

Web Service Security Through A Guard

Roxanne Yee
University of Hawai'i at Mānoa
Internship Site Akimeka, LLC

Mentor & Co-author Marc Lefebvre
Program Manager & Advisor Todd Lawson
Project Hierarchy Information Assurance Group, Cross Domain Solutions Group, GWSG Project, SOA Test Lab

1.0 Motivation

Cross Domain refers to the connection and data transfer between domains that have differing security standards. Following are two Cross Domain Solutions (CDS) currently used to allow a soldier in the field on a classified network, where strategic planning occurs, to gain access to needed data stored on an unclassified network, where data is collected.



CDS1 Physically copy data from the unclassified network database to a medium such as a CD and transport it to the classified network database.



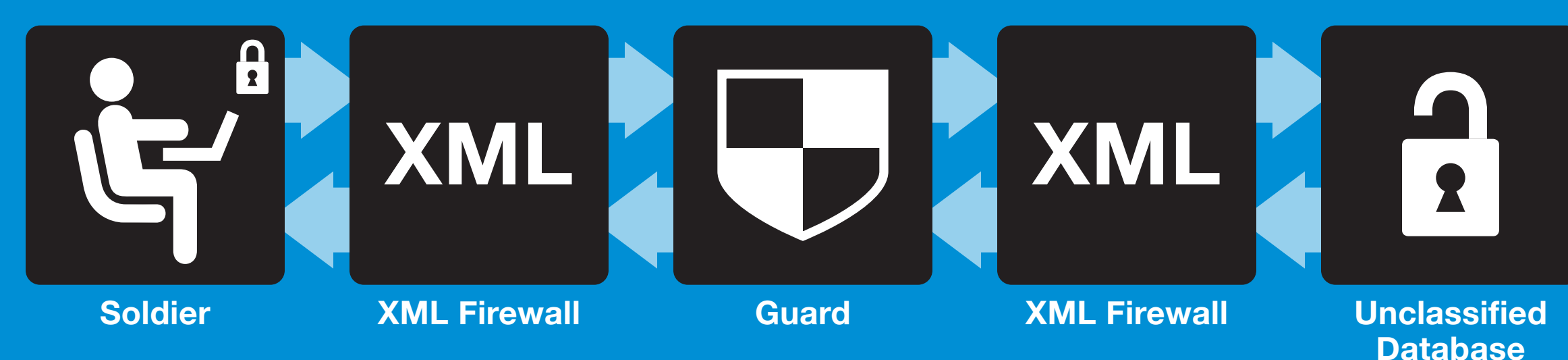
CDS2 Replicate data on the unclassified network database and send it through a guard to be stored in the classified network database. The guard is a government-proprietary and government-certified device that acts as a filter for the classified side; it keeps out harmful traffic and keeps in sensitive data.

1.1 Disadvantages to Current CDS

- Redundant Copies of Data
- Costly Use of Time
 - » Replication
 - » Transportation
- Need For Data Synchronization
 - » Frequent Manual Updates
- Unidirectional Data Flow
 - » Unclassified to Classified
- No Guarantee of Data Availability
- Extra Manpower by Man-In-The-Loop

