Attacking WCF Web Services

Session Objectives
- Introduction to WCF
- Tools & Techniques for Pen-testing WCF Services

Session Outline
- WCF Overview
- Silverlight WCF Web Services
- WCF and WS-Security
- Duplex Services
WTF is WCF?

- Core Communications Framework for .NET Applications and Services
  - Introduced in .NET 3.0, enhanced in .NET 3.5
  - Supports various protocol bindings and message formats
  - Includes backwards compatible for legacy services
## What’s new with WCF?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Interoperable Web Services</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Binary .NET – .NET Communication</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Distributed Transactions, etc.</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Support for WS-* Specifications</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Queued Messaging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>RESTful Communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Security Properties of ServiceHostBase

“Security is by far the most intricate area of the WCF”

-- Juval Lowy, Programming WCF Services (O’Reilly)
ABCs of WCF Endpoints

- WCF Services are exposed through **Endpoints**

- Each Endpoint has three required elements, commonly referred to as the A-B-C’s
  - **Address**
  - **Binding**
  - **Contract**
WCF Addresses

"Where can I find this service?"

Every WCF Service has a Unique Address

- Transport Protocol
- Location
- Often use .svc file extension when hosted in IIS

[transport]://[machine or domain][:optional port]/[optional uri]
WCF Bindings

"What protocol can I use to talk to this service?"

- Bindings specify how a service communicates
  - Transport Protocol
  - Encoding (Message Format)

- Several out-of-the-box bindings, or can be customized
WCF Bindings

- WCF Transport Protocols
  - NET.TCP
  - HTTP/HTTPS
  - Named Pipes (IPC)
  - Peer to Peer (P2P)
  - Message Queuing (MSMQ)

- WCF Encoding Formats
  - Text
    - SOAP, XML, JavaScript
  - Binary
  - MTOM
WCF Contracts

"What can I do with this service?"

- WCF Contracts specify what is communicated to the outside world

- Four types of Contracts
  - Service: Operations the client can perform
  - Data: Define the data types passed by the service
  - Fault: Error handling and propagation
  - Message: Allows direct interaction with messages
WCF Contracts: Opt-In Approach

- Nothing is part of a service contract by default

  - Developer must explicitly use ServiceContract and OperationContract attributes to indicate methods exposed by the endpoint

  - DataContract and DataMember attributes can also be used to specify whether all or part of a complex type is exposed to clients
Attacking WCF Services

- Example 1: Silverlight 3 Client Service
- Example 2: WS-Security & Message Encryption
- Example 3: WCF Duplex Services
Example 1: Silverlight Client Service

- WCF is commonly consumed by Silverlight for browser-based services
  - Broad Support for WCF in Silverlight 3+
  - By default, uses .NET Binary SOAP Messages
    - Smaller message sizes
    - Better messaging performance
    - Content-Type: application/soap+xml
    - MC-NBFS Protocol Specification
HTTP/S Proxies and MC-NBFS

- Limited (if any) support for MC-NBFS/MSBin1 in most common proxy tools

  - Fiddler: Binary XML Inspector (Richard Berg)
    - Read Only inspection of Binary XML Messages
Fiddler: Binary XML Inspector
MSBin1 Burp Proxy Plug-In

- Plug-In for Burp Suite Free Edition

- Burp: MSBin1 Plug-In (Gotham Digital Science)
  - Leverages Richard Berg’s XML Encoder/Decoder
  - Allows full edit/update of Binary XML Messages

- Implements processProxyMessage method of BurpExtender interface
  - Requires two chained proxy instances to perform encoding and decoding of intercepted requests
  - Sets “X-WCF-Proxy: must-encode” header to notify downstream Burp proxy
MSBin1 Burp Proxy Plug-In

Workaround for Burp Extender Limitation

- Silverlight Client
  - Burp Proxy 1
    - Decode & Edit Requests
    - Encode Edited Responses
  - Attacker
  - Burp Proxy 2
    - Decode & Edit Responses
    - Encode Edited Requests
- WCF Service
MSBin1 Burp Proxy Plug-In

Plug-In for Burp Suite Professional Edition

- Implements `processProxyMessage` and `processHttpMessage` methods of BurpExtender
  - These methods will be included in Free v1.3

- Still requires 2\textsuperscript{nd} chained proxy to edit responses
  - Above methods both invoked before response edit, not after

- Both plug-ins available for free on GDS website (after this talk)
Obtaining MetaData from a WCF Endpoint

- HTTP-GET
  - Same as legacy ASMX
  - Retrieved by appending “?wsdl” to the address

- Metadata Exchange (MEX) Binding
  - Based on WS-MetadataExchange Standard
  - W3C Working Draft 25 June 2009
You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

```c#
class Test
{
    static void Main()
    {
```

```c#
By default, WCF services do not publish metadata

- Both WSDL and MEX are enabled by default when generating WCF configuration in Visual Studio

```
[snip]
<endpoint address="mex" binding="mexHttpBinding"
    contract="IMetadataExchange"/>

[...]
<!-- To avoid disclosing metadata information, set the value below to false and remove the metadata endpoint above before deployment -->

