



Pentesting Adobe Flex Applications

Marcin Wielgoszewski
Gotham Digital Science

OWASP

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation
<http://www.owasp.org>

Introduction

Marcin Wielgoszewski

- Security Engineer
- Gotham Digital Science
- OWASP NYNJ Metro Board Member

Intro to Flash, Flex and AIR

What is Flex and how does it differ from Flash?

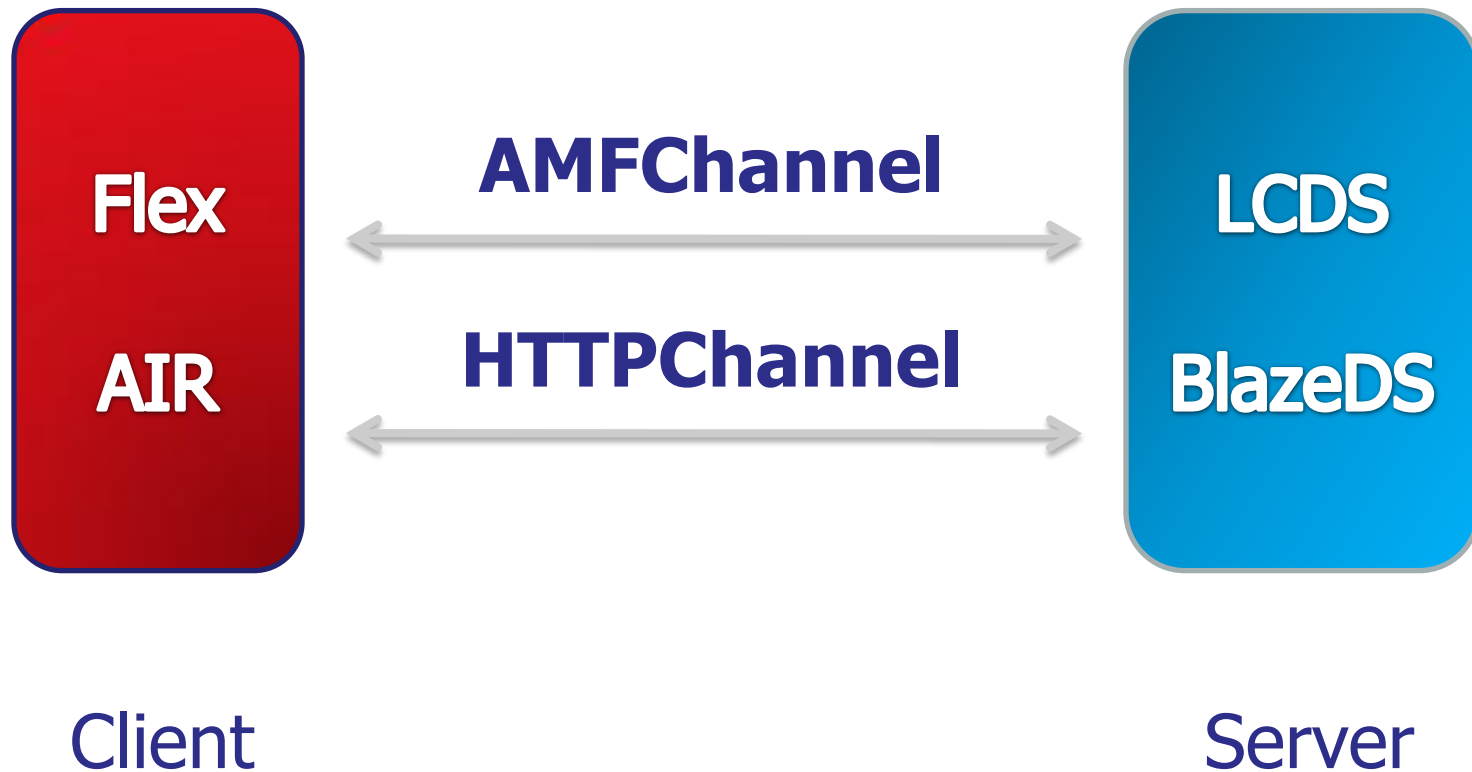
- Flash originally developed for client-side, vector-based animations and video
- Flex provides the framework for building RIA's using the Adobe Flash platform
- AIR allows developers to build desktop applications using Adobe Flash

Adobe LiveCycle DS, BlazeDS, et al.

