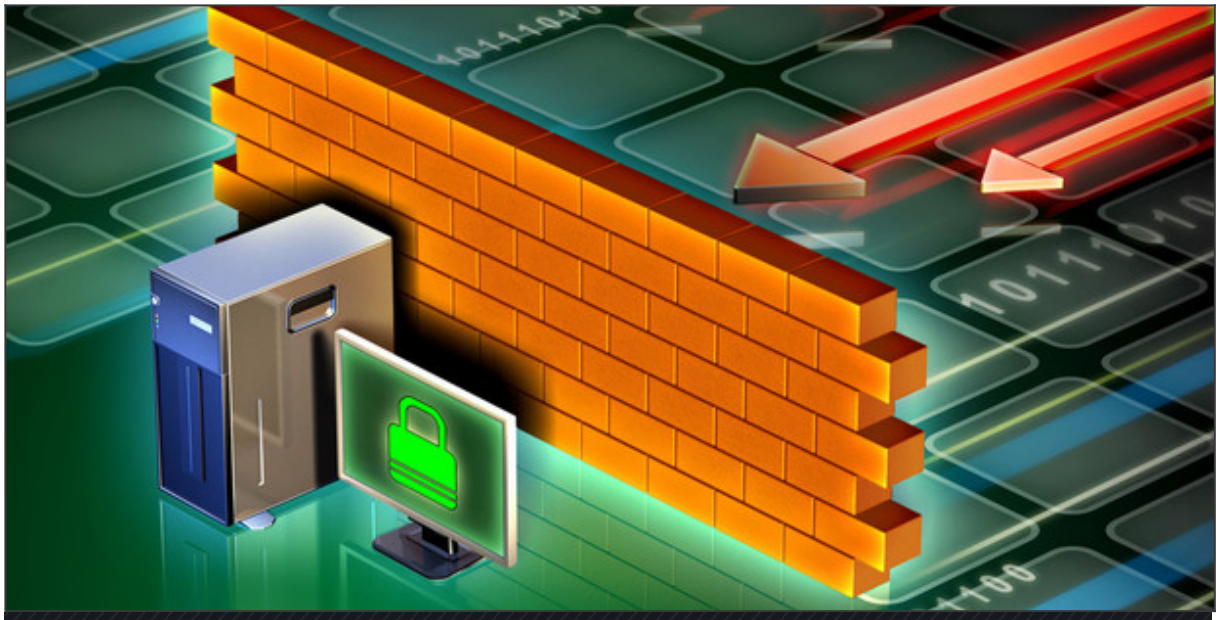
Schwachstellen-Test am Beispiel der Web Application Firewall ModSecurity

Auch Sicherheitssoftware ist unsicher

10.02.14 | Autor / Redakteur: Valeri Milke, Sascha Böhm und Prof. Dr. Hartmut Pohl / Stephan Augsten



Web Application Firewalls bieten nur dann guten Schutz, wenn sie selbst auf ihre Sicherheit hin geprüft werden. (Bild: Andrea Danti - Fotolia.com)

Angriffe auf die IT sind in der Regel besonders erfolgreich, wenn sie sich die Sicherheitsanfälligkeit einer Anwendung zunutze machen. In diesem Beitrag zeigen wir am konkreten Beispiel der Web Application Firewall (WAF) ModSecurity, dass auch Security Software ausnutzbare Sicherheitslücken enthalten kann und dadurch angreifbar ist.

Kaum eine Software ist vor Schwachstellen gefeit, Sicherheitsanwendungen wie Firewalls oder Intrusion-Detection- und –Protection-Systeme bilden da keine Ausnahme. Dies zeigten mehrstufige, systematische Security Tests an einer ganzen Reihe von Sicherheitsprodukten.

Web Application Firewalls (WAFs) funktionieren mit Black- und Whitelists und filtern den http-Transfer zwischen Server und Client. Der Vorteil gegenüber herkömmlichen Firewalls ist, dass eine WAF nicht auf den unteren Netzwerkebenen filtert, sondern auf der Anwendungsschicht (Layer 7 nach dem ISO/OSI-Referenzmodell).

