Web Services Hacking and Hardening

Adam Vincent, Sr. Federal Solutions Architect

March 8th, 2007

Presenter Bio and Honorable Mention

Adam Vincent, Sr. Federal Solutions Architect

Adam Vincent has his BS in Computer Science and will be completing his MS in Computer Science this year. Adam's government related experience began as a systems administrator, working as a contractor for the U.S. Department of State where he functioned as a Network Administrator, and later a Software Engineer. After State Department, Adam joined The MITRE Corporation, a Federally Funded Research and Development Center (FFRDC) where he held the title of Senior Information Security Engineer and specialized in SOA security and cross boundary information sharing. In this position, Adam worked closely with Federal and Civilian organizations in pursuit of building Secure Service Oriented Architectures and securely sharing information across security boundaries. In addition, while at MITRE, Adam taught classes on SOA vulnerability assessment and XML Firewalls to FFRDC and Government personnel.

Phone: 703-965-1771 Email: avincent@gov.layer7tech.com

Some of the concepts portrayed in this presentation were based on the book “Hacking Web Services” by Shreeraj Shah. This is the first book of its kind in my opinion and portrayed the topic of Web Services Hacking in a concise and correct fashion.

Caveats

The following presentation will **NOT** be a vendor Pitch but will hopefully educate the audience in Web Services Hacking, Testing, and Hardening Techniques.

Real life examples may be offered that relate to deployment of Layer 7 Technologies product line.

Hardening of Web Services will have some focus on technologies like those Layer 7 Technologies provides. Layer 7's product will be used as an example in this portion of the presentation.

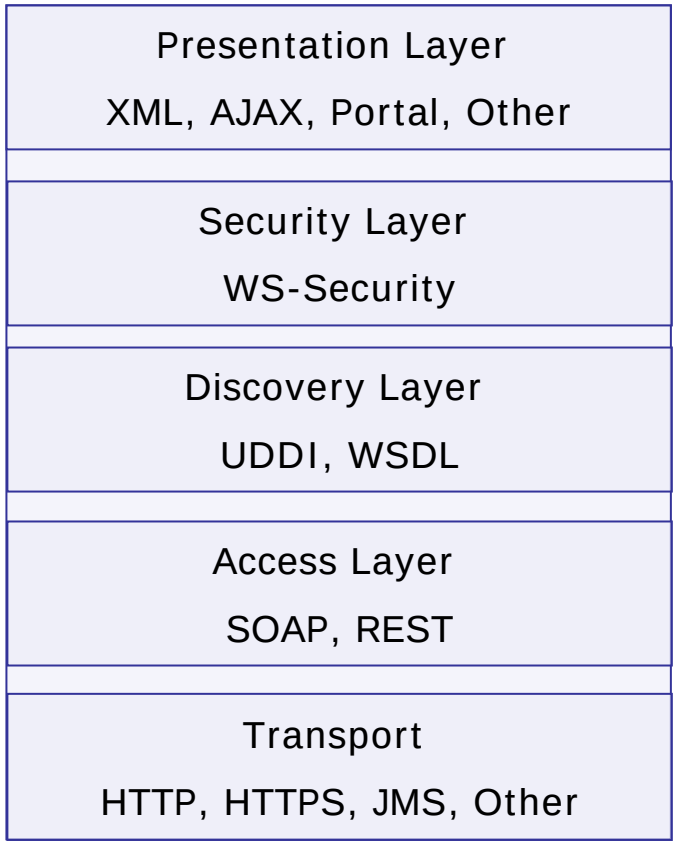
LAYER 7 TECHNOLOGIES

Agenda

- Components and Terminology
- Web Services Threats
- Web Services Hacking
- Web Services Hardening
- Conclusion and Questions