1.2 New CDS Under Study

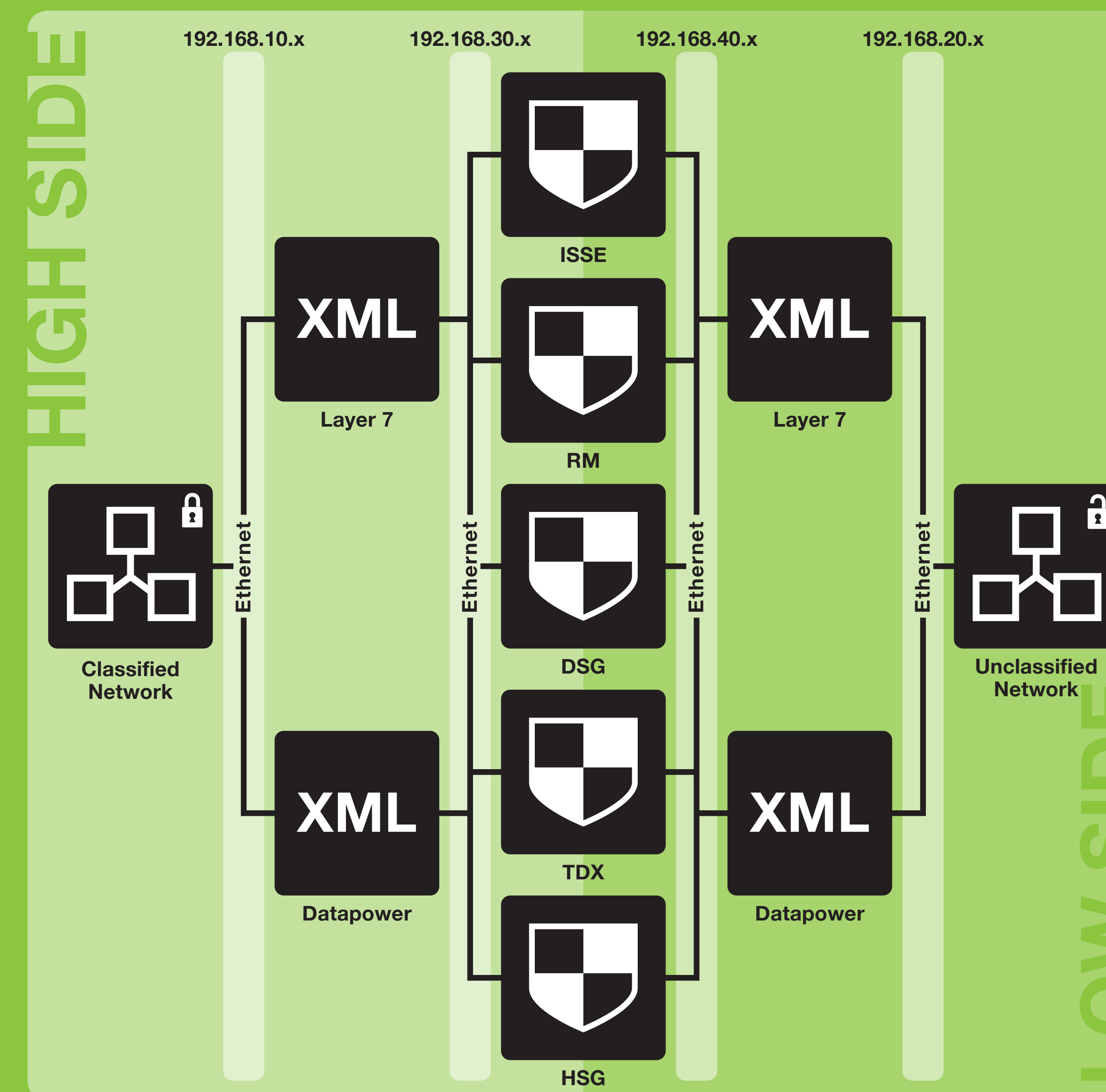
Akimeka is currently researching and developing new CDS using modern web services technology. One scheme under study utilizes already existent guards.



CDS3 Allow soldier on the classified client side to access the unclassified server side database by request-response interaction. XML (Extensible Markup Language) firewalls provide an extra layer of security as well as translate between the guard's proprietary language and protocol and the SOAP (Simple Object Access Protocol) over HTTP (Hypertext Transfer Protocol) that most modern web services use.

2.0 Project Perspective

CDS3 utilizes modern guards that must be tested for compatibility, capability, and efficiency concerning web service technology. Akimeka's Service Oriented Architecture (SOA) Test Lab is an evaluation facility for the guards specified by the National Security Agency (NSA) and Defense Information Systems Agency (DISA). From lab results, Akimeka will be able to suggest the best guard solution given any specific situation.



2.1 Project Goal: Web Service Security (WSS)

WSS, or WS-Security, is a standard developed by OASIS (Organization for the Advancement of Structured Information Standards) applied to SOAP messages. This project's goal was to research and document implementation of WSS with the open source software used by Akimeka. Documentation will be a foundation and template when Akimeka tests WSS compatibility in their SOA Test Lab.