Herkömmliche Firewalls arbeiten in der Regel auf Layer 3 (Vermittlungsschicht) oder Layer 4 (Transportschicht), wodurch sie ankommende Anfragen nach IP-Adressen oder Ports filtern können. Eine WAF hingegen untersucht auch die Inhalte der ankommenden Pakete. Dadurch ist sie dazu in der Lage, Angriffe wie SQL-Injections und Cross Site Scripting abzuwehren, die von herkömmlichen Firewalls nicht erkannt werden können.

Web Application Firewalls untersuchen ausschließlich http-Pakete und dienen daher dazu, die Ausnutzung von Sicherheitslücken speziell in Webanwendungen zu verhindern. Hierzu bedienen sie sich definierter Regeln, die mit regulären Ausdrücken arbeiten, um nach dem Black- und Whitelisting-Ansatz bösartige http-Anfragen zu blockieren

▼ Weiterführende Literatur
▶ Weiterführende Literatur

BSI: Sicherheit von Webanwendungen (Bonn, 2006)

Intermoves GmbH: Web-Applikation Firewall – Grundlagen und Marktübersicht (2013)

OWASP: Best Pratices Guide WAF (2008)

Trustwave: ModSecurity Open Source Web Application Firewall (Chicago, 2013)

Sicherheitssoftware – ein zweischneidiges Schwert

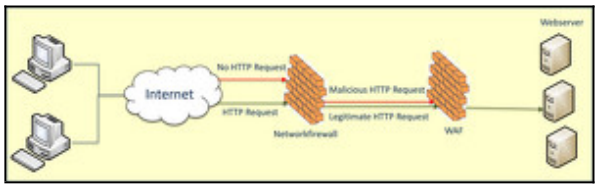
Firewalls erhöhen einerseits das Sicherheitsniveau, indem sie den Traffic filtern und damit Server, Computer und Webanwendungen vor Angriffen schützen. Andererseits jedoch müssen sie selbst frei von Sicherheitslücken sein, damit sie nicht selbst zum Angriffsziel werden.

Hinzu kommt, dass Security Software zeitnah gepatcht und generell korrekt konfiguriert werden muss, um das Sicherheitsniveau maßgeblich erhöhen zu können. Damit solche Lösungen nicht zum Einfallstor werden, sollten sie im Rahmen eines mehrstufigen systematischen Security Test Process auf Sicherheitslücken hin untersucht werden.

Funktionsweise von Web Application Firewalls

Durch die Verlagerung von Anwendungen auf Server im Internet, denken wie zum

Beispiel ans Einkaufen oder ans Reservieren von Sitzplätzen im Kino, steigt das Bedrohungspotenzial auf Webanwendungen. Zu den verbreitetsten Angriffen im Internet gehören SQL Injections, Cross Site Scripting und Session Hijacking. Um solche Attacken zu identifizieren, muss der Inhalt einer http-Anfrage untersucht werden.



WAFs stellen eine erweiterte Firewall-Funktionalität bereit, indem http-Pakete auf Anwendungsebene untersucht werden, um Angriffe zu erkennen und ggf. Anfragen zu verwerfen, noch bevor die Anfrage die Webanwendung erreicht. Wie in der Abbildung zu erkennen ist, wird die WAF hinter der Netzwerk-Firewall installiert. Bevor die Daten zum Server gelangen, müssen sie also zusätzlich die WAF passieren.

Die WAF prüft den Traffic auf ungefährliche und gefährliche http-Anfragen hin. Erst wenn diese Prüfung positiv (kein Angriff) ausgefallen ist, werden die Daten zum Webserver weitergeleitet. Dieses Grundprinzip gilt für alle Arten von WAFs; jede Art filtert die http-Anfragen, bevor diese weiter zum Server geleitet werden.

