Secure Reverse Proxy FAQ

Table of Contents

Allgemeines

Was ist das Ziel dieses Dokuments?

Dieses Dokument versucht, Antworten auf die wichtigsten Fragen im Zusammenhang mit einem Secure HTTP Reverse Proxy zu geben.

Wer sollte dieses Dokument lesen?

Dieses Dokument ist technisch orientiert und richtet sich in erster Linie an Entwickler von Webapplikationen. Teile dieses Dokuments sollten aber auch für Projektleiter, Systemadministratoren und Sicherheitsverantwortliche interessant sein.

Welche Vorkenntnisse sind zum Verständnis dieses Dokumentes notwendig?

Gewisse Teile dieses Dokuments erfordern eine Kenntnis des HTTP Protokolls. Ausserdem ist es nützlich, über die Funktionsweise eines Webservers und eines -clients (Browser) Bescheid zu wissen.

Warum wurde dieses FAQ geschrieben?

Die Erfahrung hat gezeigt, dass Entwickler von Webapplikationen häufig mit dem Konzept und der Funktionsweise eines authentisierenden Reverse Proxy nicht vertraut sind und dass gewisse Fragen in diesem Zusammenhang immer wieder auftauchen. Andererseits haben wir ebenfalls festgestellt, dass gewisse notwendige Fragen gar nicht erst gestellt werden.

Dieses FAQ ist ein Versuch, alle diese Fragen zumindest teilweise zu beantworten. Allerdings kann die theoretische Lektüre nur einen Beitrag zum Verständnis liefern. Sie kann die praktische Erfahrung im Umgang mit Reverse Proxies und Webapplikationen jedoch keinesfalls ersetzen.

Übersicht

Was ist ein "Proxy"?

Proxy heisst auf Englisch Stellvertreter, Bevollmächtigter. Im Internet-Umfeld ist ein Proxy generell ein System, welches stellvertretend für einen Benutzer eine Aktion durchführt und das Resultat zurückmeldet.

Das bekannteste Beispiel ist der "normale" HTTP Proxy, der im Browser konfiguriert werden kann und in gewissen Firmen auch zwingend verwendet werden muss. Der Browser schickt dann alle Requests an diesen Proxy statt direkt ans Zielsystem. Ein solcher Proxy erfüllt vor allem die Funktion eines Filters, meist kombiniert mit Caching Funktionalität.

Was ist ein "HTTP Reverse Proxy"?

Im Gegensatz zum "normalen" oder "Forward" Proxy dient der HTTP Reverse Proxy nicht als Stellvertreter des Benutzers bzw. seines Browsers, sondern als Stellvertreter des Webservers, welcher den Request abarbeiten soll. Der HTTP Reverse Proxy leitet den Request des Browsers an einen Webserver weiter und gibt die Antwort an den Browser des Benutzers zurück.

Wie funktioniert ein HTTP Reverse Proxy?

Die Grundfunktionalität des Reverse Proxy liegt darin, Requests für eine URL entgegenzunehmen, daraus anhand eines vorkonfigurierten Mappings einen Request an einen der angeschlossenen Webserver zu generieren und anschliessend das Ergebnis dieses Requests an den Client zurückzugeben.

Wieso wird ein HTTP Reverse Proxy verwendet?

Ein HTTP Reverse Proxy bietet sich als Lösung an, wenn gewisse Funktionalitäten (z.B. Authentisierung, Autorisierung und Verschlüsselung) zentral gelöst werden sollen. Diese müssen dann nicht mehr von jedem Webserver einzeln übernommen werden. Dies kann die Konfiguration und Administration der Webserver vereinfachen.

Der HTTP Reverse Proxy ermöglicht es auch, heterogene Infrastruktur (mehrere Webserver, ev. mit unterschiedlichen Technologien für verschiedene Services) für den Benutzer als ein logisches System zusammenzufassen und zu präsentieren.

Ein weiterer Grund für den Einsatz eines Reverse Proxy sind dessen Sicherheits-Features.

Welche Sicherheits-Features bietet ein Secure HTTP Reverse Proxy?

Ein Secure HTTP Reverse Proxy bietet folgende sicherheitsrelevanten Merkmale:

- Minimale Funktionalität und daher potenziell weniger Sicherheitslücken als ein "voller" Webserver.
- "Verstecken" des eigentlichen Webservers, der vom Internet her nicht mehr direkt zugreifbar ist.
- Untersuchung des HTTP Requests und Filtern von ungültigen oder potenziell gefährlichen Requests.
- Endpunkt für die Verschlüsselung zwischen Browser und interner Infrastruktur. Die Konfiguration der Verschlüsselung kann zentral vorgenommen werden.
- Zentrale Authentisierung und Single-Sign-On über alle angeschlossenen Webserver.
- Autorisierung der Zugriffe auf die angeschlossenen Webserver.

Ein Wort der Warnung: Der Reverse Proxy ist nur ein Element einer Sicherheitsstrategie und enthebt den Systemadministrator nicht von der Pflicht, die Webserver und die Netzwerkkomponenten mittels Sicherheitspatches auf dem neuesten Stand zu halten.

Ist das nicht eine sehr aufwendige Lösung?

Dies hängt von der Grösse und dem Sicherheitsbedürfnis einer Firma ab.

Für grössere Firmen mit heterogener Infrastruktur kann diese Lösung eine grosse Kostenersparnis bedeuten, weil nicht für jede Webapplikation dieselben Probleme immer wieder gelöst werden müssen. Ausserdem ist für solche Firmen das Schadenspotenzial durch unautorisierte Zugriffe auf die Webplattform (Defacement, unkontrollierte Preisgabe von Kundendaten) so hoch, dass sich eine solche Investition lohnt.