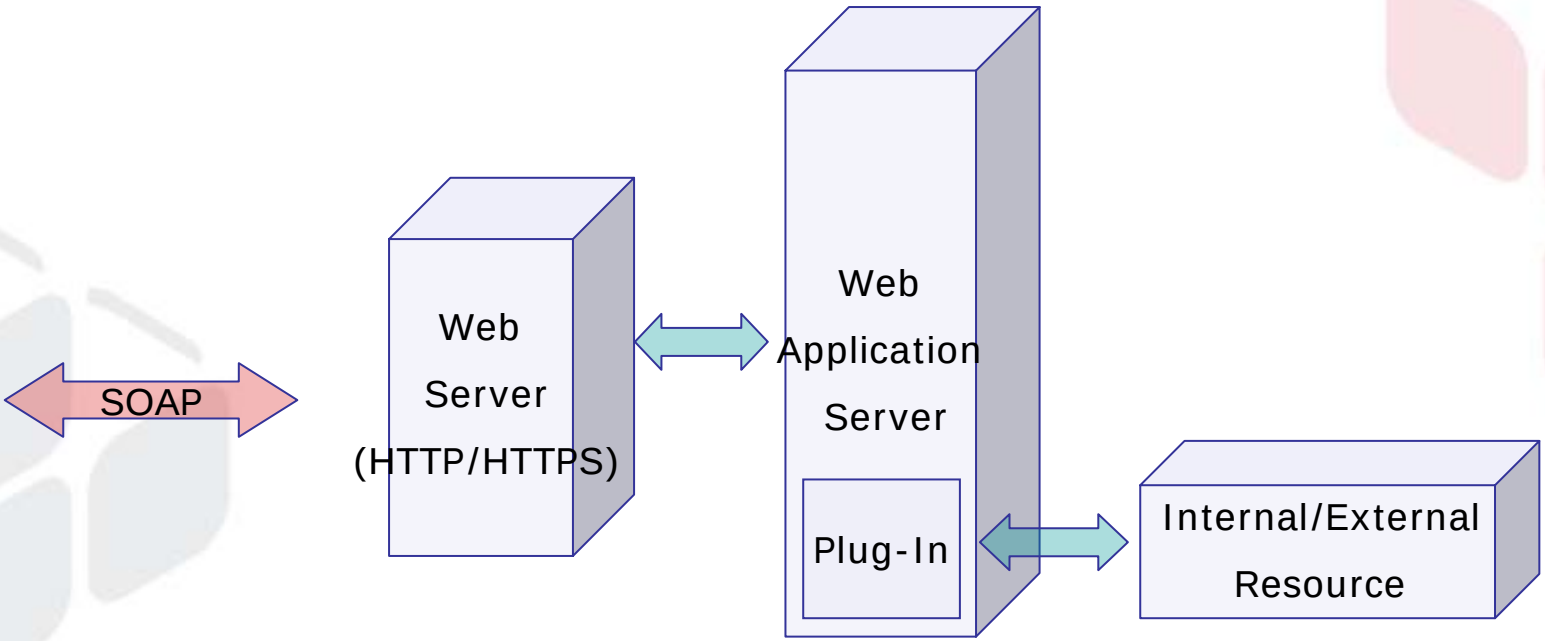
LAYER 7 TECHNOLOGIES

Web Services Stack



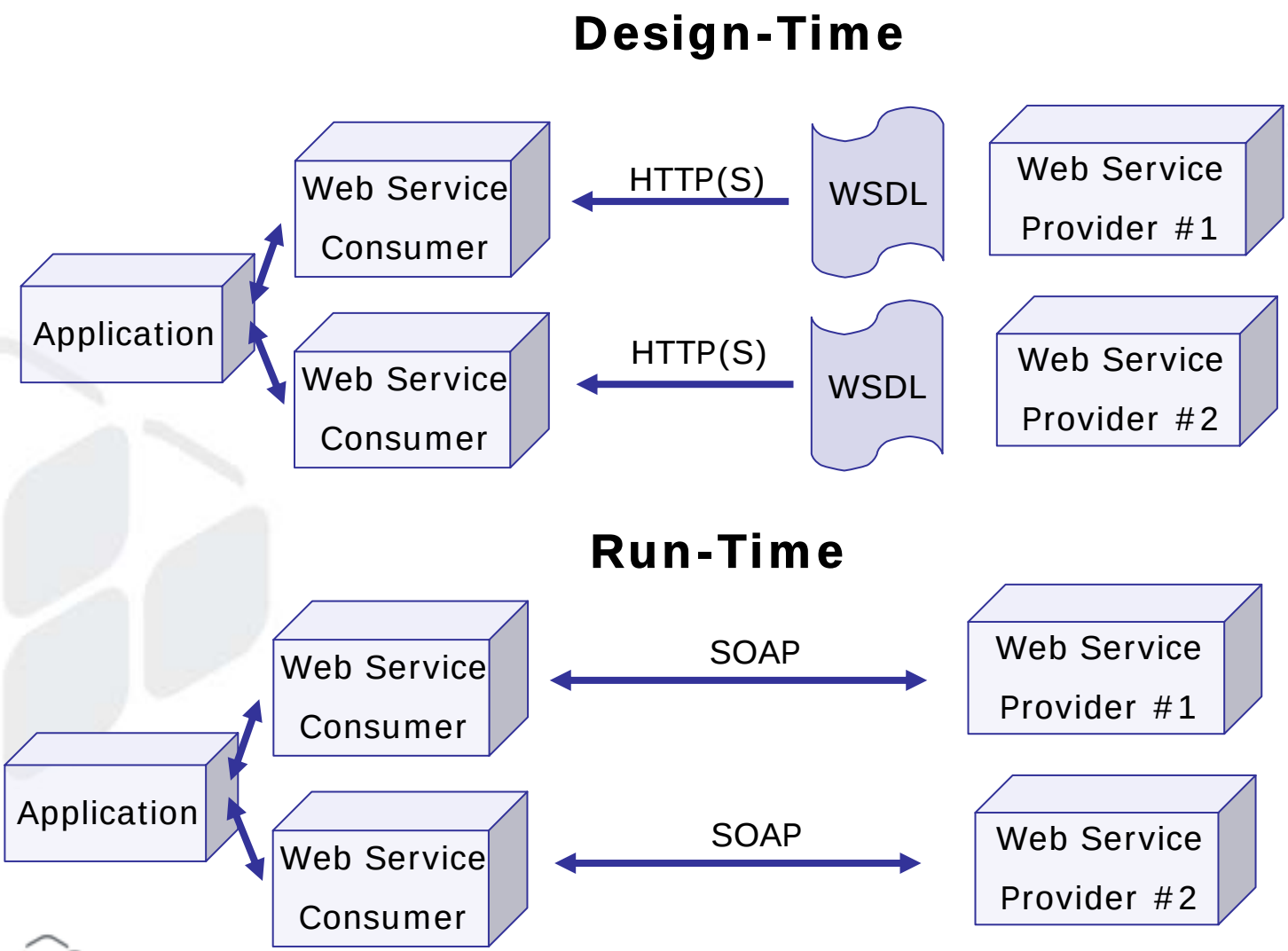
LAYER 7 TECHNOLOGIES

Web Service Provider or Server-Side



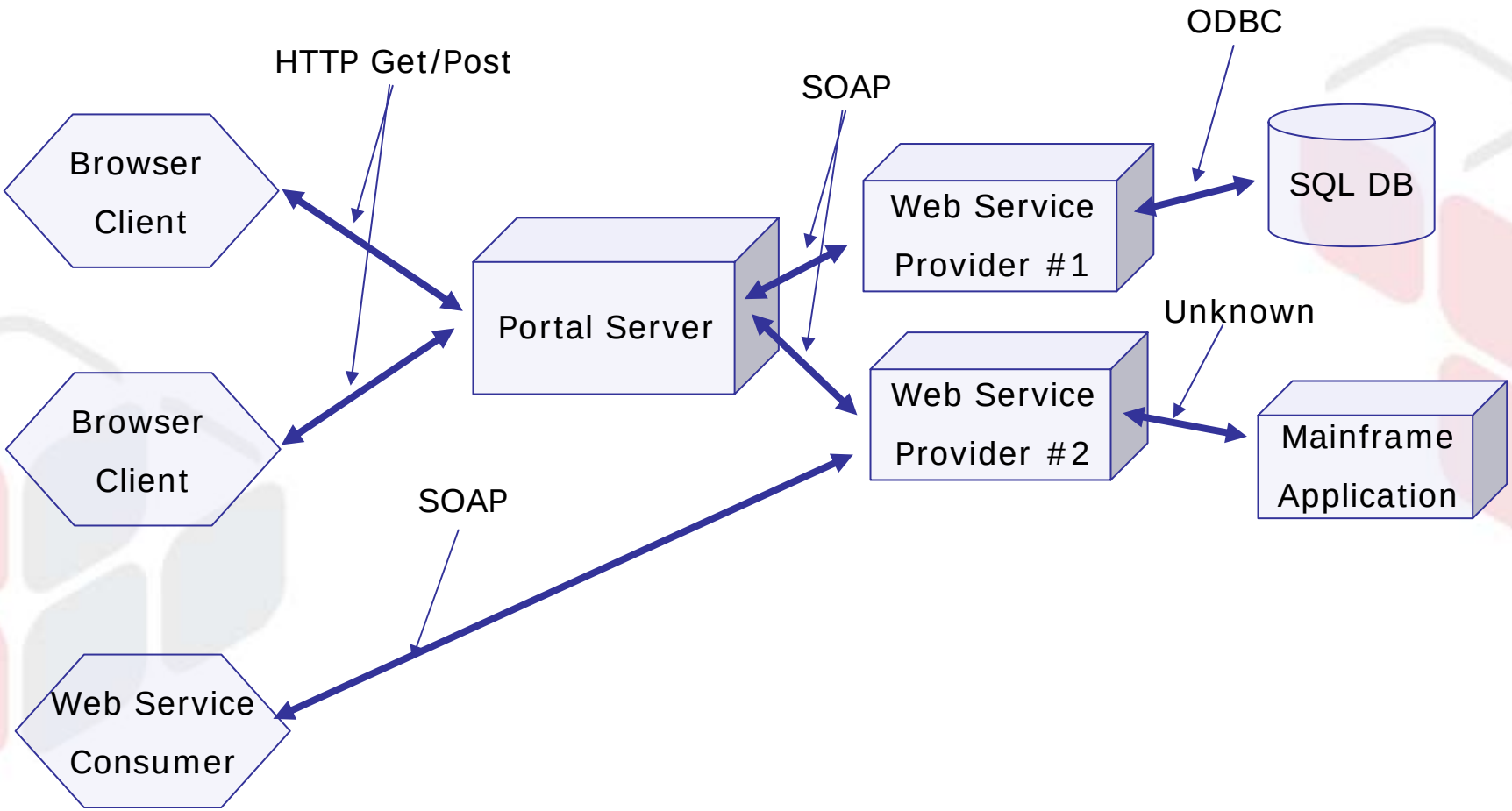
LAYER 7 TECHNOLOGIES

Web Services Consumer or Client-Side



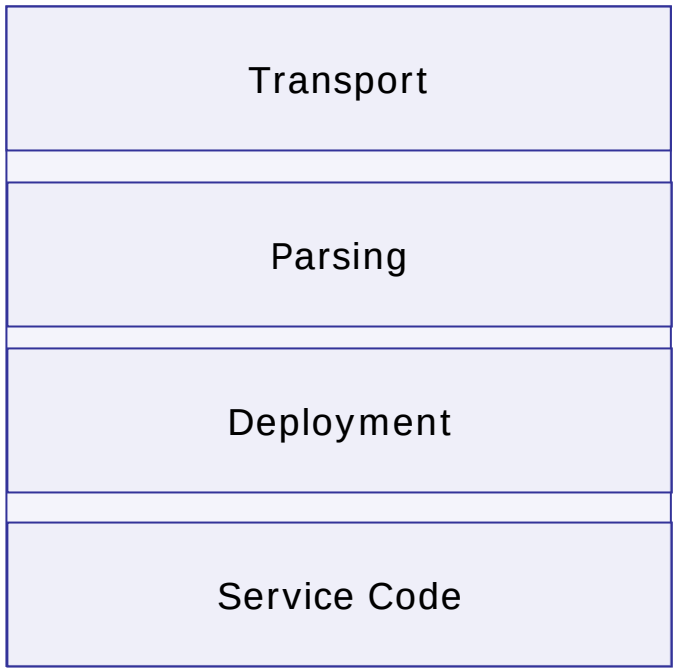
LAYER 7 TECHNOLOGIES

Common Web Services Usage



LAYER 7 TECHNOLOGIES

Web Service Threats



LAYER 7 TECHNOLOGIES

Transport Threats

Sniffing and Snooping

- Message confidentiality concerns

WS-Routing

- SOAP messages can contain verbose instructions on their desired routing. If a single node in this routing path is compromised multiple threats can be realized.

Replay Attacks

- Message integrity concerns and potential Denial of Service by taking a correct message with valid credential and sending it 1000+ times

Denial of Service

- Same old threat in regard to network Denial of Service

LAYER 7 TECHNOLOGIES

Parsing Threats

Almost all products employ the same parsers, therefore if a vulnerability exists in a single product leveraging MS Parser then all others have the same threat.

The XML specification itself does not put any restrictions on the structure itself and rather is open to interpretation by the creator of the parser. Example: Some parsers will stop reading an XML Attribute value once they reach some number of characters and others will continue.

<Name Organization="I'm a parser attack, ……>

The following will be discussed:

- Buffer, Heap, Integer Overflows
- XML Parser Attacks

LAYER 7 TECHNOLOGIES

Buffer, Heap, or Integer Overflow Threats

Warning: Through a successful buffer overflow a malicious command may be executed on your system.

We see these all the time! Through passing a malicious buffer to a Web Server or Application server the attacker can create an overflow condition where a segmentation fault occurs.

- This oversized/malicious buffer can be sent as part of the transport header OR as part of the SOAP message.
- An expected integer value can be overflowed by exceeding the value allowed causing a segmentation fault.

Once an attacker knows that a overflow is possible they can then use this to potentially execute malicious code on the system. Commonly called a buffer overflow attack.

LAYER 7 TECHNOLOGIES

XML Parser Attack Threats

The following threats can result in a denial of service commonly referred to as XML Denial of Service (XDOS) by consuming 100% of processing power on the system doing the parsing.

Complex or Recursive Payload

- Again, the XML specification and structure has no limits!
- Automated applications are available which create Fuzzed data for XDOS attacks.