▼ Weiterführende Literatur
▶ Weiterführende Literatur

BSI: Sicherheit von Webanwendungen (Bonn, 2006)

Intermoves GmbH: Web-Applikation Firewall – Grundlagen und Marktübersicht (2013)

OWASP: Best Pratices Guide WAF (2008)

Trustwave: ModSecurity Open Source Web Application Firewall (Chicago, 2013)

Man unterscheidet drei Typen von Web Application Firewalls:

- Als **Hardware Appliance** wird die WAF wie eine klassische Hardware Firewall vor dem Webserver in das lokale Netz eingebunden. Dies geschieht im Regelfall direkt hinter der Netzwerk-Firewall.
- Arbeitet die WAF als **Reverse Proxy Server**, dann dient sie als Verbindungsschnittstelle für den Server. Sämtliche Anfragen laufen über die WAF

und werden an den Server weitergeleitet. Die wahre Adresse des Servers bleibt geheim.

- In Form eines **Software-Plug-In** wird die WAF direkt auf den Webserver installiert.

Behebung häufiger Sicherheitslücken durch eine WAF

Während eine WAF am besten vor Sicherheitslücken in Web-Anwendungen schützt, eignen sich Netzwerk-Firewalls besser, wenn potenzielle Sicherheitslücken beispielsweise andere Protokolle als http betreffen. Im Folgenden schauen wir uns einige bekannte Angriffsformen an und erläutern, wie eine WAF ihnen entgegenwirkt.

Cross-Site-Scripting (XSS)

Cross-Site-Scripting wird dazu verwendet, Identitätsdaten eines Angriffsopfers auszuspähen, um damit zum Beispiel an die Daten des Opfers zu gelangen. Eine Möglichkeit ist, sich mit Hilfe von Cookies in die Sitzung einzuwählen und somit die Login-Daten des Opfers abzugreifen. Dies ist kann dann geschehen, wenn das Opfer einen Link anklickt, der Cookies zum Angreifer schickt.

Um zu testen, ob eine XSS-Attacke möglich ist, kann ein einfacher JavaScript-Befehl mitgeschickt werden.

<script type="text/javascript">alert("XSS");</script>

Hier wird zwar nur ein Popup-Fenster geöffnet, das den Text „XSS“ beinhaltet. Dies zeigt jedoch, dass Cross-Site-Scripting prinzipiell möglich ist, da JavaScript zugelassen ist. Eine WAF kann die Ausnutzung dieser Art von Sicherheitslücken durch die Kontrolle der Eingaben schließen.

Wenn es zum Beispiel nicht erlaubt ist, die Zeichen < und > zu verwenden, wird diese Anfrage geblockt. Eine Möglichkeit dieses zu umgehen wäre zum Beispiel eine Nutzung von URL Maskierungen – und zwar für „<“ z.B. „%3C“ und für „>“ etwa „%3E“. Diese Zeichen werden vom HTML Interpreter dann als „<“ und „>“ erkannt aber möglicherweise nicht von der Firewall geblockt.

▼ Weiterführende Literatur
▶ Weiterführende Literatur

BSI: Sicherheit von Webanwendungen (Bonn, 2006)

Intermoves GmbH: Web-Applikation Firewall – Grundlagen und Marktübersicht (2013)

OWASP: Best Pratices Guide WAF (2008)

Trustwave: ModSecurity Open Source Web Application Firewall (Chicago, 2013)

SQL Injection

Ähnlich wie beim XSS wird bei einer SQL-Injektion in die http-Anfrage eine manipulierende SQL-Query eingeschleust. Hierbei handelt es sich um SQL-Befehle, die das Ziel verfolgen, Datenbanken zu verändern oder Informationen zu beschaffen. SQL-Injections sind dann erfolgreich, wenn es gelingt, Daten in den SQL-Interpreter einzugeben.