Für kleine Firmen erachten wir den Einsatz eines dedizierten, gut konfigurierten und mit den neuesten Sicherheits-Patches versehenen Webservers zusammen mit einer restriktiv konfigurierten Firewall als genügend. Für solche Firmen stellt sich normalerweise auch die Anforderung des Single-Sign-On nicht oder diese ist relativ trivial zu erreichen, da nur ein Webserver verwendet wird.

Welche Softwareprodukte bieten die Funktionalität

eines Secure Reverse Proxies?

Die Firma AdNovum aus Zürich (http://www.adnovum.ch) bietet mit ihrem Produkt ISIWEB eine fertige Lösung an.

Bei ISIWEB handelt es sich um einen modifizierten Apache-Webserver, bei dem alle nicht zwingend benötigte Funktionalität entfernt wurde. Dafür wurde zusätzliche Funktionalität eingebaut, um SSO (Single-Sign-On) und die Kommunikation mit dem Authentisierungsserver esAuth zu ermöglichen.

ISIWEB ist Bestandteil der Nevis Web Architektur (http://www.nevis-web.com).

Der *Sun Java System Web Proxy Server* (früher *Sun ONE Web Proxy Server*) von Sun bietet ebenfalls die Funktionalität eines Reverse Proxies. (http://wwws.sun.com/software/products/web_proxy/ds_web_proxy.html).

Lässt sich das auch kostengünstig mit Open Source realisieren?

Zurzeit ist uns kein Open Source Produkt bekannt, welches die volle Funktionalität eines Secure Reverse Proxies bietet.

Für kleine und mittlere Unternehmen, die meistens nicht auf alle Features angewiesen sind, lässt sich jedoch mit Apache eine vergleichbare Lösung implementieren:

- Das Apache Modul *mod_proxy* bietet die volle Funktionalität eines Reverse Proxies.
- Die Authentisierung lässt sich mittels "Basic Authentication" realisieren. Die User-Credentials werden dabei vom Browser mit jedem Request an den Reverse Proxy mit geschickt. Wenn die Verbindung zusätzlich mittels https verschlüsselt wird, genügt diese Lösung den meisten Sicherheitsanforderungen.
- Zusammen mit einem Directory Server wie OpenLDAP für die Benutzerverwaltung haben Sie schnell einmal eine recht ansehliche Lösung.

Mapping von URLs

Können URLs auf verschiedene Arten gemappt werden?

Erstens können URLs direkt (1:1) gemappt werden, d.h. eine URL auf dem Reverse Proxy wird auf exakt dieselbe URL auf dem Webserver abgebildet. Ein Request an den Reverse Proxy für die URL /serviceX/welcome.html wird in einen Request für die URL /serviceX/welcome.html an den Webserver abgebildet:

Zweitens können URLs auch auf eine beliebige andere Art gemappt werden, d.h. der Request für /serviceX/welcome.html könnte auf einen Request für die URL /web/xyz/welcome.html abgebildet werden:

Welches ist das empfohlene Mapping?

Es wird empfohlen, alle Inhalte soweit wie möglich direkt 1:1 zu mappen. Dies ist für den Entwickler normalerweise einfacher und vermeidet gewisse Probleme mit aktiven Inhalten, welche z.B. auf Cookies angewiesen sind. Für dynamische Inhalte sollte also nur im Notfall vom 1:1 Mapping abgewichen werden.

Statische Inhalte, welche nur relative URLs verwenden, können aber auch auf sehr einfache Weise anders gemappt werden.

Wieviele Mappings können pro Service gemacht werden?

Prinzipiell könnten beliebig viele URLs gemappt werden, wobei aber sichergestellt werden muss, dass die URLs auf dem Reverse Proxy eindeutig sind. Aus praktischen Überlegungen (Wartbarkeit der Konfiguration, Minimierung der Namenskollisionen zwischen den Services) sollte aber pro Service nur eine Basis-URL der Form /servicename verwendet werden. Unter diese URL können dann wenn notwendig auch weitere URL-Teilbäume wie z.B. /servicename/images und /servicename/other_images "eingeblendet" werden.

Fragen von Entwicklern

Wie läuft das Login ab?

Jeder Request auf einen geschützten Bereich auf dem Reverse Proxy, der mit einem unbekannten (noch nicht authentifizieren) Benutzer erfolgt, wird vom Reverse Proxy abgefangen. Anstatt der angeforderten Seite wird dem Benutzer eine Login-Seite präsentiert.

Die User-Credentials werden wiederum vom Reverse Proxy abgefangen und damit eine Authentifizierung gegen ein Backend System (LDAP, Datenbank...) durchgeführt.

Das Login ist damit abgeschlossen und es erfolgt ein Redirect auf die ursprünglich angefragte URL, umgesetzt durch das im Reverse Proxy definierte Mapping.

Benötigt eine Webapplikation eine spezielle Einstiegsseite für das Login?

Entgegen einem häufig vorkommenden Missverständnis ist keine explizite Startseite für den Anstoss der Login-Funktionalität nötig. Das Login wird bei jeder URL durchgeführt, die sich innerhalb eines geschützten Bereichs befindet, sofern noch keine gültige Benutzer-Session zwischen dem Reverse Proxy und dem Browser besteht.

Dies bedeutet, dass jede beliebige Seite innerhalb einer Webapplikation potenziell als Einstigesseite verwendet werden kann. Sofern ein Service jedoch zwingend eine spezielle Einstiegsseite benötigt, z.B. um die Benutzerdaten in seine Session abzuspeichern, so muss dies durch die Applikation selber umgesetzt werden, indem alle Requests, die noch keine Session auf dem Webserver besitzen, auf die Startseite umgeleitet werden.

Wie wird ein korrektes Logout durchgeführt und was geschieht dabei?

Ein Logout bedeutet bei Webapplikation in der Regel das Abbauen einer bis anhin gültigen Session und das Verwerfen der zwischengespeicherten Benutzerdaten. Bei Verwendung eines authentisierenden Reverse Proxy besitzt ein Benutzer in der Regel jedoch zwei Sessions: Eine mit dem Reverse Proxy und eine mit dem Webserver.

