

## Sonderdruck aus LANline 11/03

### SOAP-SECURITY-GATEWAYS

## Gesicherte Tore

Sicherheitsmodelle und -standards für Webservices sind seit Monaten in Arbeit, gilt doch fehlende Sicherheit als eines der Haupthindernisse für deren weite Akzeptanz und kommerziellen Einsatz. Spezifikationen wie XML Digital Signature und SAML liegen mittlerweile als verabschiedete offene Standards vor. Doch wie kann man mit diesen Technologien tatsächlich eigene Anwendungen absichern? Dieser Artikel stellt SOAP-Security-Gateways als integrierbares, standardbasiertes Konzept für die Sicherheit von Webservices vor.

Das große Potenzial von Webservices für die Integration von Anwendungen, sowohl Inhouse als auch zwischen Unternehmen wurde schon oft beschworen. Dennoch sind viele Anwender noch durch die Unübersichtlichkeit der Technologien und die große Zahl von Standards verunsichert. Hinzu kommen starke Sicherheitsbedenken, denn das "HTTP-Tunneln" von SOAP-Nachrichten, mit denen sich Webservices-Komponenten nutzen

lassen, setzt herkömmliche Firewalls als Schutzmechanismen außer Kraft, sobald diese überhaupt HTTP-Verkehr erlauben. Diese Bedenken wiegen umso schwerer, als der SOAP-Standard selbst keinerlei Sicherheitsmechanismen aufweist. Schießt die große Offenheit von Webservices also über das Integrationsziel hinaus?

**SICHERHEITSSTANDARDS** Sicherheitsanstrengungen im XML- und Webservices-Kontext gibt es seit einiger Zeit. Welche Technologie aber steht nur auf dem Papier, und welche tatsächlich zur Verfügung? Und wie kann man damit Webservices praktisch sichern? Vor dem Beantworten dieser Fragen, soll zunächst kurz das Zusammenspiel der wichtigsten Sicherheitsstandards (WS-Security, SAML (Security Assertion Markup Language) und XML Digital Signature) an einem konkreten Beispiel erläutert werden. Als Ausgangspunkt

dient die leicht vereinfachte SOAP-Nachricht in Abbildung 1, die den Aufruf einer Funktion "newOrder" mit drei Parametern darstellt. Beim Versand dieser Nachricht an einen Webservice sind zahlreiche Angriffe auf die übertragene Nachricht selbst oder auf den empfangenden Dienst denkbar, so das Mithören, Verfälschen, Umleiten oder Unterdrücken der Nachricht oder das Einbringen eigener Nachrichten durch einen nichtautorisierten Sender.

Da eine solche Nachricht prinzipiell über verschiedene Zwischenstationen zum Empfänger gelangen kann, reicht die Sicherung der Transportverbindung zum Empfänger mit SSL/TLS allein nicht aus. Vielmehr muss jede Nachricht selbst alle Sicherheitsinformation enthalten, die

```
<Envelope>
  <Header/>
  <Body>
    <newOrder>
      <string>Documents</string>
      <intVal>350</intVal>
      <string0>urgent</string0>
    </newOrder>
  </Body>
</Envelope>
```

Abbildung 1. Eine einfache SOAP-Nachricht

```
<Assertion xmlns="urn:oasis:names:tc:
SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="0"
  AssertionID="4711"
  Issuer="Xtradyne Webservices DBC"
  IssueInstant="2003-08-19T14:54:43">
  <Conditions NotBefore="
2003-08-19T14:54:43"
  NotOnOrAfter="2003-08-
19T15:04:43"/>
  <AuthenticationStatement
    AuthenticationMethod="urn:oasis:
names:tc:
SAML:1.0:am:unspecified"
    AuthenticationInstant="2003-08-
19T14:54:43">
    <Subject>
      <NameIdentifier>Raccoon
      </NameIdentifier>
    </Subject>
    </AuthenticationStatement>
  </Assertion>
```

Abbildung 2. Eine SAML-Assertion

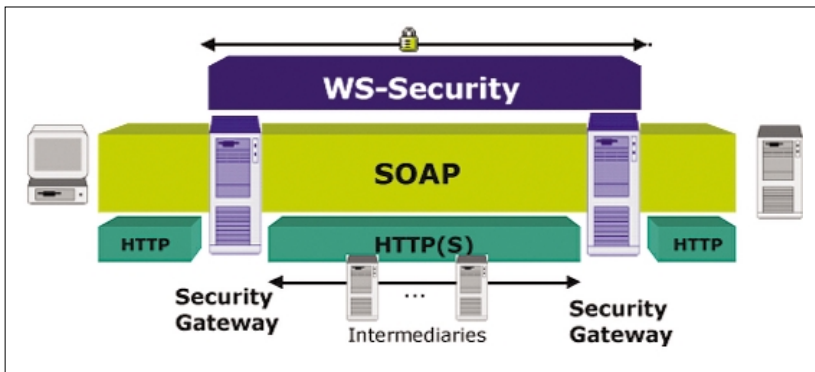


Bild 3. SOAP-Security-Gateways eignen sich zum Absichern von Webservices

Mechanismen wie Authentisierung- oder Autorisierungsdienste benötigen, um Sicherheitsentscheidungen treffen zu können. Dies wirft zwei Fragen auf: In welchem Format lässt sich die Sicherheitsinformation interoperabel übertragen? Und wie kann sie zuverlässig an SOAP-Nachrichten gebunden werden, um auf diese anwendbar zu sein?