Eine Möglichkeit SQL-Injectionen vorzubeugen, ist das Blacklisting-Verfahren. Hierbei werden bestimmte Zeichen oder Zeichenketten abgeblockt. Zusätzlich zu einer WAF erschweren URL-Encryption oder Hidden-Form-Parameter-Protection es dem Angreifer, Angriffsziele zu identifizieren, da die URL des Opfers für den Angreifer unkenntlich ist bzw. die Parameter nicht identifizierbar sind.

Je nach Art des Eingabefeldes bietet sich auch Whitelisting an. Dabei werden nur zugelassene Eingaben angenommen und alle anderen verworfen. Auch hier wird versucht, mit Hilfe von URL-Maskierungen die Whitelist zu umgehen – etwa mit einem Tabulator oder durch Verwendung verschiedener Zeichen als URL-Maskierung.

Cross-Site-Request-Forgery

Beim Cross-Site-Request-Forgery (CSRF) wird einem Dritten ohne sein Wissen ein http-Request untergeschoben, womit eine Aktion auf der Webanwendung durchgeführt wird. Dies funktioniert im Regelfall nur dann, wenn der Dritte bei der Webanwendung angemeldet ist. Der Grund für diese Sicherheitslücke ist die Statuslosigkeit des http-Protokolls.

Es gibt verschiedene Möglichkeiten, dem Betroffenen die manipulierte URL unterzuschieben. So kann auf eine andere Sicherheitslücke (wie etwa XSS) zurückgegriffen werden, auf eine andere Webseite, auf der der Betroffene einen Button anklickt, oder auch auf eine E-Mail. Auch die Verbreitung über soziale Netzwerke oder Blogs ist möglich.

Um zu verhindern, dass die URL direkt als Risiko bewertet wird, kann sie etwa durch einen Kurz-URL-Dienst verschleiert werden.

Ein harmloses Beispiel ist, wenn ein Anwender bei Wikipedia eingeloggt ist und diesen Link nun anklickt:

http://de.wikipedia.org/w/index.php?title=Spezial:Userlogout

Damit würde der Anwender nun bei Wikipedia ausloggt. Gefährlicher wäre es beim Online Banking oder anderen Transaktionen übers Internet. Hier besteht zum Beispiel das Risiko, dass ungewollt Geld von einem Konto auf ein anderes überwiesen wird.

▼ Weiterführende Literatur
▶ Weiterführende Literatur

BSI: Sicherheit von Webanwendungen (Bonn, 2006)

Intermoves GmbH: Web-Applikation Firewall – Grundlagen und Marktübersicht (2013)

OWASP: Best Pratices Guide WAF (2008)

Trustwave: ModSecurity Open Source Web Application Firewall (Chicago, 2013)

Um hiergegen etwas zu unternehmen, empfiehlt es sich, Page-Token zu verwenden. Das heißt, dass zusätzlich zu jeder Sitzung jede aufgerufene Seite ihre eigene einzigartige und zufällige ID bekommt und somit die Benutzung des Opferlinks nicht mehr möglich ist.

Eine URL mit Page Token würde zum Beispiel so aussehen:

http://www.sehr_sichere_sessions.de/page.php?pagetoken:25638789342/index.html

Die Web Application Firewall ModSecurity

Eine bekannte WAF ist ModSecurity, die als Plug-In direkt auf dem Webserver installiert wird. ModSecurity unterstützt Apache, IIS und Nginx und ist ein Open-Source-Produkt. Sie arbeitet mit Black- und White-Listen, die durch Regeln genau konfiguriert werden. Die Regeln bedienen sich der regulären Ausdrücke.

ModSecurity besitzt nach der anfänglichen Installation nur eine geringe Anzahl an Regeln. Jedoch wird auf der Webseite auf zwei verschiedene Regelpakete hingewiesen, welche zusätzlich installiert werden müssen: Ein Open Source Regelpaket von OWASP und eine kommerzielle Variante von Trustwave Spiderlabs.