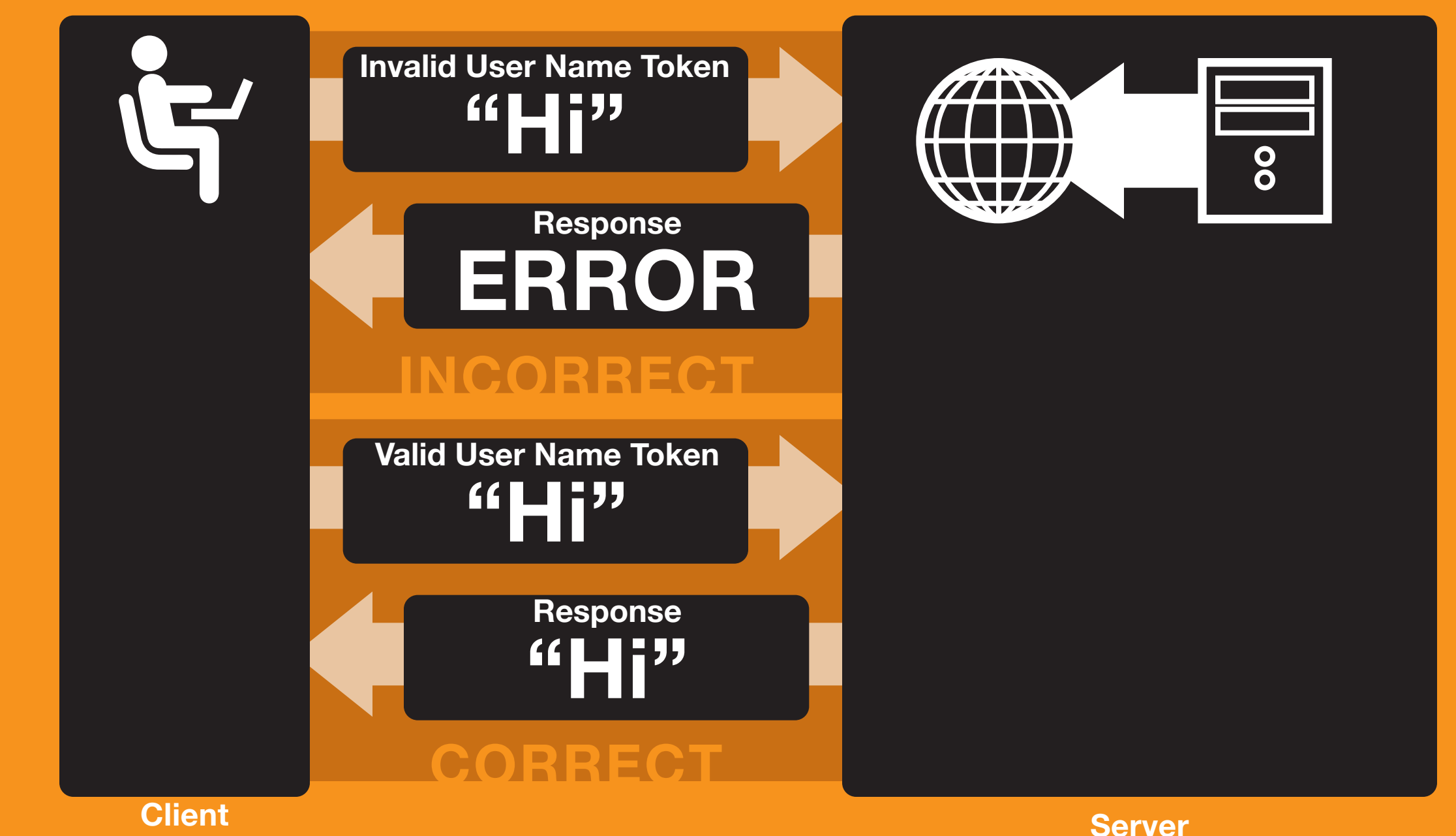
2.2 Test Bench Outside of SOA Test Lab

As a test bench, both client and server were simulated on a single computer. Communication occurred through the localhost interface. The test web service used was a simple echo service. To implement WSS, soapUI was used on the client side and Apache Rampart was used on the server side.



3.0 Example: User Name Token

One of the WSS mechanisms researched was User Name Token. For proper web service functionality, the client needed to send both a correct username and password with their request.



