

## Introduction to SOAP

Scott D. Stoller

CSE608: Advanced Computer Security



1

## References

- This is a slightly edited version of the presentation “Introduction to SOAP” by Joe Futrelle, National Center for Supercomputing Applications, available at [www.ncsa.uiuc.edu/People/futrelle/ppt/ntuSOAP.ppt](http://www.ncsa.uiuc.edu/People/futrelle/ppt/ntuSOAP.ppt)

2

## What is SOAP?

- The Simple Object Access Protocol
- “Platform-agnostic” protocol for messaging and remote procedure calling (RPC) between distributed applications
- Developed by the World-Wide-Web Consortium (W3C)
- Encoded in XML, transported over HTTP
  - Data encoding described in XML-Schema
  - Can use any transport mechanism
- Stateless (request/response)

3

## What SOAP isn't

- A full-fledged distributed object system with support for
  - Objects-by-reference
  - Activation
  - Message batching
  - Distributed garbage collection
  - Naming and directory services
- SOAP sacrifices these features for platform-neutrality and extensibility

4

## SOAP in a Nutshell

- A SOAP client formats a message in XML including a SOAP “envelope” element describing the message
- The client sends the message to a SOAP server in the body of an HTTP request
- The server determines whether the message is valid and supported
- The server formats its response in XML and sends it to the client in the body of an HTTP response

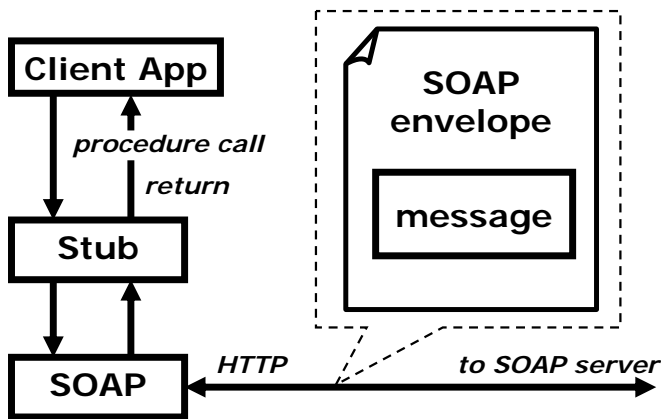
5

## Is SOAP Web-Based?

- No
  - Browsers don't support it
  - Likely won't: SOAP is for application-to-application communication
- But it's based on WWW infrastructure
  - SOAP servers can be implemented as CGI scripts or Servlets
  - SOAP messages work through proxies and firewalls
  - SOAP clients use HTTP client libraries

6

## SOAP Use Scenario: RPC



7

## SOAP Message Components

- **Envelope (required)**
  - Contains Header and Body
  - Defines namespaces (optional)
  - Declares encoding rules (optional)
- **Header (optional)**
  - Contains metadata entries about message *a la* HTTP headers
  - Specifies which entries must be understood and by which target “actor” in chain of recipients

8

## SOAP Message Components

- **Body (required)**
  - Contains application-specific message
  - May be encoded variously
- **Fault element (optional)**
  - Contained in Body
  - Describes error class (version mismatch, headers not understood, client error, server error)
  - Extensible, hierarchical fault codes, e.g. `Server.Availability.NotAvailable`

9

## Example SOAP Message

```

<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <lib:PatronID
      SOAP-ENV:mustUnderstand="1">
      007
    </lib:PatronID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <lib:Checkout>
      <callNo>435.33</callNo>
    </lib:Checkout>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
  
```

header

body

10

## Why Not Just Put Everything in the Header?

- The Header is optional
- SOAP specializations for RPC require using the body to pass parameters and results (we’ll get to that later)
- Otherwise, no reason
- The SOAP spec notes that a body entry is semantically-equivalent to an optional header entry (with “mustUnderstand” = 0)

11

## Encoding Data in SOAP

- SOAP permits arbitrary “encoding styles” and defines a default encoding style
- Based on XML-Schema
- Supports data types
  - All built-in XML-Schema types (e.g. string, float, integer, date, IDREF)
  - Derived types: enumerations, arrays, structs, generic compound types

12

## Encoding Data in SOAP: accessors

- Typed elements are called “accessors”
- Accessor types are specified in externally-referenced XML-Schema definitions
- Or with the `xsi:type` attribute, e.g.

```
<a:uid xsi:type="xsd:integer">
  4737
</a:uid>
```

13

## Encoding Data in SOAP: structs

- Structs are simple compound types consisting of uniquely-named accessors, e.g.

```
<naut:Ship>
  <name>Titanic</name>
  <length>882'</length>
  <height>104'</height>
  <weight>46,328 tons</weight>
</naut:Ship>
```