Laut ModSecurity könnten aber auch beide Regelpakete gleichzeitig verwendet werden. Weitere Informationen zu ModSecurity befinden sich auf der Projekt-Homepage www.modsecurity.org.

„XML External Entity“-Lücke in ModSecurity

ModSecurity ist ein gutes Beispiel dafür, dass WAFs zwar durchaus eine Maßnahme sind, um die Ausnutzung von Sicherheitslücken in Webanwendungen zu verhindern. Allerdings können sie auch selbst Sicherheitslücken enthalten. So wurde im Jahr 2013 eine „XML External Entity“-Anfälligkeit in der Version 2.7.2 identifiziert.

Die Schwachstelle entsteht dadurch, dass die XML-Bibliothek libxml2 das Einbinden externer Dateien erlaubt. Die Abbildung zeigt, wie eine normale XML-Anfrage verarbeitet wird. Hierbei untersucht die Web Application Firewall die XML-Anfrage, parst diese und leitet sie weiter an den Apache-Server.

Dieser antwortet nun wie gewohnt auf die Anfrage. Erhält der Server jedoch zum Beispiel folgende Anfrage, dann wird auf eine externe Datei verwiesen:

<?xml version=“1.0“ encoding=“ISO-8859-1“ß>

<!DOCTYPE foo[

<!ENTITY xxe SYSTEM “file:///dev/random” >

]>

<foo>&xxe;</foo>

In diesem Fall ist es die externe Anwendung /dev/random zur Generierung von Zufallsdaten. Die libxml2-Bibliothek versucht nun, auf diese Datei zuzugreifen. Sie erhält aber keine Antwort, da /dev/random ein Pseudo-Device ist, das Zufallszahlen mit Hilfe von User-Eingaben generiert. Da nichts eingegeben wird, werden sämtliche Zugriffe blockiert.

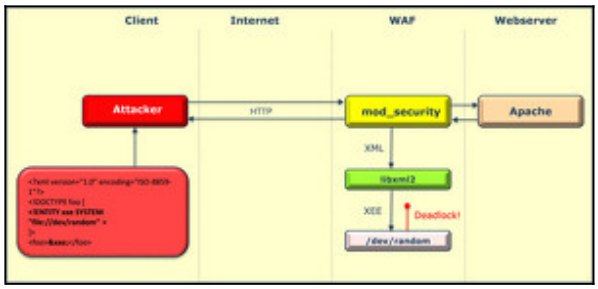
▼ Weiterführende Literatur
▶ Weiterführende Literatur

BSI: Sicherheit von Webanwendungen (Bonn, 2006)

Intermoves GmbH: Web-Applikation Firewall – Grundlagen und Marktübersicht (2013)

OWASP: Best Pratices Guide WAF (2008)

Trustwave: ModSecurity Open Source Web Application Firewall (Chicago, 2013)



