API Software Development

PUSL3111 Level 3

“WSDL: Web Service Description Language”

Dr. Rasika Ranaweera (ranaweera.r@nsbm.lk | +94 11 544 6126)

School of Computing | NSBM

What is WSDL?

- Web Service Description Language
- WSDL is a document written in XML
- The document describes a Web service
- Specifies the location of the service and the methods the service exposes

Why WSDL?

- Without WSDL, calling syntax must be determined from documentation that must be provided, or from examining wire messages
- With WSDL, the generation of proxies for Web services is automated in a truly language- and platform-independent way

Where does WSDL fit?

- SOAP is the envelope containing the message
- WSDL describes the service
- UDDI is a listing of web services described by WSDL

Document Structure

- Written in XML
- Two types of sections
  - Abstract and Concrete
- *Abstract* sections define SOAP messages in a platform- and language-independent manner
- Site-specific matters such as serialization are relegated to the *Concrete* sections

Abstract Definitions

- **Types:** Machine- and language-independent type definitions.
- **Messages:** Contains function parameters (inputs are separate from outputs) or document descriptions.
- **PortTypes:** Refers to message definitions in Messages section that describe function signatures (operation name, input parameters, output parameters).

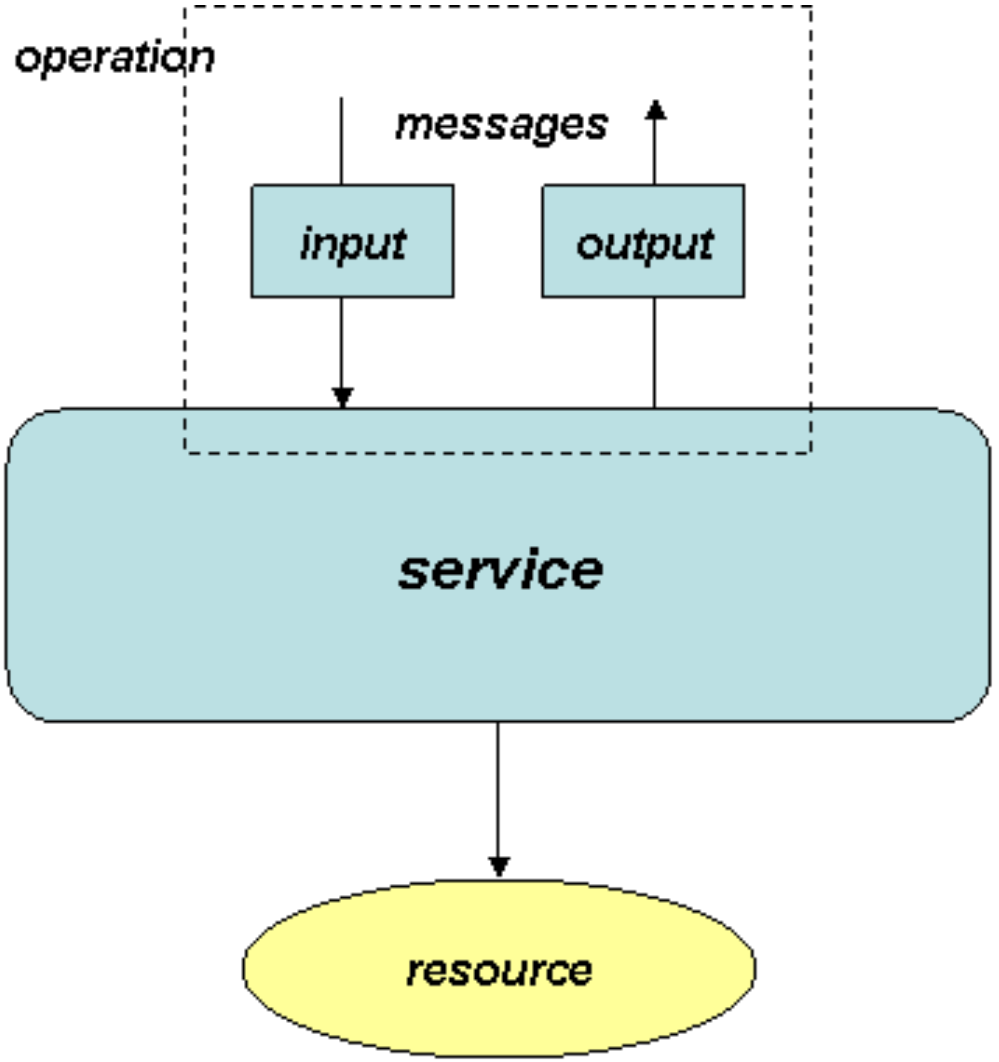
Concrete Descriptions

- **Bindings:** Specifies binding(s) of each operation in the PortTypes section.
- **Services:** Specifies port address(es) of each binding.