Die erste dieser beiden Fragen beantwortet die Security Assertion Markup Language, ein XML-Standard des OASIS-Konsortiums, der Sicherheitsinformation als "Assertion" ausdrückt. Hat sich ein Sender beispielsweise bei einem Authentisierungsdienst ausgewiesen, könnte dieser eine Authentication Assertion ausstellen, wie Abbildung 2 zeigt.

Eine solche Assertion drückt aus, dass der Aussteller der Assertion das bezeichnete Subjekt authentisiert hat. Andere Sicherheitsdienste können diese Aussage als Nachweis der Identität akzeptieren, wenn sie dem Aussteller das entsprechende Vertrauen entgegenbringen. Um nun diese Assertion untrennbar mit einer SOAP-Nachricht zu verbinden und zudem sowohl ihre Integrität wie Authentizität zu garantieren, kommen digitale Signaturen zum Einsatz. Wie und wo diese anzubringen sind, und wo ferner die Assertion in der Nachricht unterzubringen ist, wird von zwei weiteren Sicherheitsstandards beantwortet, nämlich der Webservices-Security-Spezifikation (WS-S) von OASIS und dem XML-Digital-Signature-Standard der W3C.

Abbildung 4 zeigt so eine kombinierte SOAP-Nachricht, bei der im Kopf der Nachricht ein spezieller Sicherheitsabschnitt `<wsse:Security>` eingebettet

wurde, der die Assertion sowie eine digitale Signatur enthält. Diese erstreckt sich über die Assertion sowie den Nachrichtenrumpf, was durch die beiden `<Reference>`-Elemente der Signatur zum Ausdruck kommt. Der Empfänger kann die Gültigkeit der Signatur prüfen, um zu entscheiden, ob die Assertion unverändert ist und ob sie tatsächlich zu der empfangenen Nachricht gehört. Da die Signatur mit einem Public-Key-Verfahren angebracht wurde, lässt sich zudem feststellen, ob die Assertion von einem vertrauenswürdigen Aussteller stammt. Falls ja, dient die Assertion nun beispielsweise als Grundlage für Zugriffszwangsentscheidungen, oder die angegebene Identität findet für Einträge in Audit-Logs Verwendung.

Für den Einsatz der beschriebenen Technologien und Standards kommen prinzipiell drei Architekturalternativen in Betracht, nämlich plattformbasiert (die Sicherheitsmechanismen in Application-Servern), anwendungsbasiert (die Verwendung von Sicherheits-APIs) und protokollbasiert (Sicherheits-Gateways).

**PLATTFORMBASIERTE SICHERHEIT** Serverseitige Webservice-Plattformen (Application-Server) stellen die Laufzeitumgebung für Webservices dar. Application-Server-Produkte bieten eine Reihe eigener Sicherheitsmechanismen, beispielsweise Transportverschlüsselung mit SSL, Authentisierung, Audit etc. Auch neuere Standards wie SAML und WS-Security finden zunehmend Verbreitung. Neben der allerdings oft nur grobkörnigen Sicherheits-

funktionalität liegen die Nachteile dieses Ansatzes in der Herstellerabhängigkeit und der einseitig serverbezogenen Architektur. Diese Nachteile zeigen sich besonders beim gemeinsamen Einsatz verschiedener Application-Server: Jedes Produkt erfordert eigene Managementwerkzeuge mit eigenen Managementkonzepten, was die zentrale Formulierung und Durchsetzung einer einheitlichen Security Policy sehr erschwert. Das Einbetten von Sicherheit in die Laufzeitumgebung der Application-Server impliziert außerdem, dass Sicherheitsprüfungen erst dann erfolgen, wenn tatsächlich Nachrichten im Server eingetroffen sind. Das bedeutet wiederum, dass eine ausschließlich plattformbasierte Sicherheit potenziell gefährliche Nachrichten nicht schon in vorgelagerten Subnetzen (DMZ) prüft und abweist. Schließlich verlangt eine solche Architektur, dass die Sender von Nachrichten alle benötigten Credentials bereits in der erwarteten Form vorlegen.

**ANWENDUNGSBASIERTE SICHERHEIT** Plattformhersteller bieten häufig zusätzlich APIs, mit denen Serviceimplementierungen selbst bestimmte Sicherheitsprüfungen durchführen können. Microsoft liefert beispielsweise für seine Dotnet-Plattform "Web Services Enhancements"-Bibliotheken, mit denen sich unter anderem XML-Digitalsignaturen prüfen lassen, sodass Anwendungen selbst für den benötigten Schutz sorgen. Der Nachteil bei diesem Vorgehen liegt in dem erhöhten Entwicklungsaufwand und dem häufig fehlenden Sicherheitsbewusstsein bei Entwicklern. Sehr nachteilig sind zudem die Verquickung von funktionalem Anwendungscode mit Aufrufen in das Sicherheits-API, oder das Einbetten von Security Policies im Code.