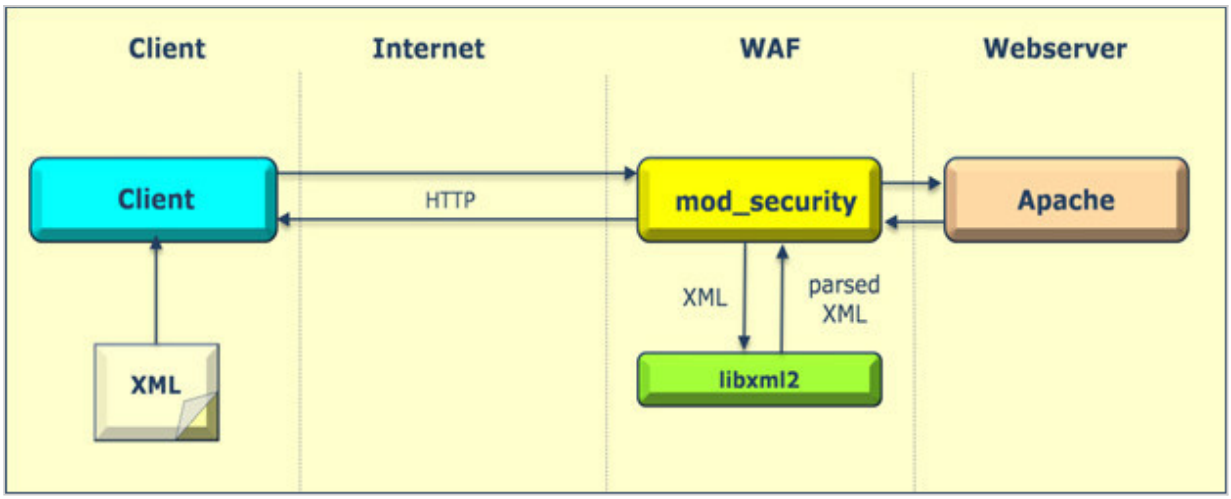
Dementsprechend entsteht ein Deadlock, wie in dieser Abbildung dargestellt. Durch diesen Deadlock warten nun alle Prozesse einschließlich des Apache-Prozesses von ModSecurity auf eine Antwort von /dev/random. Da Apache aber mehrere Clients gleichzeitig bearbeitet, erstellt der Server in diesem Fall einen weiteren Prozess.

Die Anfrage wird nun solange geschickt, bis Apache die maximal mögliche Anzahl an Prozessen erreicht hat. Danach stürzt Apache ab und reagiert nicht mehr auf Anfragen. Daraus wird deutlich, dass Sicherheitssoftware zwar nützlich ist, aber selbst Anfälligkeiten enthalten kann.

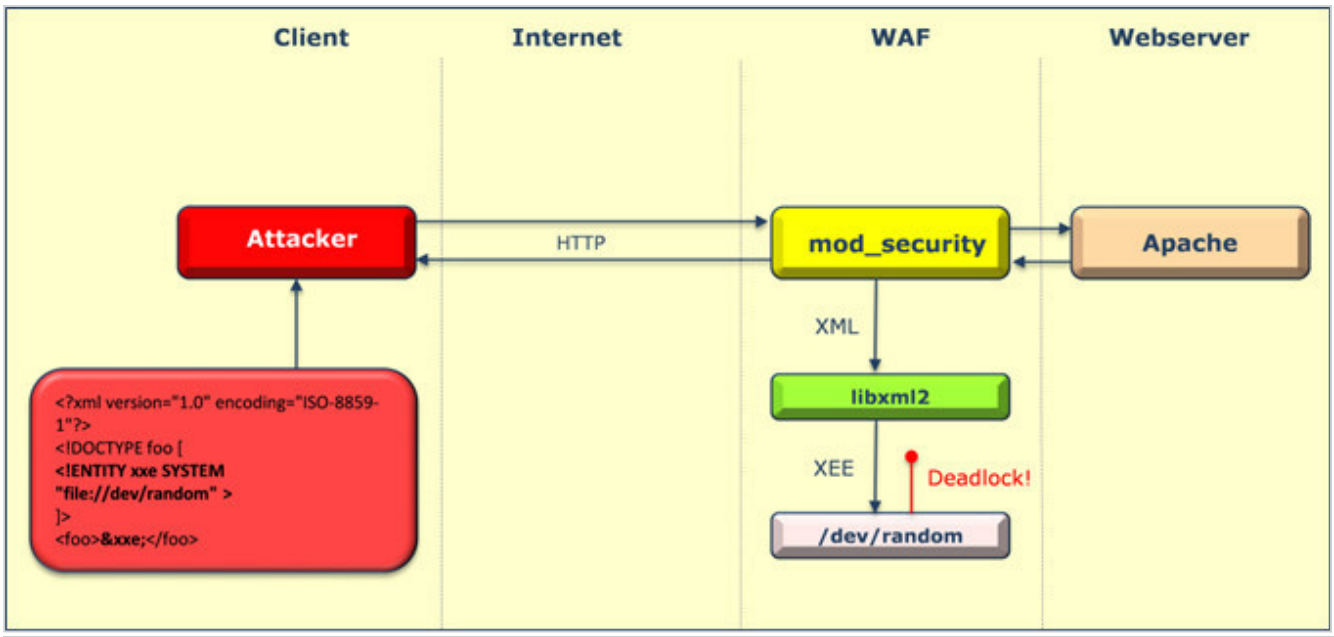
Security Software sollte also generell auf Schwachstellen hin geprüft werden. Bekannte Methoden hierfür sind beispielsweise Threat Modeling, statische Quellcode-Analyse, Penetration Testing und dynamische Analysen (bspw. Fuzzing). Erst durch einen umfassenden Security Test Process wird der Patch-Aufwand nach Auslieferung von Sicherheitsprodukten minimiert.