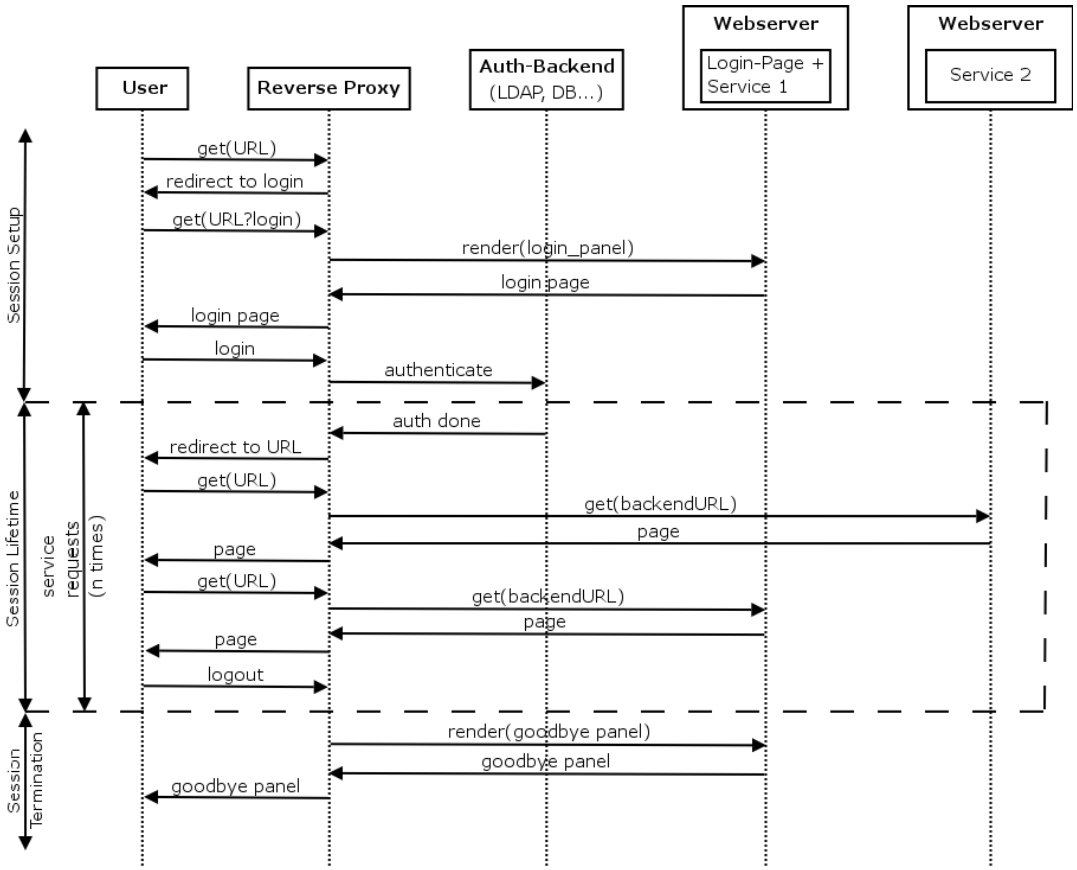
Der Mechanismus für das Logout hängt vom verwendeten Produkt ab. ISIWEB beispielsweise prüft bei jedem Request, ob in der URL der Parameter *?logout* enthalten ist. Falls ja, wird die aktuelle Session zwischen Browser und Reverse Proxy zerstört und der Benutzer ist damit ausgeloggt. Ein Link für einen expliziten Logout könnte daher beispielsweise so aussehen: *href="/serviceX/?logout"*.

Es ist allerdings zu beachten, dass die logische Session zwischen dem Browser und dem Webserver (welche üblicherweise über ein Cookie vermittelt wird) damit nicht automatisch abgebaut wird. Dies stellt eine potientielle Gefahr dar und muss vom Entwickler berücksichtigt werden, siehe dazu auch *Welche neuen Sicherheitslücken müssen bei der Verwendung eines authentifizierenden Reverse Proxy vermieden werden?*.

In diesem Zusammenhang ist es auch wichtig zu erkennen, dass diese unterschiedlichen Sessions zwischen dem Browser und dem Reverse Proxy zum einen und dem Browser und der Webapplikation zum anderen unterschiedliche Timeouts besitzen können. Damit hier keine unerwünschten Effekte entstehen können, müssen diese Timeouts miteinander synchronisiert werden, siehe dazu auch *Welches ist das empfohlene Setup für die Session-Timeouts auf dem Reverse Proxy und den Webservern?*

Wie sieht der gesamte Ablauf von Login bis Logout im Detail aus?

Das folgende Sequenzdiagramm zeigt den Ablauf von Login und Logout sowie den Lebenszyklus der Benutzersession im Detail:



Was muss ich bei der Entwicklung von statischem Content berücksichtigen?

Statischer Content (HTML-Seiten, Bilder, Scripts, etc.) ist unproblematisch, wenn soweit wie möglich mit relativen URLs gearbeitet wird.

Verwenden Sie auf keinen Fall Servernamen in Ihrem Code, ausser Sie verlinken auf Seiten ausserhalb des durch den Reverse Proxy geschützten Bereiches. Verzichten Sie auch soweit wie möglich auf absolute URLs.

Was sind absolute und relative URLs und was soll ich verwenden?

Eine absolute URL ist eine URL, welche entweder den Servernamen oder einen führenden "/" enthält.

Eine relative URL ist dagegen jede nicht absolute URL, d.h. eine URL, welche vom Browser in Bezug auf die momentan angeforderte Ressource dynamisch zusammengesetzt werden muss.

