



# Web Services Security

**Nathan Sportsman**

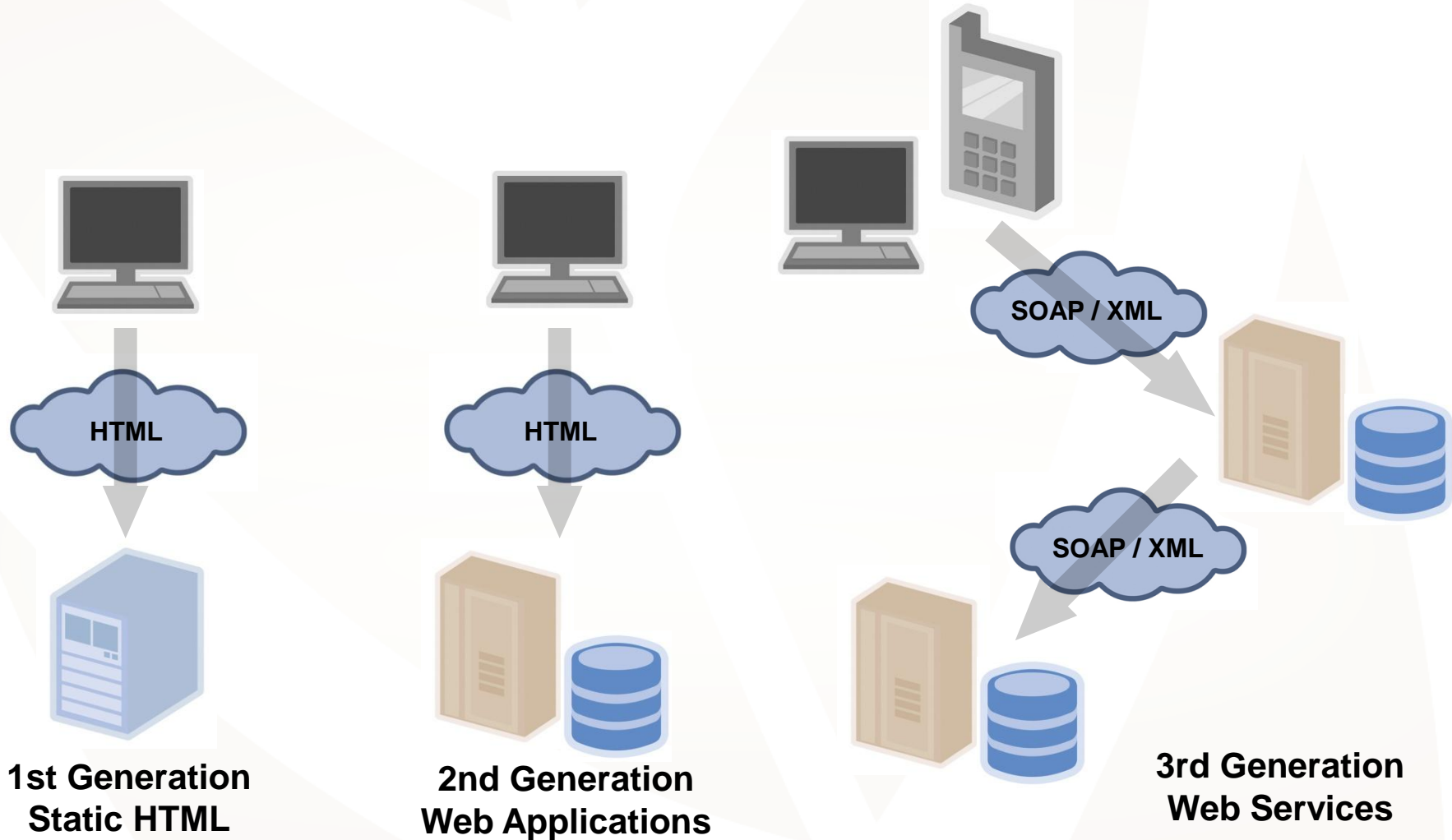
Founder and Chief Executive Officer

# Agenda

- ❖ Web Service Introduction
- ❖ Web Service Vulnerabilities
- ❖ Web Service Countermeasures