Über die Autoren

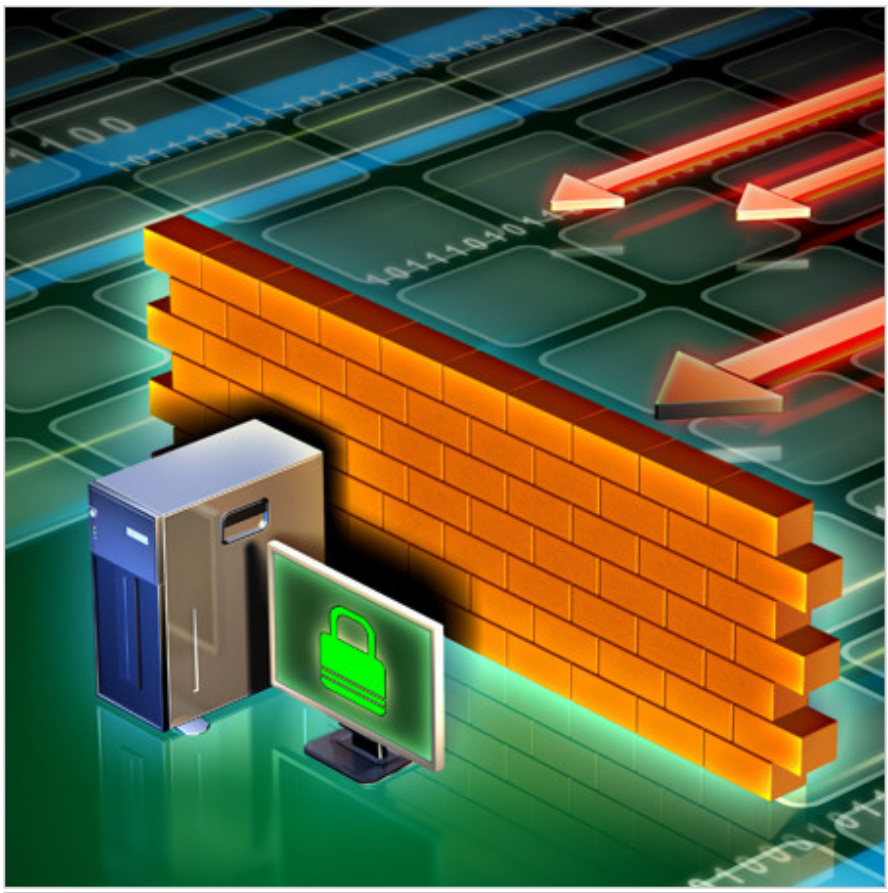
Prof. Dr. Hartmut Pohl ist Geschäftsführender Gesellschafter der softScheck GmbH in Köln, Valeri Milke ist als Consultant bei softScheck tätig, Sascha Böhm arbeitet beim gleichen Unternehmen als Junior Consultant.