URL	Beispiele
relativ	
	• page2.html • images/back.jpg • ../main/showDetails.jsp
absolut	
	• /page2.html • /images/back.jpg
voll qualifiziert	
	• https://www.mydomain.ch/myapp/welcome.html

Wenn Sie nur relative URLs verwenden, kann Ihr Inhalt problemlos innerhalb des Webservers in ein anderes Directory (d.h. an eine andere URL) verschoben werden. Das heisst, der Inhalt ist relozierbar. Damit wird es auch viel einfacher, diesen Inhalt über einen Reverse Proxy anzuzeigen, weil nicht unbedingt ein 1:1 Mapping verwendet werden muss.

Falls in gewissen Situationen relative URL's nicht ausreichen und stattdessen absolute oder sogar voll qualifizierte URL's notwendig sind (beispielsweise für einen Redirect), bleibt die Webapplikation am flexibelsten, wenn Sie diese URL mit hilfe der Informationen des Reverse Proxy dynamisch zusammensetzen.

Falls der verwendete Reverse Proxy die Möglichkeit bietet, dem dahinterliegenden Webserver gewisse Informationen im HTTP Header zu liefern, empfiehlt sich eine Konfiguration in der folgenden Art: Ein Header "Proxy-Name" enthält die Domain resp. den Servernamen unter dem der Reverse Proxy von aussen angesprochen werden muss. Ein Header "Proxy-Location" enthält den Namen des aufgerufenen Services. Eine URL könnte damit beispielsweise so zusammengesetzt werden:

```
"https://" + Proxy-Name + Proxy-Location + "/mypath/dynamic.jsp"
```

Was soll ich machen, wenn etwas nicht so funktioniert, wie ich es erwarte?

Versuchen Sie sich an die Stelle des Webbrowsers zu versetzen. Sehen Sie sich den HTML-Code an, den der Browser anzeigt. Im Internet Explorer finden Sie diese Funktion unter "Ansicht" -> "Quelltext".

Ein weiterer Hinweis liefert der Blick ins Logfile des Reverse Proxy (access_log). Vielfach entstehen in den frühen Phasen der Integration eine grosse Anzahl an "not found" Meldungen (Error 404), weil vom Reverse Proxy URLs angefordert werden, welche diesem nicht bekannt sind. Solchen Meldungen sollten Sie unbedingt nachgehen, auch wenn der Fehler unter Umständen im Browser nicht sichtbar wird.

Was muss bei der Entwicklung von aktiven Inhalten berücksichtigt werden?

Für aktive Inhalte gilt in erster Linie dasselbe Prinzip wie bei der Entwicklung von statischen Inhalten. Normalerweise wird eine Webapplikation ausgehend von einer Stamm-URL (z.B. "/service") entwickelt, von der aus alle URLs innerhalb der Webapplikation abgeleitet sind, d.h. es handelt sich um relative URLs. Für das Session Tracking werden normalerweise Cookies eingesetzt, was zu Problemen führen kann, wenn die Webapplikation auf dem Reverse Proxy nicht 1:1 gemappt wird (siehe *Wieso funktionieren meine Cookies nicht?*).

Die Entwicklung von aktiven Inhalten bedeutet in den meisten Fällen, dass einem Benutzer gewisse Daten angezeigt werden, die nur er und kein anderer Benutzer sehen darf (personalisierter Inhalt). Dies bedeutet, dass der Benutzer authentisiert werden muss (was im vorliegenden Fall vom Reverse Proxy sichergestellt wird) und dass die Webapplikation diesen Benutzer anhand des vom Proxy kommenden Requests erkennen muss. Anschliessend muss sichergestellt werden, dass dem Benutzer die richtigen Daten angezeigt werden und dass diese Daten auch von keinem anderen Benutzer eingesehen werden können. Schliesslich muss sichergestellt werden, dass der Benutzer nur diejenigen Informationen modifizieren kann, welche explizit dafür vorgesehen sind, vom Benutzer modifiziert zu werden.

Wieso funktionieren meine Cookies nicht?

Wird Ihr Service nicht über ein 1:1 Mapping angebunden? Dies ist der häufigste Grund für einen solchen Fehler. Ihr Webserver generiert Cookies für eine bestimmte URL (beispielsweise "/service"), weil dies die URL für die Webapplikation ist. Wird dieser Service auf dem Reverse Proxy als /srv angesprochen, so sendet der Browser das Cookie nicht mehr an Ihren Service zurück, weil das Cookie nicht für die URL /srv gesetzt ist. Wenn es unmöglich ist, Ihren Service 1:1 zu mappen, so müssen Sie das Cookie "von Hand" setzen.

Gibt es mehrere Web-Applikationen, welche jeweils auf einem eigenen Webserver liegen, aber alle dieselbe URL verwenden? Dann setzen diese Web-Applikationen alle automatisch "dasselbe" Cookie, d.h. ein Cookie für dieselbe URL, und überschreiben damit das Cookie der jeweils anderen Applikation.

Woran kann ich anhand des Requests den authentisierten Benutzer erkennen?

Der Reverse Proxy schickt bei jedem Request im HTTP Header Informationen über den Benutzer an den Webserver. Dies ist mindestens der Login-Name des Benutzers ("Remote-User"). Möglicherweise wird auch noch die aktuelle Session-ID mitgegeben.

Für Anwendungen mit niedrigen Sicherheitsanforderungen genügt es, auf dem Webserver das Vorhandensein dieser HTTP Headers zu prüfen und Requests ohne diese Headers zurückzuweisen. Eventuell kann auch noch die IP-Adresse des Absenders (also hier des Reverse Proxy) überprüft werden. Dies ist bei praktisch jeder Webserver-Software über die Konfiguration möglich und muss nicht selber programmiert werden.

