comparing two values of type `character`, and they will be removed when converting a `character` value to one of the other string types. Note that trailing spaces *are* semantically significant in `character varying` and `text` values.

The storage requirement for data of these types is 4 bytes plus the actual string, and in case of `character` plus the padding. Long strings are compressed by the system automatically, so the physical requirement on disk may be less. Long values are also stored in background tables so they do not interfere with rapid access to the shorter column values. In any case, the longest possible character string that can be stored is about 1 GB. (The maximum value that will be allowed

for n in the data type declaration is less than that. It wouldn't be very useful to change this because with multibyte character encodings the number of characters and bytes can be quite different anyway. If you desire to store long strings with no specific upper limit, use `text` or `character varying` without a length specifier, rather than making up an arbitrary length limit.)

Tip: There are no performance differences between these three types, apart from the increased storage size when using the blank-padded type. While `character(n)` has performance advantages in some other database systems, it has no such advantages in PostgreSQL. In most situations `text` or `character varying` should be used instead.

Refer to [Section 4.1.2.1](#) for information about the syntax of string literals, and to [Chapter 9](#) for information about available operators and functions. The database character set determines the character set used to store textual values; for more information on character set support, refer to [Section 21.2](#).

Example 8-1. Using the character types

```
CREATE TABLE test1 (a character(4));
INSERT INTO test1 VALUES ('ok');
SELECT a, char_length(a) FROM test1; -- (1)
```

a	char_length
ok	2

```
CREATE TABLE test2 (b varchar(5));
INSERT INTO test2 VALUES ('ok');
INSERT INTO test2 VALUES ('good');
INSERT INTO test2 VALUES ('too long');
ERROR: value too long for type character varying(5)
INSERT INTO test2 VALUES ('too long'::varchar(5)); -- explicit truncation
SELECT b, char_length(b) FROM test2;
```

b	char_length
ok	2
good	5
too l	5

(1)

The `char_length` function is discussed in [Section 9.4](#).

There are two other fixed-length character types in PostgreSQL, shown in [Table 8-5](#). The `name` type exists *only* for storage of identifiers in the internal system catalogs and is not intended for use by the general user. Its length is currently defined as 64 bytes (63 usable characters plus terminator) but should be referenced using the constant `NAMEDATALEN`. The length is set at compile time (and is therefore adjustable for special uses); the default maximum length may change in a future release. The type `"char"` (note the quotes) is different from `char(1)` in that it only uses one byte of storage. It is internally used in the system catalogs as a poor-man's enumeration type.

Table 8-5. Special Character Types

Name	Storage Size	Description
"char"	1 byte	single-character internal type
name	64 bytes	internal type for object names

[Prev](#)

Monetary Types

[Home](#)
[Up](#)

[Next](#)

Binary Data Types

[Privacy Policy](#) | Project hosted by [our server sponsors](#). | Designed by [tinysofa](#)

Copyright © 1996 – 2008 PostgreSQL Global Development Group



SQL-Injektion

aus Wikipedia, der freien Enzyklopädie

SQL-Injektion (engl. *SQL Injection*) bezeichnet das Ausnutzen einer Sicherheitslücke in Zusammenhang mit SQL-Datenbanken, die durch mangelnde Maskierung oder Überprüfung von Metazeichen in Benutzereingaben entsteht. Der Angreifer versucht dabei, über die Anwendung, die den Zugriff auf die Datenbank bereitstellt, eigene Datenbankbefehle einzuschleusen. Sein Ziel ist es, Daten in seinem Sinne zu verändern oder Kontrolle über den Server zu erhalten.

Inhaltsverzeichnis

- 1 Auftreten
- 2 Vorgang
 - 2.1 Veränderung von Daten
 - 2.2 Datenbank-Server verändern
 - 2.3 Ausspähen von Daten
 - 2.4 Einschleusen von beliebigem Code
 - 2.5 Zeitbasierte Angriffe
 - 2.6 Erlangen von Administratorrechten
 - 2.7 Blinde SQL-Injektion
- 3 Gegenmaßnahmen
 - 3.1 Visual Basic (ADOdb)
 - 3.2 Microsoft .NET Framework (ADO.NET)
 - 3.3 Java (JDBC)
 - 3.4 PHP
 - 3.5 Perl
 - 3.6 ColdFusion
 - 3.7 MS-SQL
- 4 Siehe auch
- 5 Weblinks
- 6 Einzelnachweise und Ressourcen

Auftreten

SQL-Injektionen sind dann möglich wenn Daten wie beispielsweise Benutzereingaben in den SQL-Interpreter gelangen. Denn Benutzereingaben können Zeichen enthalten, die für den SQL-Interpreter Sonderfunktion besitzen und so Einfluß von außen auf die ausgeführten Datenbankbefehle ermöglichen. Solche Metazeichen in SQL sind zum Beispiel der umgekehrte Schrägstrich, das Anführungszeichen, der Apostroph und das Semikolon.