Oversized Payload

- Many parsing technologies load entire documents into memory
- Web Services were generally NOT designed around large message sizes.

Other

- Unique attacks will be found where underlying parsers have vulnerabilities

LAYER 7 TECHNOLOGIES

Deployment Threats

Web Service Automation is Our Friend…..Or Is It?

UDDI, WSDL, SOAP Faults (errors), Descriptions….OH BOY!

UDDI

- UDDI contains asset information
- Automated War-Dialers (scanners) can search for UDDI's for services (i.e. Bank service found here)

WSDL

- Contains adequate information to attack service (i.e Here is how the bank service works)
- Automated programs consume WSDL and commence scanning the service (i.e. Automatically issue scanning/attack messages)

SOAP Faults

- SOAP Faults return information about the service (i.e Bank service is running on IIS version ?? and uses .Net parser)
- SOAP Faults returns errors from the backend resources such as the SQL DB, or Mainframe (i.e Bank service is using Oracle DB version ??)

LAYER 7 TECHNOLOGIES

Service Code Threats

Good development practices can alleviate this threat.
How many programs or programmers are perfect though?

Parameter Tampering

- Parameters are changed
  - <file_location>C:/INET/file.txt</file_location> changed to
  - <file_location>C:/*</file_location>

Code Injection

- Code is injected within an XML element
  - <SQL>SELECT name FROM DB1 WHERE name = 'Adam'</SQL> changed to
  - <SQL>SELECT * From DB1 WHERE name = *</SQL

Virus/Spyware/Malware Injections

- XML Attachments (MTOM, DIME, MIME) are used as a delivery mechanism for virus

Session Tampering and Identity Hijacking

- Some Web Services keep track of session with a Unique ID. Attackers can use that ID to become part of the transaction taking place.

LAYER 7 TECHNOLOGIES

Web Services Hacking

Attackers See Opportunities!

Web Services offer a entirely new dimension to the traditional security stack. This new *layer* is a business layer and current security practices DO NOT offer sufficient protection.

Why:

- Totally new technology, with new comes problems
- Operates over common web transports, traditional firewalls are based on the concept of stopping attacks at the OS level not at the Message Level (Layer 3-5).
- Automation and Toolkit development (Reuse of these tools)
- Standardization of attack vectors, you can attack .NET and Java business applications using the same messages.
- Inherent Descriptions (WSDL, Tool kit web pages, etc.)

LAYER 7 TECHNOLOGIES

A Significant Problem in System Distribution

The problem with any distributed system is that a single failure within the system can have an unknown impact on the system in its entirety.

In the use of Web Services we are adopting a practice of reuse and system distribution that spans one or more networks and potentially the internet.

LAYER 7 TECHNOLOGIES

Steps in Web Services Hacking

I've broken this tutorial down into 4 steps:

1.) Learn as much as you can about the system

2.) Do your Homework

3.) Launch the Attack

4.) Clean up after yourself

LAYER 7 TECHNOLOGIES

Step 1: Learning, Finding a Web Service

You could search a Public UDDI in this step but in most cases public UDDI information will describe services that have been created for public usage and are protected as such.

We're going to be a bit trickier here by looking for services that are not up for public consumption. The technique is known as crawling

Command: wget –l 50 –r http://bankwebserver.com

- Where "–l 50" is the maximum number of links to display
- And "–r" recursively crawls the site

Returns: You have received 27 files form the server

Command: find . –name *wsdl*

Returns: ./ws/bankservice.asmx?wsdl

We found a WSDL, now lets look to see what we can find out about the service….