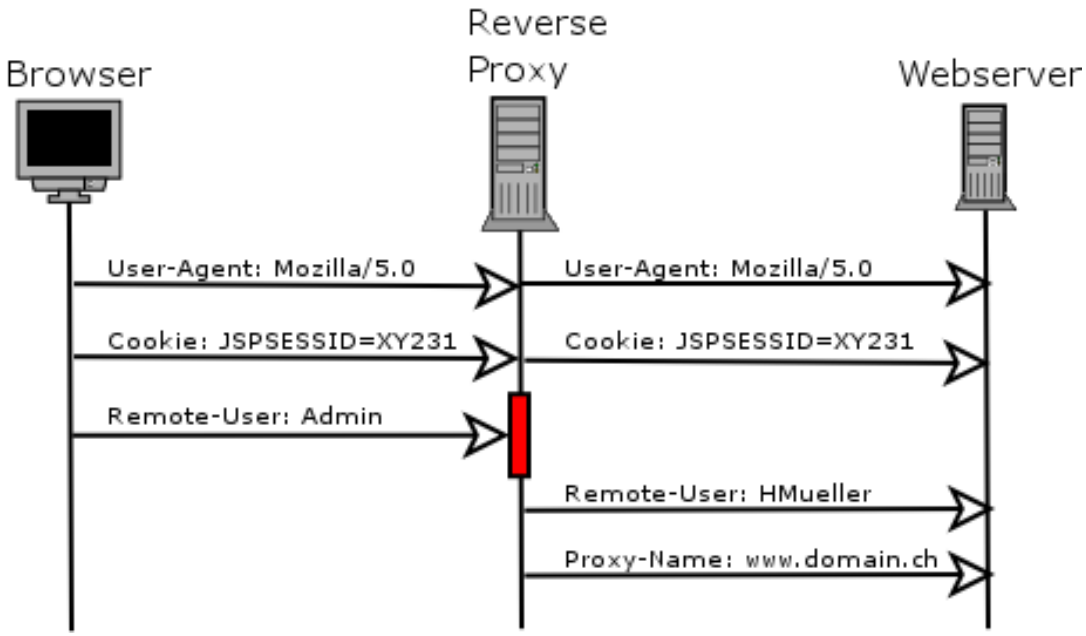
Soll eine Webapplikation mittleren oder hohen Sicherheitsanforderungen genügen, muss über weitere technische Massnahmen (z.B. Firewalls und/oder Client-Zertifikate) sichergestellt werden, dass

nur der Reverse Proxy auf den Webserver zugreifen kann.

Können in einer Webapplikation HTTP Headers, bsp. für Cookies, verwendet werden?

Der Reverse Proxy leitet den Request des Browsers bis auf einige Ausnahmen unverändert an den Webserver weiter. Es ist daher problemlos möglich, Headers des Browsers (bsp. User-Agent) in der Webapplikation auszuwerten. Ebenso verhält es sich mit Cookies.

Ein Reverse Proxy sollte zuerst alle HTTP Headers aus dem Client Request entfernen, deren Namen mit den zu übergebenden Headers (Remote-User, Session-ID, etc.) übereinstimmen. Anschliessend wird der neue Request generiert, welcher die in der Session gespeicherten Header-Daten enthält:



Wie können HTTP Headers in einer Webapplikation aus dem Request ausgelesen werden?

In Java: *request.getHeader("name")*

In ASP: *Request.ServerVariables("HTTP_name")*

oder: *Request.ServerVariables("HEADER_name")*

CGI (Perl): *$ENV("HTTP_name")*

Für Tests und zum Verifizieren, dass ein gesuchter HTTP Header tatsächlich im Request vorhanden ist, kann auch eine spezielle Webseite programmiert werden, welche die Headers im Request anzeigt, siehe die folgende Frage.

Wie können alle übergebenen HTTP Headers aus dem Request ausgelesen werden?

Für Testzwecke ist es nützlich, eine Seite zu erstellen, die alle übergebenen HTTP Headers ausliest und anzeigt. Im produktiven Betrieb müssen diese Seiten jedoch zwingend entfernt werden, damit für einen potenziellen Angreifer keine Informationen zur Verfügung gestellt werden, die ihm weiterhelfen könnten.

Für Java kann dies beispielsweise mittels eines "SnoopServlet" genannten Servlets geschehen, das einigen Servlet Engines beiliegt und auch auf dem Web zu finden ist. Der wesentliche Teil dieses Codes ist:

```
Enumeration e = req.getHeaderNames();
while (e.hasMoreElements()) {
        String name = (String)e.nextElement();
        out.println(name + ": " + req.getHeader(name));
}
```

(Quelle: http://www.geo.unizh.ch:8080/wirz/servlet_source/SnoopServlet.java.html)

Für ASP kann zum Beispiel folgender Code verwendet werden:

```
<TABLE BORDER="1">
<TR><TD><B>Server Variable</B></TD><TD><B>Value</B></TD></TR>
<% For Each strKey In Request.ServerVariables %>
<TR>
<TD><%= strKey %></TD>
<TD><%= Request.ServerVariables(strKey) %></TD>
</TR>
<% Next %>
</TABLE>
```

(Quelle:http://msdn.microsoft.com/library/en-us/iissdk/iis/ref_vbom_reqocsv.asp)