# INTRODUCTION

# How Did We Get Here?

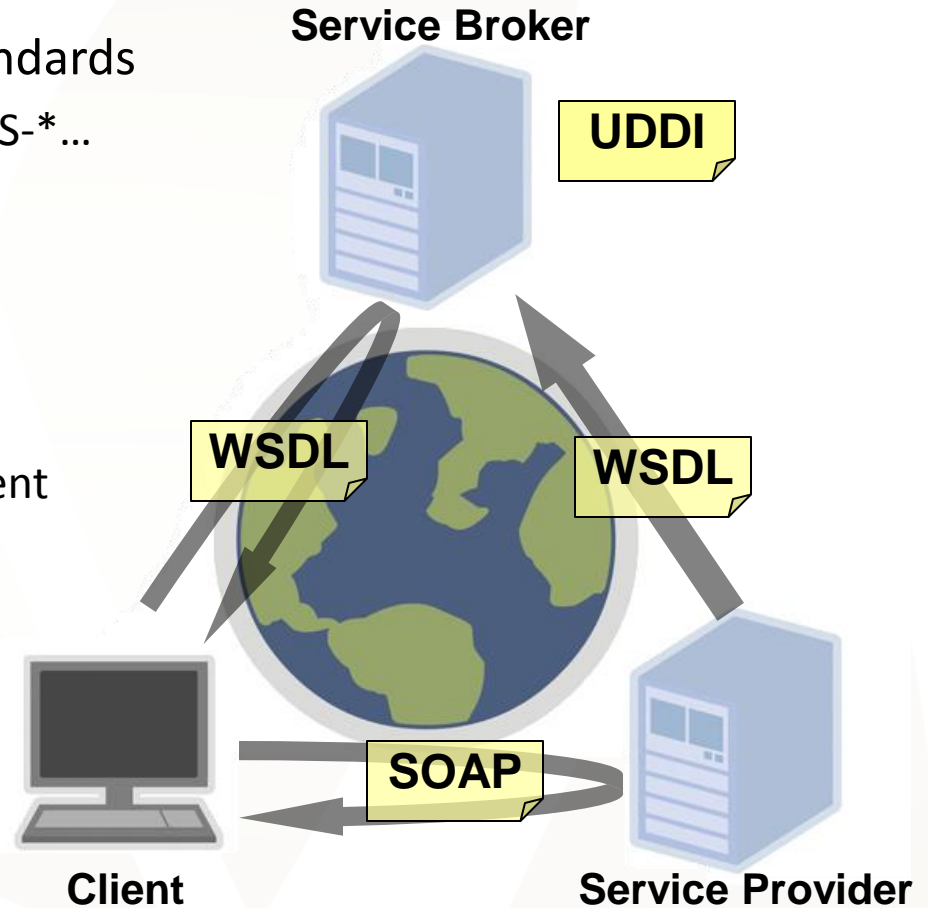


# Web Services Are

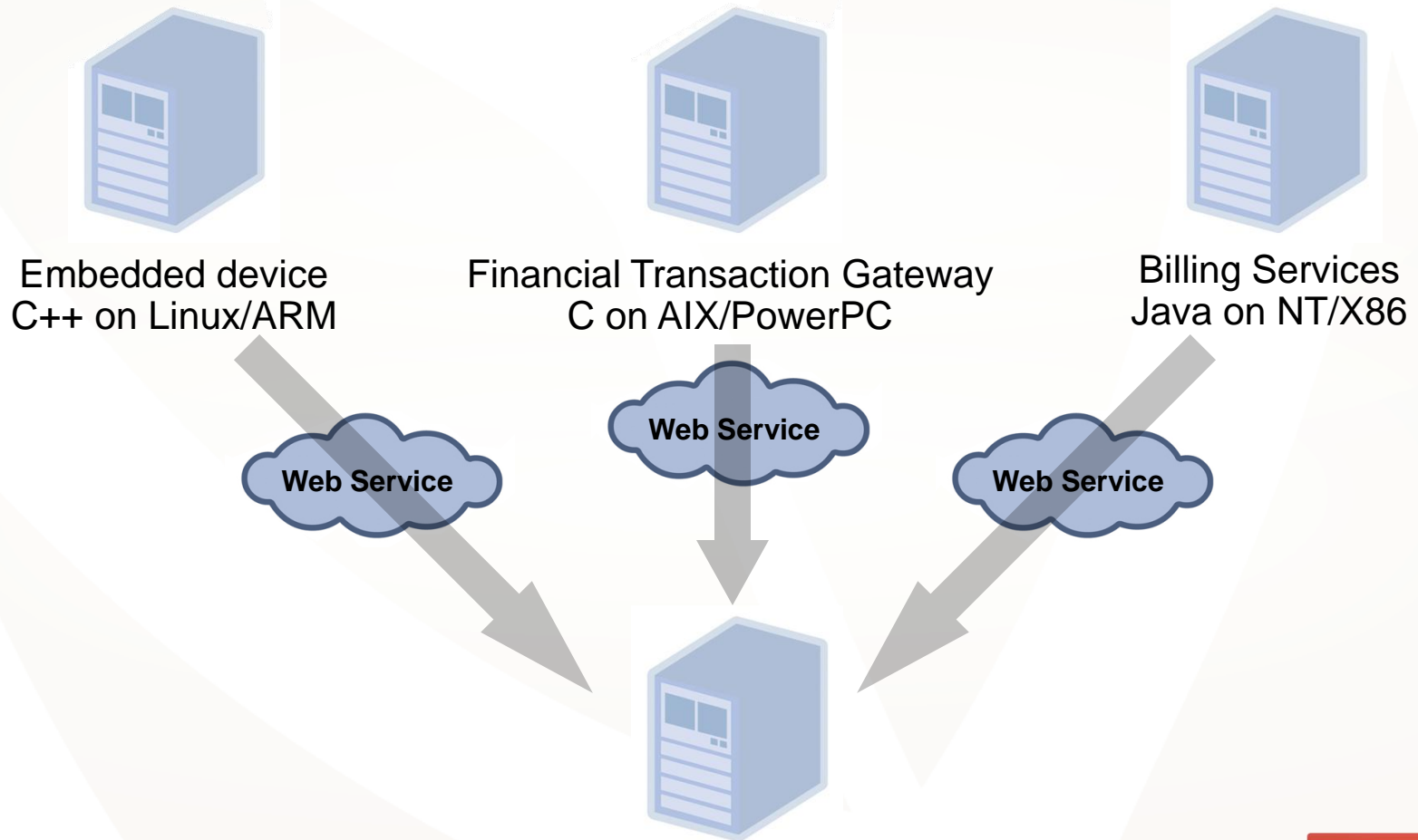
- ❖ “...a software system designed to support interoperable machine-to-machine interaction over a network.”, W3C
- ❖ Capable of connecting to external computing resources
  - Supply chain infrastructure
  - Outsourced computing infrastructure

# Web Services Primer

- ❖ Built on existing and emerging standards
  - HTTP, XML, SOAP, UDDI, WSDL, WS-\* ...
- ❖ Capabilities
  - Loosely coupled
  - Language neutral
  - Platform and transport independent
  - Interoperability



# Web Service Interoperability Example



# WEB SERVICE VULNERABILITIES

# Attack Taxonomies

- ❖ Spoofing
- ❖ Tampering
- ❖ Repudiation
- ❖ Information Disclosure
- ❖ Denial of Service
- ❖ Escalation of Privileges

# Web Services Vulnerabilities

## ❖ Existing and emerging vulnerabilities apply

- Brute Force
- Information Disclosure
- **SQL Injection**
- LDAP Injection
- Session Hijacking
- Denial of Service (DoS)
- **Buffer Overflows**
- Cross Site Scripting
- **XML Injection**
- **XPATH Injection**
- WSDL Manipulation
- **DOS (Intensive XML load)**
- ...