LAYER 7 TECHNOLOGIES

Step 1: Learning, Examining a Web Service

```
- <wsdl:definitions targetNamespace="http://tempuri.org/">
  + <wsdl:types></wsdl:types>
  + <wsdl:message name="withdrawal_operationSoapIn"></wsdl:message>
  + <wsdl:message name="withdrawal_operationSoapOut"></wsdl:message>
  + <wsdl:message name="deposit_operationSoapIn"></wsdl:message>
  + <wsdl:message name="deposit_operationSoapOut"></wsdl:message>
  + <wsdl:message name="get_balance_operationSoapIn"></wsdl:message>
  + <wsdl:message name="get_balance_operationSoapOut"></wsdl:message>
  + <wsdl:portType name="ServiceSoap"></wsdl:portType>
  + <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap"></wsdl:binding>
  + <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap"></wsdl:binding>
  - <wsdl:service name="Service">
    - <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
        <soap:address location="http://bankwebserver.com/BankService/Service.asmx"/>
      </wsdl:port>
    - <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
        <soap12:address location="http://bankwebserver.com/BankService/Service.asmx"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

Three operations available: withdrawl, deposit, and get_balance

Where the service resides

LAYER 7 TECHNOLOGIES

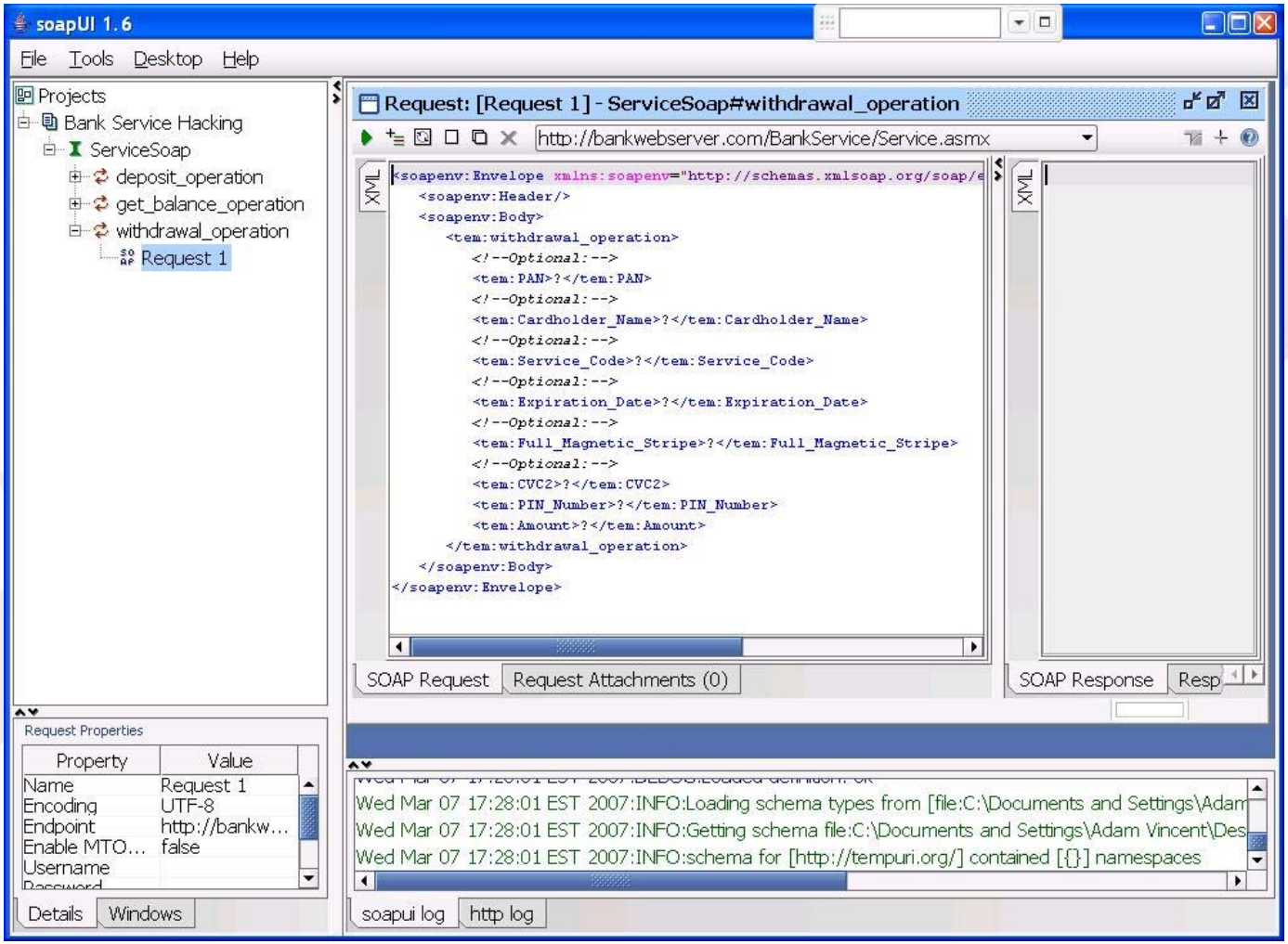
Step 1: Learning, Examining a Web Service (cont)