14

## Encoding Data in SOAP: arrays

- Arrays can be built of any type, e.g.

```
<SOAP-ENC:Array
  SOAP-ENC:arrayType="phNo[ ]">
  <phNo>217-435-4927</phNo>
  <phNo>217-957-4937</phNo>
</SOAP-ENC:Array>
```

- Array values can be of any subtype of the array's declared type

15

## Encoding Data in SOAP (cont.)

- Values can be used in multiple places using references

```
<e:Book>
  <title>My Life</title>
  <author href="#Person-1"/>
</e:Book>
<e:Person id="Person-1">
  <name>Henry Ford</name>
</e:Person>
```

- Including ref's to external resources

16

## Using SOAP for RPC

- A method is invoked by sending a SOAP request to a URI representing the target object
- Method calls are represented in the Body by a struct named after the method
- Method arguments are represented by struct members
- Return value is encoded similarly, with method name + “Response” as the struct name, and returned in a SOAP response
- Errors are signaled with the Fault entry

17

## RPC Example: Digital Library

- Supports card catalog search and checkout
- Target objects / methods
  - **Catalog / search**
  - **Circulation / checkout, return**
- **Security**
  - **Authentication: patron ID**
  - **Fault: unrecognized user**

18

## RPC Example: Digital Library

### • Catalog

- Represented by a URI, e.g.  
<http://www.lib.org/catalog/>
- Supports one method: Search
  - Find items which match a particular query
  - Result: list of entries containing information about items
  - Fault condition: malformed query

19

## RPC Example: Catalog Search

```
POST /catalog/ HTTP/1.1
Content-type: text/xml
```

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <lib:Search>
      <author>Kafka</author>
    </lib:Search>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

20

## RPC Example: Catalog Search Response

```
HTTP/1.1 200 OK
Content-type: text/xml
```

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <lib:SearchResponse>
      <lib:Item>
        <callNo>473.57</callNo>
      </lib:Item> ...
```

21

## RPC Example: Catalog Search

- Now let's make a malformed query:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <lib:Search>
      <author>**$(D-3.2</author>
    </lib:Search>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

22

## RPC Example: Catalog Search Response

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>
        Client.MalformedQuery
      </faultcode>
      <faultstring>
        Illegal Characters
      </faultstring>
    </SOAP-ENV:Fault> ...
```

23

## RPC Example: Digital Library

### • Circulation

- Represented by a URI, e.g.  
<http://www.lib.org/circulation>
- Supports two methods: Checkout / Return
  - Acquire/release rights to a particular item
  - Result: rights token / acknowledgement
  - Fault conditions: item not found, item unavailable / not checked out, unrecognized user
- Must understand authentication
  - Require patron ID in Header

24

## RPC Example: Circulation Checkout

POST /circulation/ HTTP/1.1  
Content-type: text/xml

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <lib:PatronID
      SOAP-ENV:mustUnderstand="1">
      007
    </lib:PatronID>
  </SOAP-ENV:Header>
```

... continued ...

25

## RPC Example: Circulation Checkout

... continued from last slide ...

```
<SOAP-ENV:Body>
  <lib:Checkout>
    <callNo>435.33</callNo>
  </lib:Checkout>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

26

## RPC Example: Checkout Result

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <lib:CheckoutResponse>
      <rightsToken>
        87-RJC-Q
      </rightsToken>
    </lib:CheckoutResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

27

## RPC Example: Circulation Checkout

- Now let's try that without authentication:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <lib:Checkout>
      <callNo>435.33</callNo>
    </lib:Checkout>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

28

## RPC Example: Checkout Result

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>
        Client.NoPatronID
      </faultcode>
      <faultstring>
        No Patron ID Specified
      </faultstring> ...
```

29

## Notes about RPC with SOAP

- Request and response formats must be described in XML-Schema and reside in their own namespace
- Namespaces are declared in the SOAP Envelope
- RPC calls may be made through SOAP intermediaries (proxies)
  - Faults and “must understand” contracts must be maintained between intermediaries and target objects

30

## Notes about RPC with SOAP

- SOAP servers may validate messages against XML-Schemas, ensuring that data in messages conforms to the declared data types
- Binary data may be sent in one of several XML-Schema encodings

31

## SOAP Status and Resources

- SOAP 1.2 Specification  
(a W3C Recommendation)  
– <http://www.w3.org/TR/soap/>
- Useful Microsoft whitepaper  
– [http://msdn.microsoft.com/xml/general/soap\\_webserv.asp](http://msdn.microsoft.com/xml/general/soap_webserv.asp)
- Open-source Implementation of SOAP:  
Apache Axis  
– <http://ws.apache.org/axis/>

32