Oft sind solche Lücken in CGI-Skripten und in Programmen zu finden, die Daten wie Webseiteninhalte oder E-Mails in SQL-Datenbanken eintragen. Nimmt ein solches Programm die Maskierung nicht korrekt vor, kann ein Angreifer durch den gezielten Einsatz von Funktionszeichen weitere SQL-Befehle einschleusen oder die Abfragen so manipulieren, dass zusätzliche Daten verändert oder ausgegeben werden. In einigen Fällen besteht auch die Möglichkeit, Zugriff auf eine Shell zu erhalten, was im Regelfall die Möglichkeit zur Kompromittierung des gesamten Servers bedeutet.

Vorgang

Veränderung von Daten

Auf einem Webserver befindet sich das Script `find.cgi` zum Anzeigen von Artikeln. Das Script akzeptiert den Parameter „ID“, welcher später Bestandteil der SQL-Abfrage wird. Folgende Tabelle soll dies illustrieren:

Erwarteter Aufruf	
Aufruf	<code>http://webserver/cgi-bin/find.cgi?ID=42</code>
Erzeugtes SQL	<code>SELECT author, subjekt, text FROM artikel WHERE ID=42</code>
SQL-Injektion	
Aufruf	<code>http://webserver/cgi-bin/find.cgi?ID=42;UPDATE+USER+SET+TYPE="admin"+WHERE+ID=23</code>
Erzeugtes SQL	<code>SELECT author, subjekt, text FROM artikel WHERE ID=42; UPDATE USER SET TYPE="admin" WHERE ID=23</code>

Dem Programm wird also ein zweiter SQL-Befehl untergeschoben, der die Benutzertabelle modifiziert.

Datenbank-Server verändern

Auf einem Webserver findet sich das Script `search.aspx` zum Suchen nach Webseiten. Das Script akzeptiert den Parameter „keyword“, welcher später Bestandteil des SQL-Abfrage wird. Folgende Tabelle soll dies illustrieren:

Erwarteter Aufruf	
Aufruf	<code>http://webserver/search.aspx?keyword=sql</code>
Erzeugtes SQL	<code>SELECT url, title FROM myindex WHERE keyword LIKE '%sql%'</code>
SQL-Injektion	
Aufruf	<code>http://webserver/search.aspx?keyword=sql'+;GO+EXEC+cmdshell('format+C')+--</code>
Erzeugtes SQL	<code>SELECT url, title FROM myindex WHERE keyword LIKE '%sql' ;GO EXEC cmdshell('format C') --%</code>

Hier wird der eigentlichen Abfrage ein weiterer Befehl angehängt. Die zwei Bindestriche (`--`) kommentieren das Hochkomma als Überbleibsel der eigentlichen Anfrage aus, womit es ignoriert wird. Die nun generierte Abfrage ermöglicht so das Formatieren der Festplatte. Aber auch Downloads oder Ähnliches lassen sich dadurch erzeugen (am Beispiel Microsoft SQL Server).

Ausspähen von Daten

Auf manchen SQL-Implementationen ist die `UNION`-Klausel verfügbar. Diese erlaubt es, mehrere `SELECT`s gleichzeitig abzusetzen, die dann eine gemeinsame Ergebnismenge zurückliefern. Durch eine geschickt untergeschobene `UNION`-Klausel können beliebige Tabellen und Systemvariablen ausgespäht werden.

Erwarteter Aufruf

Aufruf	<code>http://webserver/cgi-bin/find.cgi?ID=42</code>
Erzeugtes SQL	<code>SELECT author, subjekt, text FROM artikel WHERE ID=42</code>

SQL-Injektion

Aufruf	<code>http://webserver/cgi-bin/find.cgi?ID=42+UNION+SELECT+login,+password,+'x'+FROM+user</code>
Erzeugtes SQL	<code>SELECT author, subjekt, text FROM artikel WHERE ID=42 UNION SELECT login, password, 'x' FROM user</code>

Das „x“ beim `UNION SELECT` ist nötig, weil alle mit `UNION` verknüpften `SELECTS` die gleiche Anzahl von Spalten haben müssen. Der Angreifer muss also wissen, wie viele Spalten die ursprüngliche Abfrage hat.