```
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
  - <s:element name="withdrawal_operation">
    - <s:complexType>
      - <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="PAN" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="Cardholder_Name" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="Service_Code" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="Expiration_Date" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="Full_Magnetic_Stripe" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="CVC2" type="s:string"/>
          <s:element minOccurs="1" maxOccurs="1" name="PIN_Number" type="s:int"/>
          <s:element minOccurs="1" maxOccurs="1" name="Amount" type="s:double"/>
        </s:sequence>
      </s:complexType>
    </s:element>
  + <s:element name="withdrawal_operationResponse"></s:element>
  + <s:element name="deposit_operation"></s:element>
  + <s:element name="deposit_operationResponse"></s:element>
  + <s:element name="get_balance_operation"></s:element>
  + <s:element name="get_balance_operationResponse"></s:element>
  </s:schema>
</wsdl:types>
```

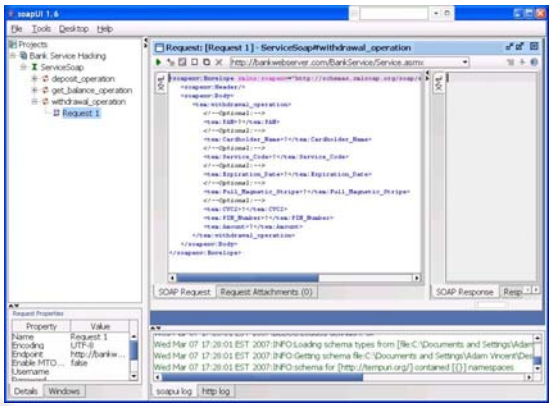
Operation parameters for withdrawl operation

LAYER 7 TECHNOLOGIES

Step 1: Learning (Attempting to Obtain Errors)



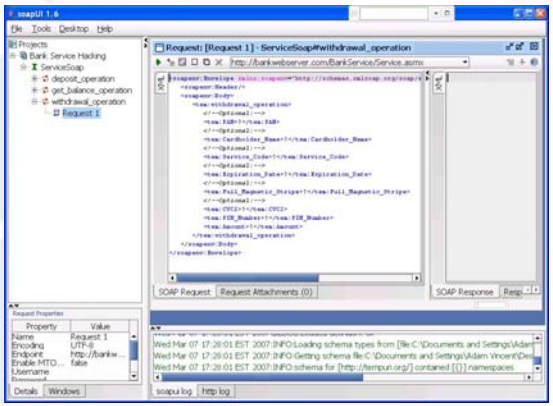
Step 1: Learning (Attempting to Obtain Errors)



Hello Bank Service

You must use me like this!

Bank Service



Ok...Hello Bank Service

You must use me like this!

Bank Service

You would continue this process while looking for areas to exploit, there are automated tools that do this for you

LAYER 7 TECHNOLOGIES

Step 1: Learning (Completed)

You now know the following:

1.) Service Location – www.bankwebserver.com

2.) Application Server Platform – IIS with .Net Version 5.0

3.) Web Service Purpose (Withdrawl, Deposit, Balance)

4.) The expected values of the request

- PAN, Cardholder_Name, Service_Code, Expiration_Date, Full_Magnetic_Stripe, CVC2, PIN_Number, and Amount.

5.) You know that the service is running

6.) The service returns errors that illustrate its not using SSL, and that it is running IIS .NET version 5.0.23.

In a real life situation you would want to know a lot more but lets continue for now.

LAYER 7 TECHNOLOGIES

Step 2: Do Your Homework

Tool of Choice: www.google.com

Research:

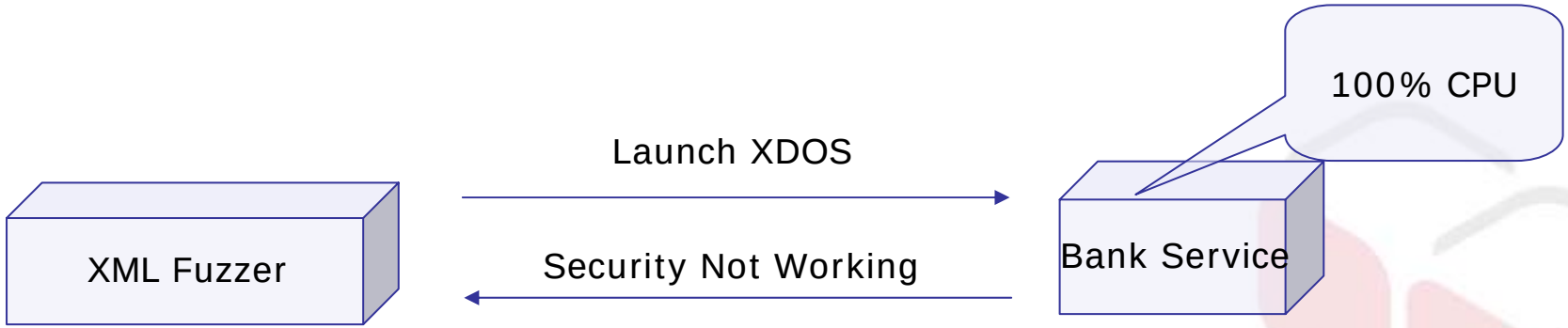
- Analyze Security capabilities in Place, Look for deficiencies
- Vulnerabilities in IIS .NET 5.0.23 application servers
- Vulnerabilities in .Net Parser's with correct version
- Analyze DOS/XDOS opportunities

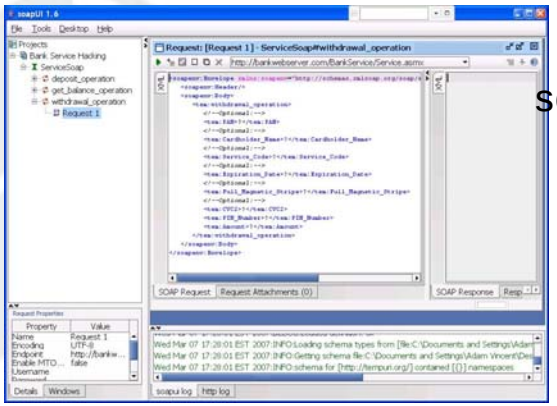
- We now would have enough information to push forward with the actual attack.

Ready Set Go!

LAYER 7 TECHNOLOGIES

Step 3: Launch the Attack