3.1 WSS on the Server

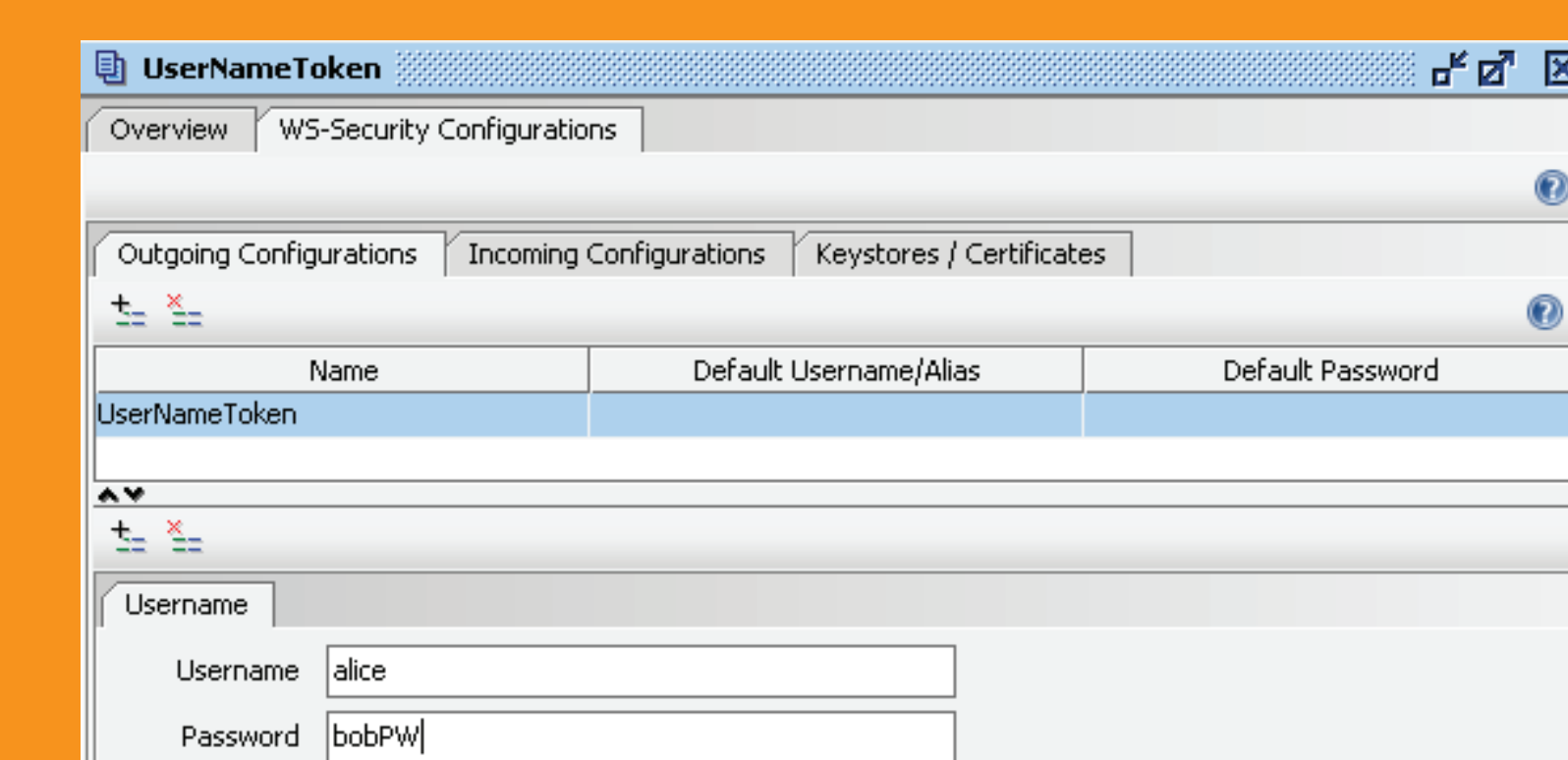
Configuring Rampart for WSS required alteration to the test service's services.xml. f

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
  <operation name="echo">
    <messageReceiver class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    <parameter name="ServiceClass" locked="false">org.apache.rampart.samples.policy.sample01.SimpleService</parameter>
    <module ref="addressing"/>
  </operation>
  <module ref="rampart"/>
  <wsp:Policy wsuid="UTF" xmlns:wsu="http://.../" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"><wsp:ExactlyOne><wsp:All>
    <sp:SupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
      <wsp:Policy><sp:UserNameToken sp:IncludeToken="http://.../IncludeToken/AlwaysToRecipient"/></wsp:Policy>
    </sp:SupportingTokens>
    <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/policy">
      <ramp:user=username/>
      <ramp:passwordCallbackClass=org.apache.rampart.samples.policy.sample01.PWCBHandler/>
    </ramp:RampartConfig>
  </wsp:All></wsp:ExactlyOne></wsp:Policy>
</services>
```

*Text outside of the box illustrates a typical services.xml file used to configure a web service. The boxed text configures Rampart to use WSS with the web service.

3.2 WSS on the Client

soapUI's Outgoing Configuration GUI (Graphical User Interface) applied WSS to the request sent from the client.



3.3 Results and Conclusion

After configuring the client and server for WSS, a tests were done, using the User Name Token WSS mechanism, to see if the configuration was accurate.

Username	Password	Result
Correct	Correct	Echo
Incorrect	Incorrect	Error
Blank	Blank	Error
Correct	Incorrect	Error
Correct	Blank	Error
Incorrect	Correct	Error
Incorrect	Blank	Error
Blank	Correct	Error
Blank	Incorrect	Error

Results of checking valid and invalid User Name Tokens were as expected. Thus, the configuration code was correct and could be used in the SOA Test Lab. With this as a basis, Akimeka will test more complex standards that are used in the field.