Für Perl kann folgender Code verwendet werden:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
foreach $key (keys %ENV) {
        print "$key --> $ENV{$key}<br>";
}
```

(Quelle: http://httpd.apache.org/docs/howto/cgi.html#environmentvariables)

Wie kann ich während der Entwicklung sicherstellen, dass mein Browser den Reverse Proxy nicht "umgeht"?

Dies lässt sich mit absoluter Sicherheit nur dann verhindern, wenn netzwerktechnisch derselbe Aufbau verwendet wird, wie er in der produktiven Umgebung vorhanden ist. Das heisst, der Webserver ist nur über eine Firewall erreichbar und Zugriffe auf Port 80 (oder 443) werden für alle Systeme ausser dem Reverse Proxy gesperrt.

Normalerweise ist dies während der Entwicklungsphase nicht praktikabel. Eine relativ hohe Sicherheit kann aber durch folgendes Vorgehen erreicht werden. Zuerst wird ein Prototyp der Applikation mit einigen wenigen Seiten erstellt und deren HTML-Output im Browser als Quelltext angezeigt. Der Quelltext wird dann auf die Verwendung absoluter URLs durchgesehen und anschliessend als Muster für die Entwicklung der restlichen Seiten verwendet.

Wie kann eine Webapplikation für einen authentisierenden Reverse Proxy entwickelt werden, wenn während der Entwicklungsphase kein entsprechender

Reverse Proxy zur Verfügung steht?

Dies stellt sicher eine ungünstige Situation dar und sollte wenn möglich vermieden werden, weil damit zu rechnen ist, dass dadurch ein höherer Integrationsaufwand entsteht. Wenn immer möglich, sollte bereits während der Entwicklung entweder der Reverse Proxy der Entwicklungsumgebung verwendet werden oder durch geeignete Hilfsmittel ein solcher simuliert werden, beispielsweise durch Verwendung eines der Applikation vorgeschalteten Servlet Filters, das die HTTP Header entsprechend abfüllt.

Wenn folgende Empfehlungen berücksichtigt werden, kann der Integrationsaufwand auch ohne die Verwendung eines Reverse Proxy während der Entwicklung in Grenzen gehalten werden:

- Die Webapplikation sollte eine Einstiegsseite enthalten, welche die benötigten HTTP Header (bspw. den Benutzernamen) ausliest und zusammen mit der Session-ID des Reverse Proxy in der Session des Webservers abspeichert.
- Alle weiteren Seiten sollten die Session-ID aus dem HTTP Header auslesen und mit der in der Session gespeicherten ID vergleichen. Wenn die Session-ID's übereinstimmen, können die in der Session gespeicherten Daten verwendet werden. Ansonsten sollte die Session abgebaut werden. Siehe dazu auch "Welche neuen Sicherheitslücken müssen bei der Verwendung eines authentifizierenden Reverse Proxy vermieden werden?"
- Während der Entwicklung kann die Einstiegsseite mit einer "Dummy-Seite" ersetzt werden, die einige Beispieldaten in der Session ablegt.

Sind alle Webapplikationen hinter einem Reverse Proxy automatisch sicher?

Nein, dieser Schluss ist mit Sicherheit falsch! Der Reverse Proxy übernimmt lediglich einige sicherheitsrelevante Mechanismen wie Authentifizierung oder Verschlüsselung, was einen wesentlichen Beitrag zu einer sicheren Applikation liefern kann. Fehler und Probleme in der Webapplikation, die zu einem Sicherheitsrisiko führen können, werden vom Reverse Proxy jedoch weder verhindert noch erkannt.

In Spezialfällen ist es eventuell denkbar, bekannte Sicherheitslücken eines Webservers durch Verwendung eines Reverse Proxies temporär abzusichern, indem gefährliche Requests auf dem Reverse Proxy erkannt und zurückgewiesen werden. Dies sollte aber nur im Notfall gemacht werden, z.B. wenn vom Hersteller der Webserver-Software noch kein Patch verfügbar ist oder unternehmenskritische Software auf dem gepatchten System nicht mehr läuft. Das Patchen des Webservers ist jedoch langfristig die einzige verlässliche und wartbare Lösung!

Wie können Sicherheitslücken in einer Webapplikation vermieden werden?