Withdrawl $1,000,000 from some account, and put in your account

Withdrawl/deposit Accomplished

LAYER 7 TECHNOLOGIES

Step 4: Clean Up After Yourself

A real hacker would be able to do some things to cover their tracks.

I'm however am not skilled enough to talk about those things

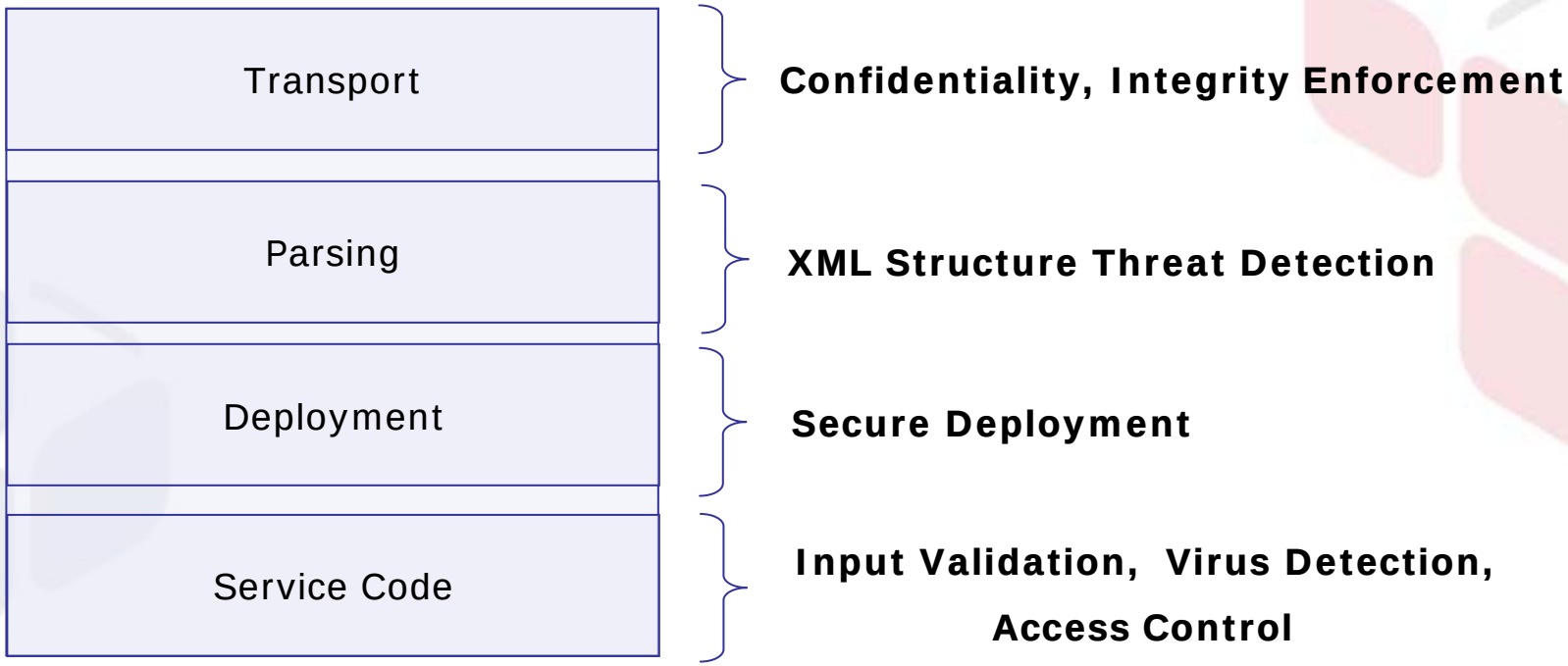
This is what I would do!

1.) Go to the Bank

2.) Leave the Country…Fast

LAYER 7 TECHNOLOGIES

Web Services Hardening



LAYER 7 TECHNOLOGIES

Confidentiality, Integrity Enforcement

Mitigate Transport Threats to Include Sniffing, Snooping, Routing Detours, and other types of transport threats

Transport Layer Encryption

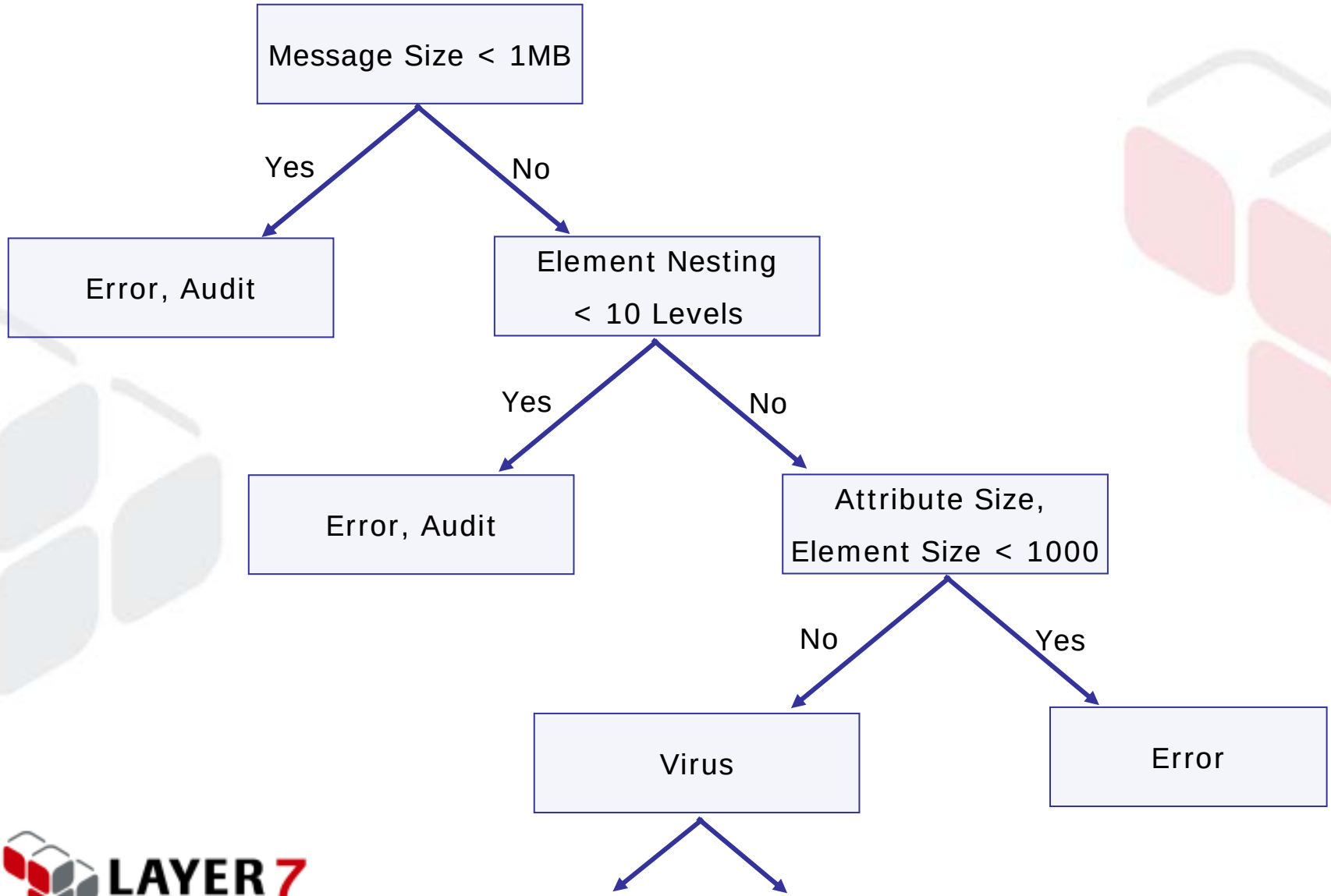
- SSL/TLS – Only good while in transit between intermediaries, does not persist from end-to-end.

WS-Security – Persists from end-to-end

- XML Encryption – Encrypted message content, does not require entire message to be encrypted.
- XML Digital Signatures – Digital signatures of message content, does not require entire message to be digitally signed.

LAYER 7 TECHNOLOGIES

XML Structure Threat Detection



LAYER 7 TECHNOLOGIES

Secure Deployment

UDDI and WSDL are like “Maps to the Treasure” and should be Treated as such. You wouldn’t leave the actual map to your treasure out in plain sight would you?

UDDI, WSDL

- Virtualize Internal Services to consumers through creation of virtual endpoints described by generalized WSDL and UDDI descriptions.

SOAP Faults and Error Messages

- Don’t allow SOAP faults and errors to be relayed to potentially malicious consumers. Generalize SOAP faults to contain no information about deployed application types and versions.

LAYER 7 TECHNOLOGIES

Input Validation (parameter tampering)

The service code layer is where development is done in creating business capabilities and is the easiest to hack. This is probably the most critical to protect.

Basic Parameter Validation

- Don't use strings as the allowed Data type. That's like allowing anything to pass.
- Validate Integer values for length

Specifically Parameter Validation

- If its supposed to be a SSN then validate it is one!
- If it's a zip code validate that its [[0-9][0-9] [0-9] [0-9] [0-9]]

XML Schema provides a tool to validate message parameters according to predetermined business usage.

LAYER 7 TECHNOLOGIES

Input Validation (code injection)

Some Code Injection protection is inherent in having a constrained schema validation on input parameters although there are some places where Schema does not suffice.

Wherever strings or more general character sets are allowed validation should be done to verify malicious code is not present.

<SQL>Some Malicous SQL Command</SQL>

Be Careful about Unicode representations of characters to avoid detection. Parsers will do funny things with these….