**SICHERHEITS-GATEWAYS FÜR WEBSERVICES** Sicherheits-Gateways sind Komponenten, die Sicherheitsfunktionen auf Protokollebene durchsetzen, bevor Nachrichten die Anwendungen erreichen. SOAP-Gateways analysieren Format und Inhalt von SOAP-Nachricht-

ten unter Berücksichtigung der genannten Sicherheitsstandards und bieten so eine vollständige AAAA-Funktionalität (Authentisierung, Autorisierung, Audit, Administration) bei transparenter Integration. Solche Gateways gehen im Funktionsumfang weit über Paketfilter-Firewalls hinaus, die lediglich einfache Formatprüfungen vornehmen. Unternehmens-Firewalls können so um Sicherheit auf der Anwendungsebene ergänzt werden, beispielsweise durch SAML-basierte Authentisierung, XML-Schemavalidierung und inhaltsbasierte Filterung von SOAP-Nachrichten sowie rollenbasierte Zugriffskontrolle. Gleichzeitig erfolgt ein zentrales Policy-Management über die Grenzen von Plattformherstellern hinweg, was die Durchsetzung einheitlicher Policies stark vereinfacht.

Bild 3 zeigt schematisch, wie sich solche Gateways auf der Sender- und Empfängerseite kombinieren lassen, um trotz mehrerer Zwischenstationen End-to-End-Sicherheit oder genauer Gateway-zu-Gateway-Sicherheit zu erreichen. In einem solchen "federated trust"-Szenario fügt das Gateway auf der Senderseite die benötigte Sicherheitssinformation im Fluge in Standardsyntax in ausgehende Nachrichten ein. Wenn das Gateway auf der Empfängerseite als vertrauenswürdig gilt, kann es beispielsweise als Authentisierungsautorität akzeptiert werden. Zudem ist ein ausgehendes Gateway in der Lage, das "credentials mapping" vorzunehmen, also beispielsweise Rollen einer Unternehmensinfrastruktur in ein beim Empfänger bekanntes Rollensystem umzusetzen.

Zum Schutz von Webservices mit SOAP-Gateways gibt es zurzeit zwei Varianten am Markt, nämlich Appliances und reine Softwarelösungen. Appliances werben mit besserer Performance durch spezielle XML-Hardware und wegen der Verpackung in einem einzelnen Gerät, vereinfachter Installation und Verwaltung. Das Management von Sicherheits-Policies sowie das Verwalten der Gateway-Komponente erfordern jedoch prinzipiell bei beiden Varianten den gleichen Aufwand. Auch bei der Performance stehen leistungsfähige,

```
?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
      <Assertion>...</Assertion>
      <Signature xmlns=http://www.w3.org/2000/09/xmldsig#>
        <SignedInfo>
          <CanonicalizationMethod
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          <SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <Reference URI="#xmlns(soap-env=...)xpointer(/soap-env:Envelope/
            soap-env:Body)">
            <Transforms>
              <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
                c14n-20010315"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>BVk207AKkaRfdlaxfVXWvNQRHHk=</DigestValue>
          </Reference>
          <Reference URI="#xmlns(soap-env=...)xmlns(wsse=...)xmlns(saml=...)xpointer
            (/soap-env:Envelope/soap-env:Header/wsse:Security/saml:Assertion)">
            <Transforms>
              <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
                c14n-20010315"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <DigestValue>kuR...S06giBxg=</DigestValue>
          </Reference>
        </SignedInfo>
        <SignatureValue>sUh2eB...GjYuo=
      </SignatureValue>
      <KeyInfo>
        <X509Data>
          <X509Certificate>...</X509Certificate>
        </X509Data>
      </KeyInfo>
    </Signature>
  </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <getAllOrders>
  </getAllOrders>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Abbildung 4. Eine SOAP-Nachricht mit Assertion und Signatur

hochoptimierte Softwarelösungen der Performance von Appliances nicht nach. Hardwarebasierte Appliances haben allerdings den Nachteil, dass sie im Allgemeinen längere Produktzyklen erfordern. Bei etwas komplexeren Deployment-Szenarien lassen sich die einzelnen Komponenten eines Software-Gateways (Bastion Host-Komponente, Policy-Server, Managementkonsole) zudem flexibler auf verschiedene Subnetze verteilen als dies bei einer abgeschlossenen Appliance möglich ist.

**FAZIT** Für den notwendigen Schutz von Webservices stehen heutzutage passende Standards und Technologien bereit. Dabei stellen reine Softwareimplementierungen von Sicherheits-Gateways wegen ihrer Transparenz die flexibelste Art dar, entsprechende Sicherheitsmechanismen einzusetzen und zu verwalten. (Gerald Brose/gg)

Dr. Gerald Brose ist Product Manager Web Services bei Xtradyne Technologies, Berlin.