Es gibt in jeder Webapplikation "Todsünden", welche Sie unbedingt vermeiden sollten. Dies gilt auch dann, wenn diese Applikation nicht hinter einem Reverse Proxy ausgeführt wird. Die zehn häufigsten sicherheitskritischen Fehler dokumentiert beispielsweise "OWASP Top Ten Web Application Vulnerabilities" [http://www.owasp.org/documentation/topten.html].

Allgemein sollten Sie niemals darauf vertrauen, dass ein Request an Ihre Applikation das richtige Format besitzt oder die richtigen Daten enthält. Alle Daten, die Sie an den Browser senden und von dort wieder zurück erhalten (z.B. Daten in einem Formular oder in einem URL-Parameter) können von einem erfahrenen Benutzer abgeändert werden. Sie sollten sich mit den Strategien vertraut machen, wie eine Webapplikation entwickelt werden muss, damit der Benutzer kritische Daten nicht abändern kann.

Welche neuen Sicherheitslücken müssen bei der Ver-

wendung eines authentifizierenden Reverse Proxy vermieden werden?

Bei der Verwendung eines authentifizierenden Reverse Proxy müssen Sie darauf vertrauen, dass die vom Proxy gelieferten Daten authentisch sind. Machen Sie aber auch Gebrauch von diesen Daten! Prüfen Sie bei jedem Request die vom Reverse Proxy übergebene Session ID. Wenn sich diese ID verändert, so bedeutet dies entweder, dass sich derselbe Benutzer aus- und wieder eingeloggt hat oder dass nach dem Logout des ersten Benutzers ein zweiter denselben Browser verwendet. Verwerfen Sie in diesem Falle alle in der Session zwischengespeicherten Daten, um zu vermeiden, dass ein Benutzer die Daten eines anderen Benutzers anschauen oder ev. sogar modifizieren kann (sog. Session-Hijacking).

Falls keine Session-ID vom Reverse Proxy zu Verfügung gestellt wird, kann derselbe Mechanismus mit dem "Remote-User" angewendet werden.

Wie kann ich vermeiden, dass ein Benutzer kritische Daten meiner Applikation abändert und somit eventuell unberechtigten Zugriff erhält?

Nehmen wir als Beispiel ein Bestellverfolgungssystem, wie es in einem Internetshop gebräuchlich ist. Sie zeigen dem Benutzer eine Liste mit Bestellnummern an, aus der er dann die Bestellung ansehen kann, welche ihn interessiert. Sie verwenden dazu Links der Form *showDetail?bestellnummer=xxx*. Die Schwäche dieses Vorgehens liegt darin, dass ein Benutzer diese URL auch direkt im Browser eintippen und dabei die Bestellnummer abändern kann. Wenn Sie die Bestellnummer in showDetail nicht überprüfen, kann sich der Benutzer auch Bestellungen anderer Benutzer anzeigen lassen und eventuell dabei auch die Identität des Bestellers erfahren. Bei diesem Vorgehen müssen Sie also überprüfen, ob die angeforderte Bestellnummer auch tatsächlich vom aktuellen Benutzer abgefragt werden darf. Dies kann aber einen erheblichen Aufwand bedeuten.

Eine weit bessere Lösung besteht darin, dass Sie die gültigen Bestellnummern als Liste in der Session des Benutzers zwischenspeichern und im Link nur noch den Listenindex verlinken: z.B. *showDetail?index=5*. Auch diesen Index kann der Benutzer natürlich abändern und Sie müssen deshalb überprüfen, ob der Index einen gültigen Wert besitzt. Der Benutzer kann aber den Inhalt der Liste, die ja nur auf dem Server vorhanden ist, nicht verändern.

Dasselbe gilt sinngemäss für Formulare, welche per POST-Request an den Server gesendet werden. Der Benutzer kann das Formular lokal speichern, darin enthaltene Werte abändern und dann das lokal gespeicherte Formular auf die richtige URL posten. Auch Hidden Fields können auf diese Weise einfach verändert werden. Verwenden Sie die Angaben aus einem Formular immer nur, wenn der Benutzer Informationen liefern muss, welche Sie noch nicht besitzen, z.B. für die Versandadresse.

Im Spezialfall einer Webapplikation hinter einem Reverse Proxy sollten Sie niemals vom Browser Informationen entgegennehmen, die Ihnen der Reverse Proxy bereits liefert. So gibt es zum Beispiel keinen Grund, den Benutzernamen mittels einer URL oder dem versteckten Feld eines Forms zu "transportieren".