# SQL Injection

- ❖ Possible when user input provided through web service used in queries to backend database

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<SOAP-ENV:Envelope
  xmlns:SOAPSDK1="http://www.w3.org/2001/XMLSchema"xmlns:SOAP
  SDK2="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAPSDK3="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/>
<SOAP-ENV:Body>
  <SOAPSDK4:MethodName xmlns:SOAPSDK4="http://urltoapp/...">
    <SOAPSDK4:username>administrator</SOAPSDK4:username>
    <SOAPSDK4:password>' OR '1'='1</SOAPSDK4:password>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Buffer Overflow

## ❖ Buffer Overflows

- Not as prevalent except on older legacy systems and embedded devices written in unmanaged code
- Large string parameters extending beyond allocated memory
- No bounds checking

```
<SOAP-ENV:Envelope>  
  <SOAP-ENV:Body>  
    <parameter1>  
      lkasdllkdlfa;jkia;refjeoinveroinanlekrngaerinrlgerinreglnag  
      linealinrglanirnaocnilrncoraeincelrgfnernginegnoeingeronger  
      ingeg...  
    </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

# XML Injection

- ❖ External input is not validated and passed in XML stream parsed by second-tier software
- ❖ Alters XML structure by injecting malicious data
- ❖ John Smith escalates privileges by changing his User ID from 100 to 0

```
<MyRec>  
  <UserId>100</UserId>  
  <Username>jsmith</Username><Uid>0</Uid><Username>jsmith</Username>  
</MyRec>
```

# XPATH Injection

- ❖ Similar to SQL injection attack
- ❖ Information stored and retrieved from XML document instead of relational database

```
//users/user[LoginID/text()=' ' or 1=1 and password/text()=' ' or 1=1]
```

# Denial of Service

## ❖ XML parsing can be expensive

- Extremely large / complex XML documents
- Deeply nested tags
- These can create extremely large memory footprints or utilize many CPU cycles

...