<serviceMetadata httpGetEnabled="true"/>
[snip]
```
Basic MEX Request Structure

POST /MyService.svc/mex HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Host: wcf.example.com
Content-Length: 565

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action>
    </a:Action>
    <a:To>
      http://wcf.example.com/MyService.svc/mex
    </a:To>
  </s:Header>
  <s:Body/>
</s:Envelope>
Manual Testing Utilities

Leveraging MetaData for Manual Testing

- **WcfTestClient**
  - Ships with Visual Studio 2008+
  - Automatically Parses WSDL or MEX

- **WCF Storm**
  - Supports most WCF bindings, including MC-NBFS over HTTP
  - Free Lite Version available
Obtaining MetaData from a Silverlight XAP

- Silverlight client can be decompiled to obtain service metadata from the XAP file
  - Service Endpoints
  - Methods & Data Types

- Download, Unzip, Decompile
  - .NET Reflector
  - FileGenerator Plug-In
Example 2: Secure WCF Bindings

- Secure bindings support Message Security based on WS-Security standards
  - NetTCPBinding
    - Binary XML Message Format
  - wsHttpBinding
    - SOAP/XML over HTTP/S
  - many more...

- Multiple credentials options
  - Windows, Certificate, Username, Anonymous, IssuedToken
Determining WCF Security Settings

- Analyze Binding Security Settings
  - Primarily Driven off “Mode”
    - Transport
      - clientCredentialType
    - Message
      - clientCredentialType
    - TransportWithMessage
      - Refer to both Transport and Message settings
    - None
WCF Message Security

- Message security uses the WS-Security specification to secure messages
  - Alternative to TLS/SSL
  - Supports message signing, encryption, or both

- Message security supports negotiation by default
  - Service is dynamically asked for the correct token before any messages are exchanged
    - Can be anonymous or require credentials
    - Negotiation requires at least one certificate
WS-S Anonymous Message Encryption


- Requires a valid server certificate
  - Signed by trusted CA or in “Trusted People” store
  - Can be disabled via client endpoint behaviorConfiguration

- Certificate may be provided within meta data
  - Client -> Endpoint -> Identity -> Certificate
Writing a Custom WCF Test Client

- Much easier than it sounds
  - Usually requires less than 10 lines of custom code!!

- Use `svcutil` to generate the following artifacts using WSDL or MEX metadata:
  - `[Service Name].cs` – Client class with accessible web methods and complex data types
  - `output.config` – Configuration file with endpoint information (address, bindings, contract)
Writing a Custom WCF Test Client

Custom WCF client in less than 10 lines of code

```csharp
public class MyClient
{
    public static void Main()
    {
        try
        {
            CalculatorClient client = new CalculatorClient();
            double sum = client.Add(1, 1);
            Console.WriteLine("1 + 1 = " + sum);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```
Writing a Custom WCF Test Client

Quick and Dirty Test Client

- Step 1: Generate [class].cs and App.config
  - svcutil <metadataPath> /out:MyClient.cs /config:MyClient.exe.config

- Step 2: Add console processing logic
  - using System;
  - main()

- Step 3: Compile MyClient.cs file with csc.exe
Writing a Custom WCF Test Client

- Disabling certificate verification

```xml
<behaviors>
  <endpointBehaviors>
    <behavior name="NoCertValidation">
      <clientCredentials>
        <serviceCertificate>
          <authentication certificateValidationMode="None"
                          revocationMode="NoCheck" />
        </serviceCertificate>
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>
```
WS-S Username Credentials

- Username & Password credentials passed with each message
  - WCF does not allow this mechanism over un-encrypted transport
  - Passed in SOAP Header as defined by standards

```xml
<o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <o:UsernameToken>
    <o:Username>wcftest</o:Username>
    <o:Password>3mb3dd3d!</o:Password>
  </o:UsernameToken>
</o:Security>
```
WS-S Username Credentials

- NOTE: MetaData not always published over SSL

Default Visual Studio Template includes:

```
<serviceMetadata httpGetEnabled="true"/>
```

but NOT:

```
<serviceMetadata httpsGetEnabled="true"/>
```
Example 3: WCF Duplex Services

- WCF also supports “Duplex” communication
  - Opens “callback” channel for each client
    - WSDualHttpBinding
    - NetTcpBinding
    - NetPeerTcpBinding
  - Ideal for “push” notification
  - Callback endpoint is a listening port on the client host
WSDualHttpBinding

- WSDualHttpBinding designed for HTTP Duplex
  - Dedicated port on client machine to accept callbacks
  - Uses Microsoft-HTTPAPI/2.0

- Client informs WCF of callback address during initial request
  - WCF server will issue an acknowledgement response to callback address
Abusing WSDualHttpBinding

Port scanning via WSDualHttpBinding callback

```xml
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:a="http://www.w3.org/2005/08/addressing">
    <s:Header>
        <a:Action s:mustUnderstand="1">
            http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
        </a:Action>
        <a:MessageID>urn:uuid:foobar</a:MessageID>
        <a:ReplyTo>
            <a:Address>http://holyfield-pc:135/test</a:Address>
        </a:ReplyTo>
        <a:To s:mustUnderstand="1">
            http://gotham-vista:88/Service.svc
        </a:To>
    </s:Header>
    <s:Body>
        <CreateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm"/>
    </s:Body>
</s:Envelope>
```
Brian Holyfield
Gotham Digital Science
http://www.gdssecurity.com
labs@gdssecurity.com