Operation

- An *operation* is similar to a function in a high level programming language
- A message exchange is also referred to as an operation
- Operations are the focal point of interacting with the service

Big Picture



An Example

- `<?xml version="1.0" encoding="UTF-8"?>`
- This first line declares the document as an XML document.
- Not required, but helps the XML parser determine whether to parse the file or signal an error

Types Section

- The *type* element defines the data types that are used by the web service.

```
<xsd:complexType name="PERSON">
  <xsd:sequence>
       <xsd:element name="firstName“ type="xsd:string"/>
       <xsd:element name="lastName" type="xsd:string"/>
       <xsd:element name="ageInYears" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>
```

Messages Section

- A *message* element defines parameters
- The name of an output message element ends in "Response" by convention

```
<message name="Simple.foo">
     <part name="arg" type="xsd:int"/>
</message>

<message name="Simple.fooResponse">
     <part name="result" type="xsd:int"/>
</message>
```

PortTypes Section

- Defines a web service, the operations that can be performed, and the messages that are involved.

```
<portType name="SimplePortType">
      <operation name="foo" parameterOrder="arg" >
            <input message="wsdlns:Simple.foo"/>
            <outputmessage="wsdlns:Simple.fooResponse"/>
      </operation>
</portType>
```

Bindings Section

- The *binding* element defines the message format and protocol details for each port.

```
<operation name="foo">
  <soap:operation
       soapAction="http://tempuri.org/action/Simple.foo"/>
  <input>
      <soap:body use="encoded"
        namespace="http://tempuri.org/message/"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
   </input>
   <output>
      <soap:body use="encoded”
        namespace=http://tempuri.org/message/
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
   </output>
</operation>
```

The Port Element

- Each <port> element associates a location with a <binding> in a one-to-one fashion

```
<port name="fooSamplePort"
      binding="fooSampleBinding">
  <soap:address
location="http://carlos:8080/fooService/foo.asp"/>
</port>
```

Services Section

- A collection of related endpoints, where an endpoint is defined as a combination of a binding and an address
-
```
<service name="FOOSAMPLEService">
<port
name="SimplePort“ binding="wsdlns:SimpleBi
nding">
<soap:address
location="http://carlos:8080/FooSample/
FooSample.asp"/>
</port>
</service>
```

An Example

```
<message name="Simple.foo">
  <part name="arg" type="xsd:int"/>
</message>
<message name="Simple.fooResponse">
  <part name="result" type="xsd:int"/>
</message>
<portType name="SimplePortType">
  <operation name="foo" parameterOrder="arg" >
   <input message="wsdlns:Simple.foo"/>
   <output message="wsdlns:Simple.fooResponse"/>
  </operation>
</portType>
```

The above describes what kind of C/C++ function call?

```
int foo(int arg);
```

Namespaces

- The purpose of namespaces is to avoid naming conflicts.
- Imagine two complimentary web services, named A and B, each with an element named “foo”.
- Each instance of foo can be referenced as A:foo and B:foo
- Example: "xmlns:xsd" defines a shorthand (xsd) for the namespace
- See http://www.w3.org/2001/XMLSchema.