Ist der Datenbankserver fehlerhaft konfiguriert und hat beispielsweise ein aktuell mit der Datenbank verbundener Benutzer, über den die SQL-Injektion abgesetzt werden soll, Zugriff auf Systemdatenbanken, so kann der Angreifer über eine einfache SQL-Syntax wie `Systemdatenbank.SystemtabelleMitTabellenAuflistung` auf die Systemtabellen zugreifen und sämtliche Tabellen einer bestimmten Datenbank auslesen. Dadurch kann er wichtige Informationen erhalten, um weitere Angriffe durchzuführen und tiefer in das System einzudringen.

Einschleusen von beliebigem Code

Eine weniger bekannte Variante stellt gleichzeitig die potenziell gefährlichste dar. Wenn der Datenbankserver die Kommandos `SELECT ... INTO OUTFILE` beziehungsweise `SELECT ... INTO DUMPFILE` unterstützt, können diese Kommandos dazu benutzt werden, Dateien auf dem Dateisystem des Datenbankserver abzulegen. Theoretisch ist es dadurch möglich, falls das Bibliotheksverzeichnis des Betriebssystems oder des Datenbankservers für denselben beschreibbar ist (wenn dieser zum Beispiel als root läuft), einen beliebigen Code auf dem System auszuführen.

Zeitbasierte Angriffe

Wenn der Datenbankserver Benchmark-Funktionen unterstützt, kann der Angreifer diese dazu nutzen, um Informationen über die Datenbankstruktur in Erfahrung zu bringen. In Verbindung mit dem `if`-Konstrukt sind der Kreativität des Angreifers kaum Grenzen gesetzt.

Das folgende Beispiel benötigt auf einem MySQL-Datenbankserver mehrere Sekunden, falls der gegenwärtige User root ist:

```
SELECT IF( USER() LIKE 'root%', BENCHMARK(100000,SHA1('test')), 'false');
```

Erlangen von Administratorrechten

Bei bestimmten Datenbankservern, wie dem Microsoft SQL Server, werden *Stored Procedures* mitgeliefert, die unter anderem dazu missbraucht werden können, einen neuen Benutzer auf dem angegriffenen System anzulegen.

Diese Möglichkeit kann dazu benutzt werden, um zum Beispiel eine Shell auf dem angegriffenen Rechner zu starten.

Blinde SQL-Injektion

Von einer *blinden SQL-Injektion* wird gesprochen, wenn ein Server keine deskriptive Fehlermeldung zurückliefert, aus der hervorgeht, ob der übergebene Query erfolgreich ausgeführt wurde oder nicht. Anhand verschiedenster Kleinigkeiten wie etwa leicht unterschiedlicher Fehlermeldungen oder charakteristisch unterschiedlicher Antwortzeiten des Servers kann ein versierter Angreifer häufig dennoch feststellen, ob ein Query erfolgreich war oder einen Fehler zurückmeldet.

Gegenmaßnahmen

Eine Maßnahme besteht darin, Daten die Bedeutung für den SQL-Interpreter zu nehmen. Zu diesem Zweck wird oft zum Ausfiltern oder Maskieren (sogenanntem Escapen) von Metazeichen in Benutzereingaben gegriffen. Hierdurch kann die Gefahr der Injektion gemildert oder beseitigt werden.

Generell ist die Webanwendung für die korrekte Prüfung der Eingabedaten verantwortlich, so dass vor allem die Metazeichen des betreffenden Datenbanksystems entsprechend zu maskieren sind, die für Ausnutzung dieser Sicherheitslücke verantwortlich sind. Weitergehend können auch die Eingabedaten auf die Eigenschaften erwarteten Werte geprüft werden. So bestehen deutsche Postleitzahlen beispielsweise nur aus Ziffern. Geeignete Schutzmaßnahmen sind in erster Linie dort zu implementieren.

Der simple und sicherere Weg ist jedoch, die Daten überhaupt vom Interpreter fernzuhalten. Dabei lässt sich auch ohne Verstümmelung der Eingabe auskommen. Die Technik dazu sind gebundene Parameter Prepared Statements. Dabei werden die Daten als Parameter an einen bereits kompilierten Befehl übergeben. Die Daten werden somit nicht interpretiert und eine Injektion verhindert. Stored Procedures bieten dagegen keinen generellen Schutz vor SQL-Injection, insbesondere dann nicht, wenn der SQL-Code der Funktion nicht bekannt ist.

Doch auch auf Seiten des Datenbankservers lassen sich Sicherheitsvorkehrungen treffen. So sollten die Benutzer, mit denen sich eine Webanwendung beim Datenbankserver authentifiziert, nur die Privilegien besitzen, die er tatsächlich benötigt. So können zumindest einige der möglichen Angriffe unwirksam werden.