```
<SOAP-ENV:Body>
  <BuildNestedXMLResponse xmlns=http://someap">
    <BuildNestedXMLResult>
      <XML 1>
        <XML 2>
          <XML 3>
            <XML 4/>
          </XML 3>
        </XML 2>
      </XML 1>
    </BuildNestedXMLResult>
  </BuildNestedXMLResponse>
```

...

# WEB SERVICES COUNTERMEASURES

# Defense Taxonomies

- ❖ Configuration Management
- ❖ Authentication
- ❖ Authorization
- ❖ User & Session Management
- ❖ Data Validation
- ❖ Error & Exception Handling
- ❖ Logging & Auditing
- ❖ Data Protection (Storage & Transit)

# Configuration Management

- ❖ Internet facing WSDLs can be found with Google hacking (filetype:wSDL inurl:wSDL)
- ❖ Review WSDLs for dangerous or antiquated functions
- ❖ Ensure hidden, debugging, or any non-production functions are removed before deployment
- ❖ Make sure they are not recreated automatically

# Authentication & Authorization

- ❖ Can be accomplished in various ways with various protocols
- ❖ Username/password, Certificates, etc
- ❖ Educate yourself on the characteristics of protocols available before deciding

# Session Management

- ❖ Use proven methods to generate session IDs
- ❖ Do not reinvent the wheel and attempt to create your own
- ❖ Utilize transport encryption to prevent eavesdropping / modification of session data
- ❖ Use transport and element encryption to prevent replay / injection attacks

# Data Validation

- ❖ Validate and sanitize *all* input from external sources
- ❖ Sanitize all output of potentially malicious characters in respect to the next tier (i.e. Database, XML stream, LDAP directory, etc.)
- ❖ If possible, consider a default deny policy with a white list of allowed input

# Logging & Auditing

- ❖ Consider using an existing logging framework
- ❖ Centralize location of log files
- ❖ Ensure logs provide enough information for non-repudiation of action
- ❖ Do not log password, credit cards or other sensitive information

# Error & Exception Handling

- ❖ Test for DoS conditions in QA/QC procedures
- ❖ Define and enforce data file types and sizes
- ❖ Check document complexity before handing to parser
  - XML “Firewall”, etc.
- ❖ Use strict XML schema verification
- ❖ Create custom error messages with minimal information to be returned by web services

# Data Protection (In Storage & Transit)

- ❖ Two mechanisms for encryption, SSL and WS-Security
- ❖ Disadvantages of WS-Security
  - Harder, more complex to implement (Easier to do wrong)
  - Larger attack surface (Attacker has a lot more to play with) vs. SSL with client certificates
  - Only explicitly encrypted / signed data are protected
- ❖ Advantages of WS-Security
  - WS-Security offers end-to-end Security (Instead of point-to-point)
  - Transport agnostic
  - No longer an all or nothing approach
  - Less over head, especially in stateless web services (debatable)

# SSL

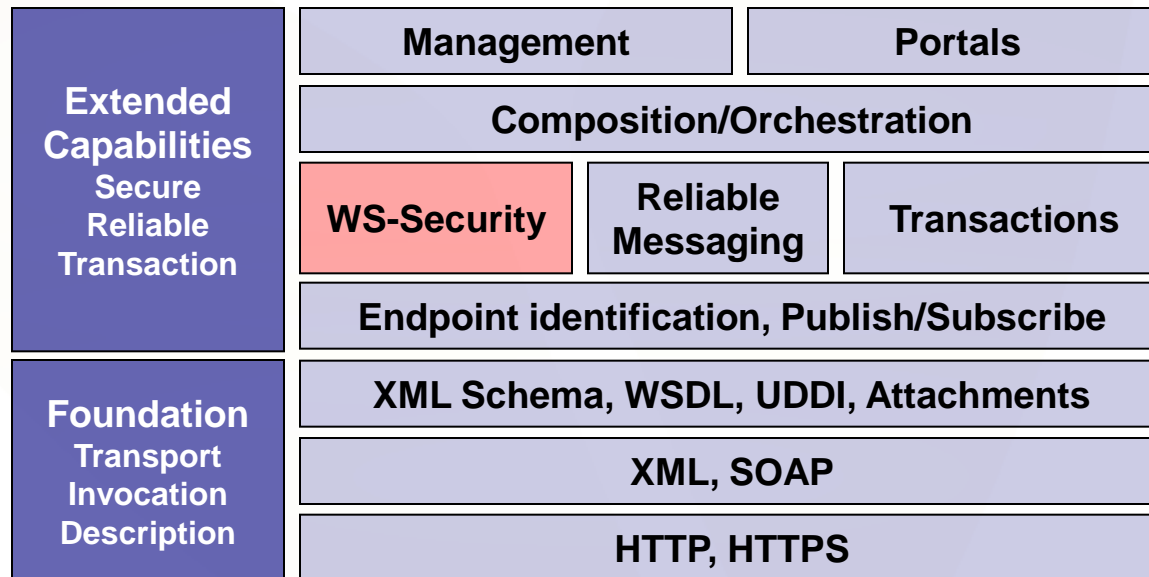
- ❖ Well understood and vetted technology
- ❖ Provides the functionality needed for most web service deployments
- ❖ Who is implementing SSL?
  - ISVs adding web service interface to their product (SSL)
  - Internet Companies exposing part of their service through web interface for consumption (SSL)
  - Internally distributed application previously using older technologies for inter-application communication (SSL)

\* By far majority of engagements, products, and web services we've seen implement SSL solution

# WS-Security

- ❖ Enhances SOAP
  - Provides a framework for message integrity and confidentiality
  - Token type-, Encryption scheme-, and Signature scheme-agnostic
- ❖ Associates security tokens with messages
- ❖ Message integrity provided by XML Digital Signatures in conjunction with security tokens
- ❖ Message confidentiality provided by XML Encryption in conjunction with security tokens
- ❖ Describes mechanism to encode binary security tokens
  - X.509 certificates, Kerberos, opaque encrypted keys
- ❖ Who is implementing?
  - B2B application for company to company exchange

# How WS-Security fits in the Web Service Stack



# Misconceptions

- ❖ Web services do not share some of the same vulnerabilities of web applications
- ❖ WS-Security is all you need to solve security concerns within web services
- ❖ XML firewalls and other technologies will protect against all WS attacks

# Integrate Secure Development Lifecycle

- ❖ Security Requirements
  - Set requirements to meet security objectives
- ❖ Threat Modeling
  - Identify issues at the time of design
  - Assist in other phases of the development life cycle
- ❖ Code Review
  - Identify issues at the time of implementation
  - Static vs Dynamic Analysis
  - Manual and Automated Tools
- ❖ Penetration Testing
  - Blackbox vs White vs Grey Box Testing
  - Manual and Automated Tools

# Web Services Security

**Nathan Sportsman**

Founder and Chief Executive Officer