Demo

Create a web service and generate its WSDL document

https://localhost:44373/PUSL3111WebService.asmx?wsdl

https://cs.au.dk/~amoeller/WWW/webservices/GoogleSearch.wsdl

API Software Development

PUSL3111 Level 3

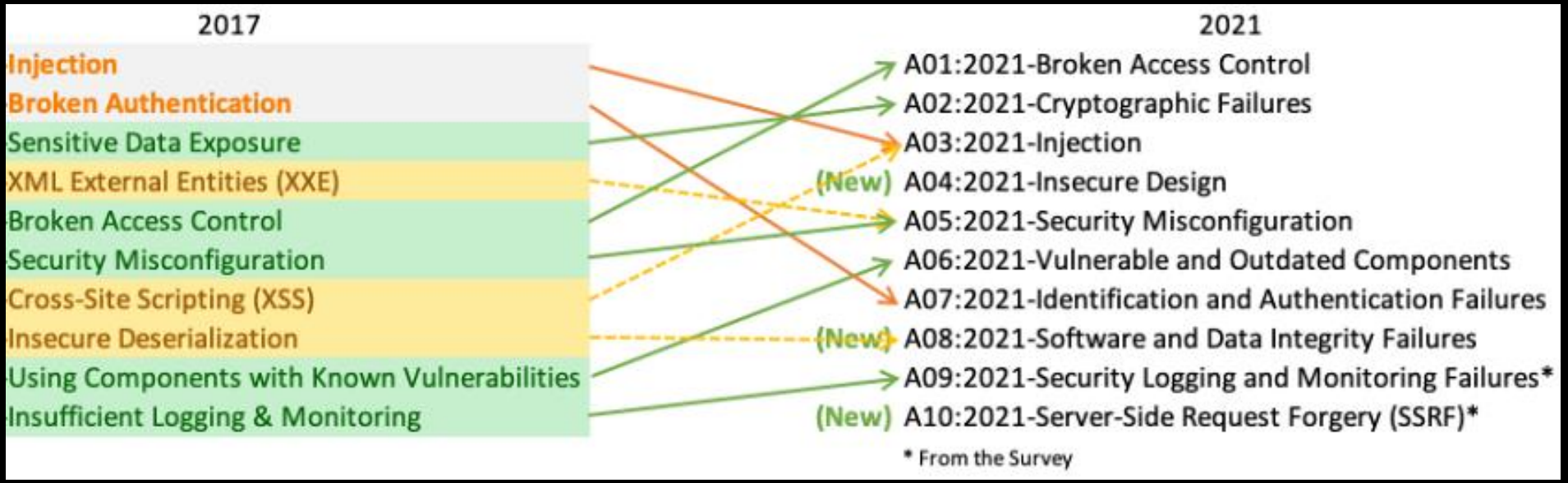
“Privacy and Ethics”

Dr. Rasika Ranaweera (ranaweera.r@nsbm.lk | +94 11 544 6126)

School of Computing | NSBM

NSBM
NATIONAL SCHOOL OF BUSINESS MANAGEMENT

Introducing OWASP
https://www.owasp.org



Open Web Application Security Project

Dealing with top 10

Injection - How does your API deal with SQL injection?

- Include a test in your API to demonstrate quality
- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Authentication and session management

- Does your API use session based authentication? It should!
- Is incoming method valid for that resource? eg: delete verb /http://modules/4 not http://modules/
- https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Security

- 100% security is not achievable goal
- Security costs money
  - Determine appropriate countermeasures for assets requiring protection
  - Address problem in structured and consistent manner
- Standards offer baseline
  - ISO/IEC 17799 standard
  - Business requires security over and above that baseline
    - **Question**: How do you ensure that a third party API provider's approach to keeping information safe is the same as yours?
    - "Computer security is the protection of a company's assets by ensuring the safe, uninterrupted operation of the system and safeguarding of its computer, programs and data files"

Security & Privacy

- Business responsibility for personal data collection
- They get sued if data is lost (in theory)
- Need to prevent unauthorized information disclosure
- Data access restricted to authorized entities
- Legitimate “need to know”
- Seriousness of disclosure dictated by whether it occurs to an unauthorized member of same organization or total outsider

Questions to ask..

- What assets are being secured here?
- How will the security measures you plan to implement impact the performance of the API?
- Who is using your API?
  - Do you need users to identify themselves before they use the client applications?
  - Or do you need to just identify the application?

Basic Security Techniques

- Identification
  - Who are you dealing with?
- Authentication
  - Are they really who they say they are?
- Authorization
  - What are they allowed to see/do?

Management

- How do users get the accounts?
- Needs a process to set up account and authentication credentials
- Processes required to add, reset, revoke or delete the accounts
- How do API keys get issued?
- May consider using roles - e.g.: all those with an x type of account get access to 1 set of data or functionality

Identification

- Commonly done through use of API key
- Simple, random identifiers
- Often Passed through the HTTP query parameter (BAD MOVE see OWASP)
  - https://www.googleapis.com/books/v1/volumes?q=restful+web+services&key=AIza..
- Included in each API request - ought to be via POST body
- Developer has to request key
- API can monitor usage
- Not encrypted so easily discovered - so used more for audit than security
- Stops unauthorized applications from flooding system (Denial, breach of terms)
- If API uses personal information, must have authentication from end user when client app gains access

Authentication

- Usernames, passwords, OAuth
- Used when sensitive data involved
- HTTP Basic authentication is easiest and most common (Over SSL)
- Client app has to store password securely & appropriately
- Security Assertion Markup Language (SAML) - based on secure distribution of public keys to individual clients along with WS-Security specifications (SOAP)

oAuth

- Open protocol using a standard method
- Manages the handshakes between applications
- Used when API publisher wants to know who is communicating with system
- Provides a proxy authentication system
  - User provides log in details to OAuth service provider
  - Service provider issues tokens
- oAuth creates a token for the user which gives ONE application access to ONE API on behalf of ONE user - single use
- Can be set to expire after a period of time
- Builds in resilience - if token discovered, cannot be reused
- http://oauth.net/

Authorisation

- States what the identity can do
  - Should user see contents?
  - Can user change contents?
  - Does user have permission to perform that action?