Hat ein Betreiber eines Webserver keine Kontrolle über die Anwendungen kann durch Einsatz von Web Application Firewalls (WAF) zumindest teilweise verhindert werden, dass SQL-Injektion-Schwachstellen ausgenutzt werden können.

Es ist nicht schwer, bestehende Programme so umzubauen, dass SQL-Injektionen nicht mehr möglich sind. Das hauptsächliche Problem der meisten Programmierer ist fehlendes Wissen über diese Art von Angriffen. Nachfolgend einige Beispiele, um die Angriffe abzuwehren.

Visual Basic (ADODB)

In Visual Basic gibt es einfache *Command*-Objekte, mit denen diese Probleme vermieden werden können.

Anstatt

```
cn.Execute "SELECT spalte1 FROM tabelle WHERE spalte2 = '"  
    & spalte2Wert & "'"
```

sollte Folgendes verwendet werden:

```
Dim cmd As ADODB.Command, rs as ADODB.Recordset
With cmd
    Set .ActiveConnection = cn
    Set .CommandType = adCmdText
    Set .CommandString = "SELECT spalte1 FROM tabelle WHERE spalte2=paramSp2"
    .Parameters.Append .CreateParameter("paramSp2", adVarChar, adParamInput, 25, spalte2Wert) '25 ist die max. l nge
    Set rs = .Execute
End With
```

Microsoft .NET Framework (ADO.NET)

Im .NET Framework gibt es einfache Objekte, mit denen solche Probleme umgangen werden k nnen.

Anstatt

```
SqlCommand cmd = new SqlCommand("SELECT spalte1 FROM tabelle WHERE spalte2 = '"
    + spalte2Wert + "';");
```

sollte Folgendes verwendet werden:

```
string spalte2Wert = "Mein Wert";
SqlCommand cmd = new SqlCommand("SELECT spalte1 FROM tabelle WHERE spalte2 = @spalte2Wert;");
cmd.Parameters.AddWithValue("spalte2Wert", spalte2Wert);
```

Java (JDBC)

Eine SQL-Injektion kann leicht durch bereits vorhandene Funktion verhindert werden. In Java wird zu diesem Zweck die PreparedStatement-Klasse verwendet (JDBC-Technologie).

Anstatt

```
Statement stmt = con.createStatement();
ResultSet rset = stmt.executeQuery("SELECT spalte1 FROM tabelle WHERE spalte2 = '"
    + spalte2Wert + "';");
```

sollte Folgendes verwendet werden:

```
PreparedStatement pstmt = con.prepareStatement("SELECT spalte1 FROM tabelle WHERE spalte2 = ?");
pstmt.setString(1, spalte2Wert);
ResultSet rset = pstmt.executeQuery();
```

Der Mehraufwand an Schreiarbeit durch die Verwendung der PreparedStatement-Klasse zahlt sich au erdem durch einen Performancegewinn aus, da durch die Angaben bereits im Voraus eine Optimierung durchgef hrt werden kann.

PHP

In PHP steht zu diesem Zweck die Funktion `mysql_real_escape_string()`^[1] zur Verf gung, die jedoch lediglich f r eine MySQL-Verbindung benutzbar ist. Wird eine andere Datenbank benutzt, so ist diese Funktion nicht geeignet. Doch PHP h lt f r fast jede Datenbank eine solche Escape-Funktion bereit. Die PHP-Oracle-Funktionen besitzen jedoch beispielsweise keine Escape-Funktion, hingegen k nnen

dort Prepared Statements verwendet werden, was bei der beliebten MySQL-Datenbank erst mit den Funktionen von MySQLi^[2] möglich geworden ist.

Anstatt

```
$abfrage = "SELECT spalte1 FROM tabelle WHERE spalte2 = '". $_POST['spalte2Wert']."'";  
$query = mysql_query($abfrage) or die("Datenbankabfrage ist fehlgeschlagen!");
```

sollte Folgendes verwendet werden:

```
$abfrage = "SELECT spalte1 FROM tabelle WHERE  
    spalte2 = '".mysql_real_escape_string($_POST['spalte2Wert'])."'";  
$query = mysql_query($abfrage) or die("Datenbankabfrage ist fehlgeschlagen!");
```

Ab PHP 5.1 sollten PHP Data Objects für Datenbankabfragen verwendet werden.

```
$dbh->exec("INSERT INTO REGISTRY (name, value)  
VALUES ('". $dbh->quote($name,PDO::PARAM_STR) .", " . $dbh->quote($value,PDO::PARAM_INT) .")");
```