Kann meine Webapplikation Auswirkungen auf das Gesamtsystem oder andere Services haben?

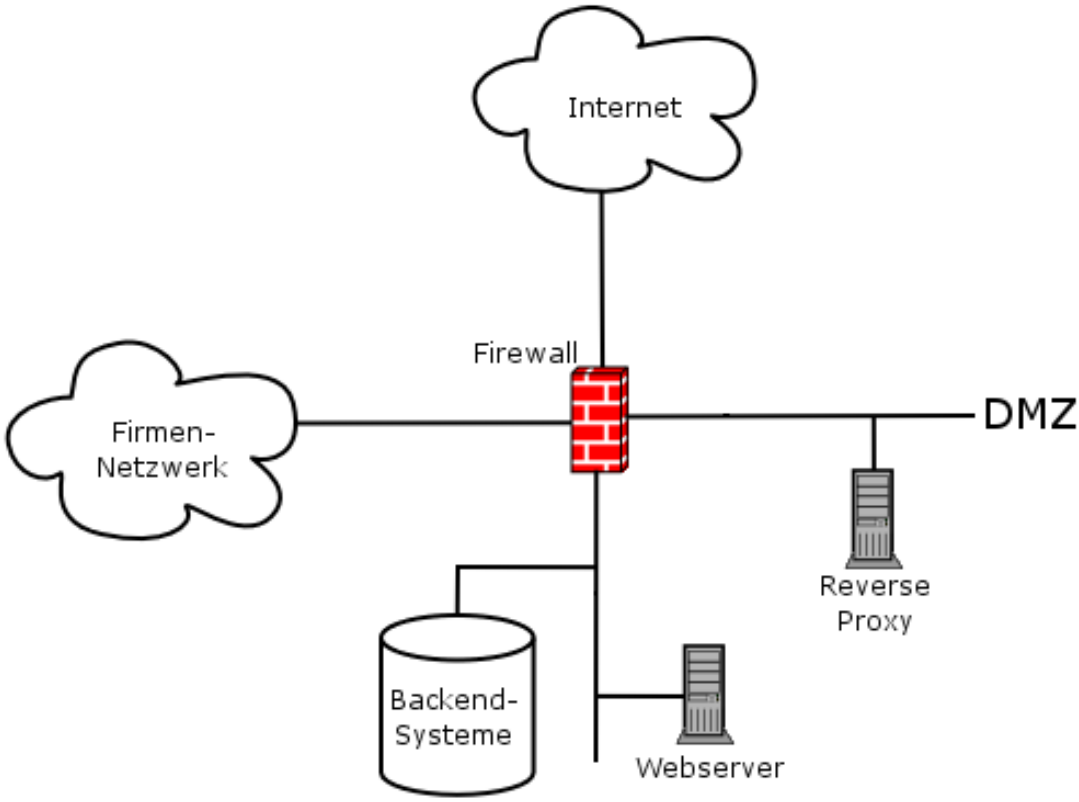
Denkbar ist eine Art *Denial of Service* (DOS) von innen.

Die Anzahl offener Requests ist beim Reverse Proxy limitiert. Falls die Antwortzeiten einer Webapplikation sehr schlecht sind, können diese langen, offenen Requests zum Webserver möglicherweise die Performance anderer Services negativ beeinflussen. Dies weil nicht mehr genügend Ressourcen für andere Requests zur Verfügung stehen.

Fragen zur System-Administration

Welches ist die empfohlene Architektur für den Aufbau einer Reverse Proxy Lösung?

Ein Reverse Proxy sollte wie jeder andere Service auch, nur hinter einer Firewall und nie direkt ans Internet (je nach grösse der Firma sogar Firmennetzwerk) angebunden werden.



Die Firewall überprüft in einem solchen Aufbau, dass Zugriffe auf die Systeme in einer Zone nur aus den dazu vorgesehenen Zonen erfolgen. So kann zum Beispiel vom Internet her nur auf genau festgelegte Server und Ports in der DMZ, der sogenannten *demilitarisierten Zone*, zugegriffen werden, nicht aber auf das interne Firmen-Netzwerk.

Welches ist das empfohlene Setup für die Session-Timeouts auf dem Reverse Proxy und den Webservern?

Vergessen Sie nicht, dass in einer Reverse Proxy Architektur immer zwei Sessions vorhanden sind, eine mit dem Reverse Proxy und eine mit dem Webserver.

Grundsätzlich sollte das Session-Timeout auf dem Webserver einige Minuten länger eingestellt sein als auf dem Reverse Proxy. Damit wird sichergestellt, dass die Session immer zuerst beim Reverse Proxy terminiert.

Welche Sicherheitsmassnahmen können zur Vermeidung unbefugter Zugriffe auf den Webserver ergriffen werden?

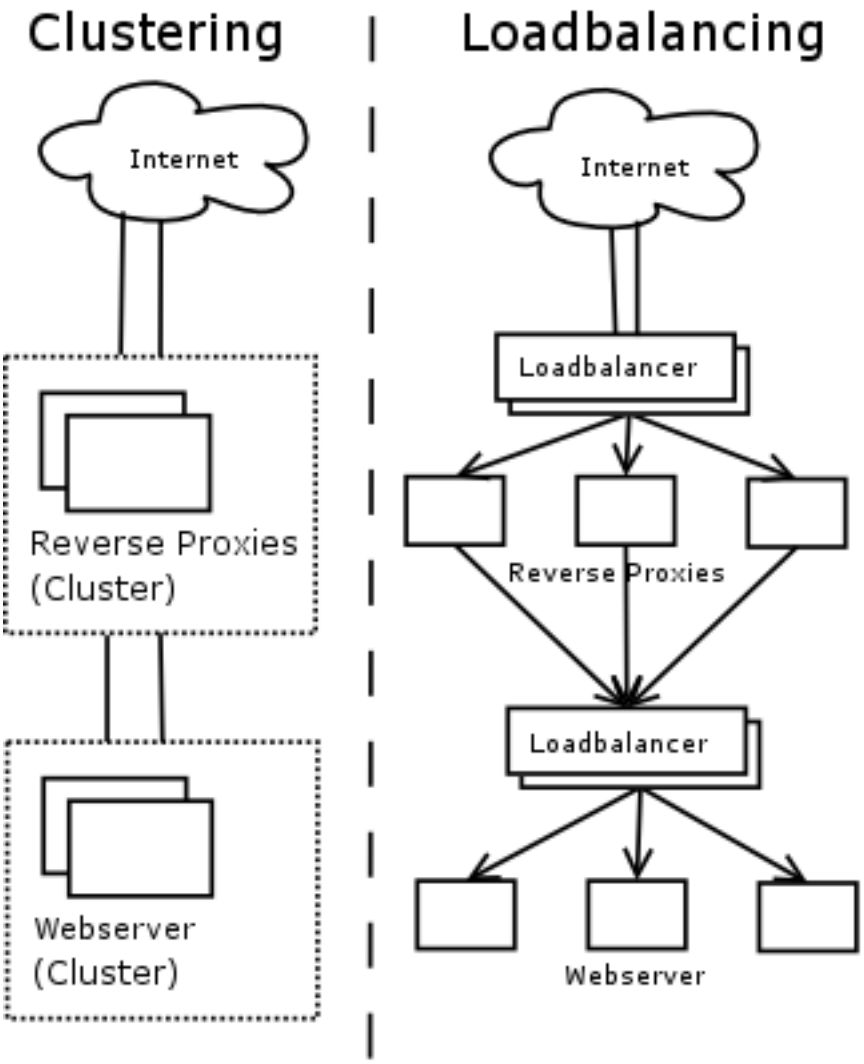
Für mittlere bis hohe Sicherheitsanforderungen muss netzwerktechnisch (typischerweise durch Aufbau einer durch Firewalls geschützten Secure Zone) ausgeschlossen werden, dass ausser dem Reverse Proxy noch andere Clients direkte Requests an den Webserver absetzen können.

Für hohe Sicherheitsanforderungen kann neben dem Schutz des Webservers auf Basis des Netzwerks auch noch eine gegenseitige Authentisierung von Reverse Proxy und Webserver mittels Serverzertifikaten verwendet werden, d.h. der Zugriff auf den Webserver erfolgt per HTTPS mit Client-Authentifizierung. Dies bedeutet wegen des Protokoll-Overheads und der Verschlüsselung aber einen Performanceverlust gegenüber der unverschlüsselten Lösung.

Wie kann Fehlertoleranz und hohe Verfügbarkeit sichergestellt werden?

Ausfallsicherheit kann grundsätzlich nur durch Redundanz auf Hardware-Ebene erreicht werden.

Dabei sind mehrere Varianten denkbar. Der Vorteil des Loadbalancings gegenüber einem Fail-Over-Cluster besteht darin, dass die vorhandenen Resourcen besser ausgenutzt werden können, was zu einer besseren Performance führt. Loadbalancing kann jedoch zu komplizierten Netzwerk-Topologien und subtilen Fehlern führen, die schwierig zu analysieren sind.



Bei allen Varianten ist allerdings darauf zu achten, dass kein *Single Point of Failure* entsteht. Dies beinhaltet alle wichtigen Komponenten, inkl. Internetanbindung und Stromversorgung.

Bekanntlich ist eine Kette nur so stark wie ihr schwächstes Glied.

Wo finde ich weitere Informationen zum Thema Secure Reverse Proxy?

Links:

- Apache Webserver [http://httpd.apache.org/]
- Nevis-Web der Firma AdNovum [http://www.nevis-web.com/]
- Sun Java System Web Proxy Server [http://wwws.sun.com/software/products/web_proxy/ds_web_proxy.html]
- OpenLDAP Directory Server [http://www.openldap.org/]

Die Mitarbeiter [http://www.indato.ch/mitarbeiter.html] der indato GmbH verfügen über langjährige Erfahrungen im Bereich Secure Reverse Proxy und Single-Sign-On. Gerne unterstützen wir Sie in diesen Belangen.