<blog_update>%lt;JAVASCRIPT%rt; Malicious Script </blog_update>

Be Careful with CDATA and XML Comments as XML parsers are designed to overlook these.

<![CDATA[function matchwo(a,b) { if (a < b && a < 0) then { return 1 } else { return 0 } }]]>

LAYER 7 TECHNOLOGIES

Virus Detection (virus, spyware, malware)

XML inherently does not have the ability to execute viruses rather it is a vector to which viruses can be conveyed to Web Services and backend applications for execution.

There are essentially two ways this can happen:

- Primary – SOAP with Attachments, MTOM, WS-Attachments
  - Web Service needs to either execute application stored within the SOAP attachment or issue the SOAP attachment to another system for later execution.
  - **Mediation:** Attachments should be scanned with a virus scanner, traditional virus scanning engines generally do not offer such a capability.
- Secondary – Base64 encoded malicious program
  - Web Service or other application needs to be programmed to decode BASE64 value and execute resulting binary. The program would have to have this purpose in mind in its inception to make this work.
  - **Mediation:** If this is the intended purpose for a large XML element and validation can not be accomplished, the element should be decoded and then scanned by a Virus Scanning Engine. Again Virus Engines are inadequate in this purpose

LAYER 7 TECHNOLOGIES

Access Control

Most security conscience Web Service developers employ some mechanism of authentication into deployed web service capabilities. This can be as simple as HTTP Basic or as complex as SAML Holder of Key (HOK).

Authorization can be based on accessing the Web Server itself or more specifically an operation within a service. With web services becoming more sophisticated the later is the recommended method in moving forward.

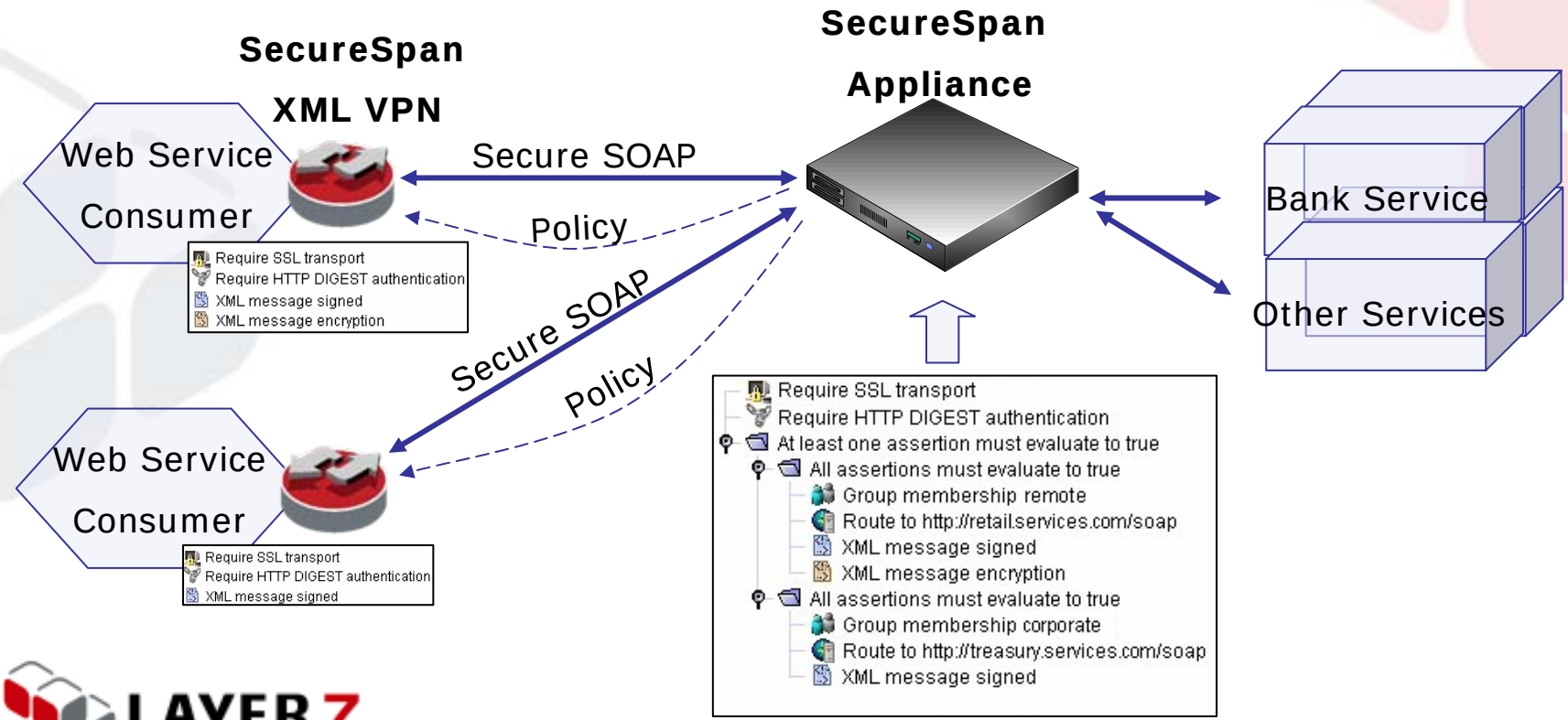
Even when access control is in place, a defense in depth approach is suggested to alleviate concern when a malicious entity has hijacked an existing authorized identity.

LAYER 7 TECHNOLOGIES

XML Appliances – All Hardening Wrapped UP

So I've suggested several hardening tactics which can be used to protect Web Services from attack. These things are not easy to implement especially with development and overhead associated with parsing and crypto operations.

The easier approach, Layer 7 to the Rescue!



Conclusion and Questions?

I'm hoping that this was a good overview for everyone!

Please feel free to contact me if you have any further questions or comments about the presentation. It's a work in progress and I'm hoping to update it based on audience feedback. You can contact me at avincent@gov.layer7tech.com or at 703-965-1771.

Questions?

Thanks For Your Time!

LAYER 7 TECHNOLOGIES