Oder als Prepared Statement:

```
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");  
$stmt->bindParam(':name', $name);  
$stmt->bindParam(':value', $value);
```

Häufig wird aus Bequemlichkeit einfach die Konfigurationsoption „magic_quotes_gpc“ auf „on“ gestellt, mit der von außen kommende Benutzereingaben automatisch maskiert werden. Doch dies ist nicht empfehlenswert. Denn manche nicht selber programmierte Scripte setzen eigenständig Funktionen wie etwa addslashes()^[3] oder das bereits weiter oben genannte mysql_real_escape_string()^[1] ein. D. h. dass bereits allen relevanten Zeichen in den Benutzereingaben durch Magic Quotes^[4] ein Backslash vorangestellt wurde und nun durch die Escape-Funktion erneut ein Backslash vorangestellt wird. Somit werden die Benutzereingaben verfälscht, und man erhält anstatt eines einfachen Anführungszeichens ein Anführungszeichen mit vorangestelltem Backslash (\\"). Auch aus Gründen der Portabilität sollte bei der Entwicklung von Anwendungen auf diese Einstellung verzichtet und stattdessen alle Eingaben manuell validiert und maskiert werden, da nicht davon ausgegangen werden kann, dass auf allen Systemen dieselben Einstellungen vorherrschen oder möglich sind.

Perl

Das datenbankunabhängige Datenbankmodul DBI unterstützt eine „prepare“-Syntax ähnlich der aus dem Java-Beispiel.

```
$statementhandle = $databasehandle->prepare("SELECT spalte1 FROM tabelle WHERE spalte2 = ?");  
$returnvalue = $statementhandle->execute( $spalte2Wert );
```

ColdFusion

Unter ColdFusion kann das <cfqueryparam>-Tag verwendet werden, welches sämtliche notwendigen

Validierungen übernimmt^[5]):

```
SELECT *
FROM courses
WHERE Course_ID = <cfqueryparam value = "#Course_ID#" CFSQLType = "CF_SQL_INTEGER">
```

MS-SQL

Über parametrisierte Kommandos kann die Datenbank vor SQL-Injektionen geschützt werden:

```
SELECT COUNT(*) FROM Users WHERE UserName=? AND UserPassword=?
```

Siehe auch

- Cross-Site Scripting

Weblinks

- *SQL Injection – Are your web applications vulnerable?*, Kevin Spett, SPI Labs (<http://www.spidynamics.com/support/whitepapers/WhitepaperSQLInjection.pdf>) (englisch, PDF)
- Abusing Poor Programming Techniques in Webserver Scripts via SQL Injection (<http://www.derkeiler.com/Mailing-Lists/securityfocus/secprog/2001-07/0001.html>) (englisch)
- *Advanced SQL Injection In SQL Server Applications*, Chris Anley, NGSSoftware Insight Security Research (http://www.nextgenss.com/papers/advanced_sql_injection.pdf) (englisch, PDF)
- Bundesamt für Sicherheit in der Informationstechnik (BSI): Maßnahmenkatalog und Best Practices für die Sicherheit von Webanwendungen (<http://www.bsi.de/literat/studien/websec/WebSec.pdf>) (PDF)
- MS Access SQL Injection Cheat Sheet (<http://www.webapptest.org/ms-access-sql-injection-cheat-sheet-EN.html>) (englisch)
- Web Security Threat Classification (http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.de.pdf) (PDF, 432kb)

Einzelnachweise und Ressourcen

- ↑ ^a ^b PHP Manual: `mysql_real_escape_string` (<http://www.php.net/manual/de/function.mysql-real-escape-string.php>)
- ↑ PHP Manual: Verbesserte MySQL-Erweiterung (MySQLi) (<http://www.php.net/manual/de/ref.mysql.php>)
- ↑ PHP Manual: `addslashes` (<http://www.php.net/manual/de/function.addslashes.php>)
- ↑ PHP Manual: Magic Quotes (<http://www.php.net/manual/de/security.magicquotes.php>)
- ↑ ColdFusion Online Hilfe (<http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000317.htm#1102474>)



Dieser Artikel wurde in die Liste der lesenswerten Artikel aufgenommen.

Von „<http://de.wikipedia.org/wiki/SQL-Injektion>“

Kategorien: Datenbanksprache | Sicherheitslücke | Lesenswert

- Diese Seite wurde zuletzt am 27. Februar 2008 um 11:10 Uhr geändert.
- Ihr Text steht unter der GNU-Lizenz für freie Dokumentation. Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.