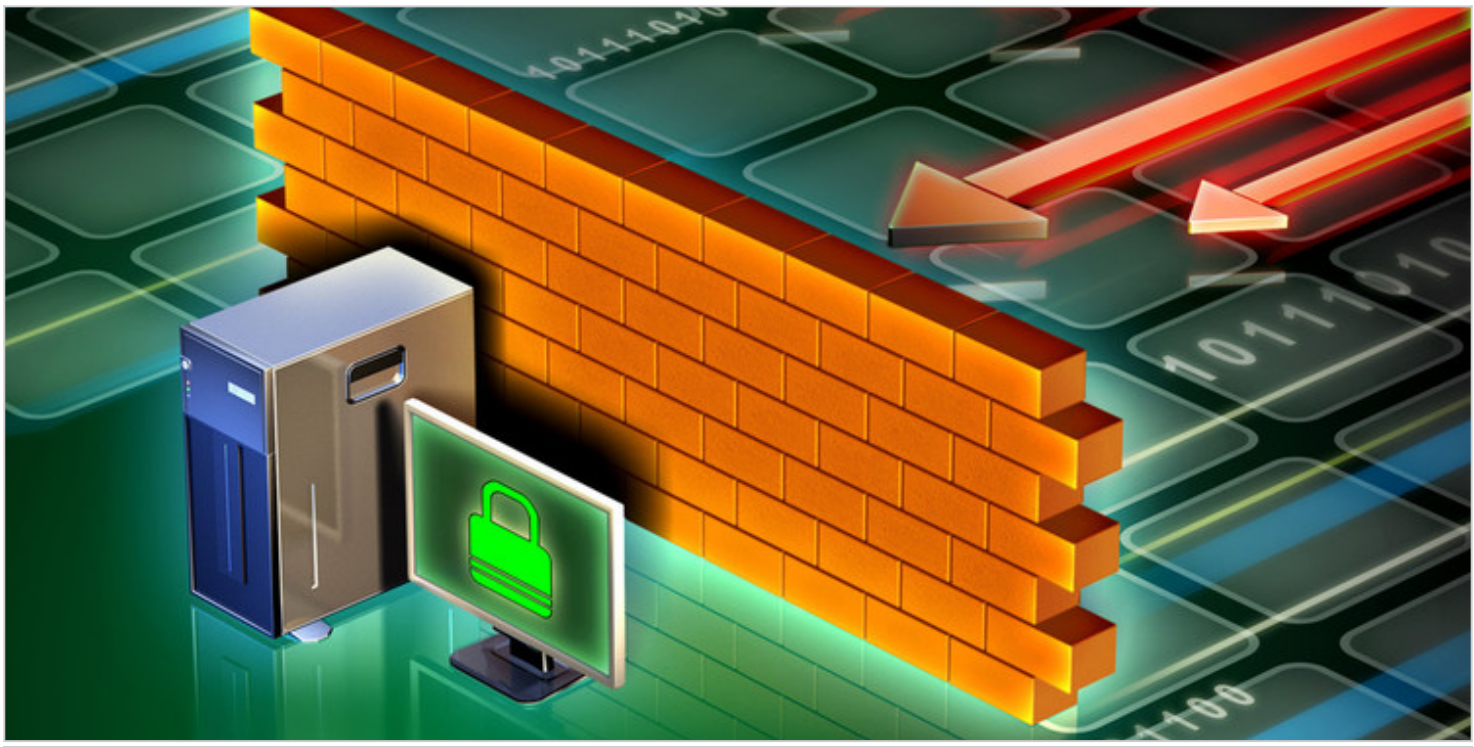
XML-Verarbeitung durch die Web Applicaiton Firewall. (Bild: SoftScheck)



Provozierter XML-Deadlock (Bild: SoftScheck)



Web Application Firewalls bieten nur dann guten Schutz, wenn sie selbst auf ihre Sicherheit hin geprüft werden. (Bild: Andrea Danti - Fotolia.com)



Web Application Firewalls bieten nur dann guten Schutz, wenn sie selbst auf ihre Sicherheit hin geprüft werden. (Bild: Andrea Danti - Fotolia.com)