Risk Assessment

- Does your API allow:
  - Freedom from intrusion?
  - Freedom from interference in choices & decisions?
  - Control over flow of personal information?

Risk Analysis activity

- Risk is analyzed as likelihood + impact
- Think about your API
  - Does it collect or use personal information?
  - Does it make use of mission critical data?
  - List 2 privacy issues you can think of?
  - Rank the likely impact of those issues
    - High, Medium, Low

Legals

- Distribution of content
  - What rights do you have?
  - What rights are you giving to clients
- May need to consider different levels of access
- Legal considerations - data protection, copyright, DDA

Contracts & Terms

- Context dependent
  - Consumption of API for public use or private use
- Terms of use directed at end user as well as developer of client application
  - The client application must prevent access until end user has agreed to terms
  - Client application must pass on changes to terms

Where are you?

- Where your API is hosted can cause an issue
- EU have a common directive (1995) allowing citizens fundamental rights
  - e.g. right to obtain copies of records
- America as a more piecemeal approach
  - No right to obtain copies
- http://www.nytimes.com/2013/02/03/technology/consumer-data-protection-laws-an-ocean-apart.html?_r=1&

Joining things together

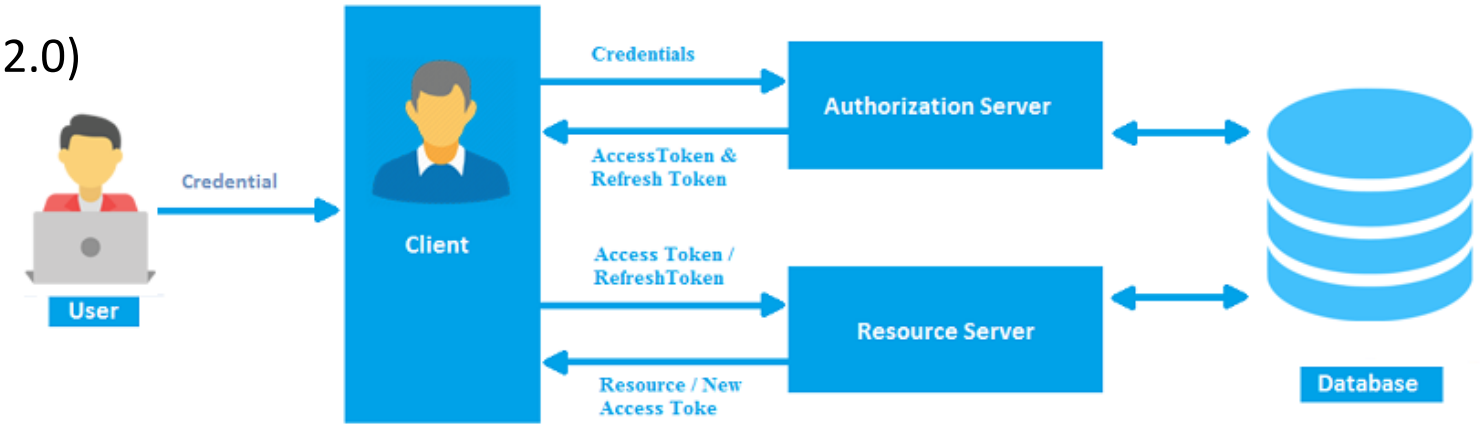
- Issues when information aggregated
- RFID, Store cards, CCTV, Credit Cards
- In a world of Big Data
  - 80% of world's data unprotected
    http://www.guardian.co.uk/news/datablog/2012/dec/19/big-data-study-digital-universe-global-volume
  - ICO new code of practice for protecting privacy rights when using it
    http://www.guardian.co.uk/news/datablog/2012/nov/21/anonymised-data-protection-code-freedom-of-information
- Issue is YOU as developer need to ensure a robust privacy model
  - consider consequences

Privacy

- Policies essential if handling personal or sensitive data
- Data retention must be communicated
  - Who sees it and how long for
- Privacy implementation based on privacy model
  - Privacy Interface Analysis - can user understand what is happening to data, how does application interact, third parties approach
  - Privacy Impact Assessment - flow of personal data through application, method of data handling

Web API Authentication

- HTTP Authentication Schemes (Basic & Bearer)
- API Keys
- OAuth (2.0)



- https://www.c-sharpcorner.com/article/web-ap/
- https://www.techieclues.com/articles/token-based-authentication-in-asp-net-web-api