Utilize existing application logic with Flex

- Provides remoting and messaging capabilities
- Connects backend data services
- Real-time data push to Flash clients

Client / Server Architecture



Channels

Client talks to a server *endpoint* over a Channel

- AMFChannel encapsulates data in AMF
- HTTPChannel encapsulates data in AMFX
- Streaming and Polling channels
- “Secure” channels occur over HTTPS

Endpoints

Channels route requests to a defined endpoint

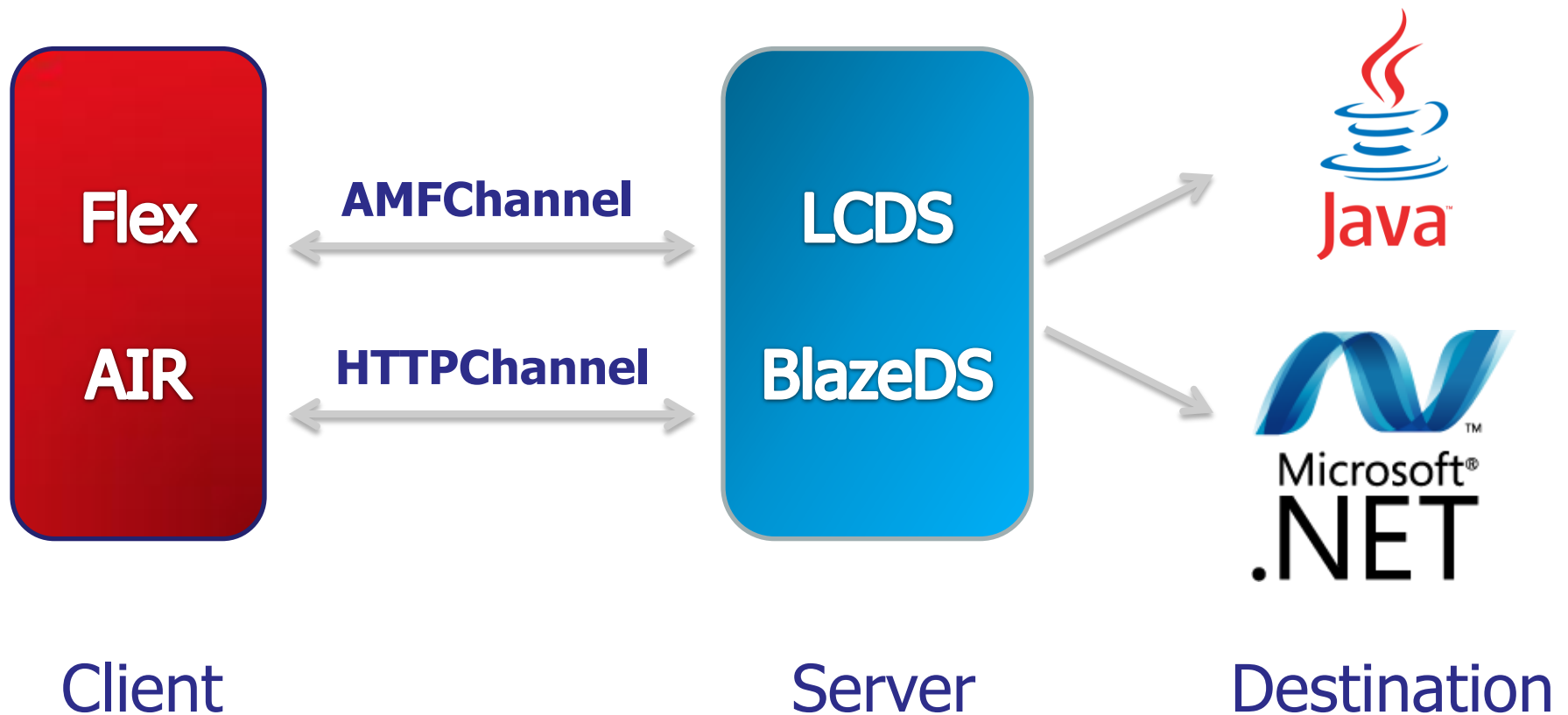
- Servlet-based, AMF, HTTP, Streaming
- Endpoint ultimately routes to a *destination*

Destinations

Here is where requests ultimately end up

- Could be one of
 - ▶ Remoting service
 - ▶ Proxy service
 - ▶ Message service

Client / Server Architecture



Action Message Format

Adobe format used for data exchange

- Used over AMFChannel/AMFEndpoints
- Requests are serialized into a compact binary format
- Responses are deserialized and processed
- 7-10x faster over XML*

Peek into AMF

AMF Envelopes contain Request Messages

- One HTTP request/response may have several AMF requests/responses
 - ▶ CommandMessage
 - ▶ RemotingMessage
 - ▶ AcknowledgeMessage
 - ▶ ErrorMessage
 - ▶ HTTPMessage / SOAPMessage

AMF over the wire

```
0x00000000: 0003 0000 0001 0004 6e75 6c6c 0002 2f31 |..... null../1|
0x00000010: 0000 0000 0a00 0000 0111 0a81 134f 666c |..... .....Of1|
0x00000020: 6578 2e6d 6573 7361 6769 6e67 2e6d 6573 |ex.messa ging.mes|
0x00000030: 7361 6765 732e 5265 6d6f 7469 6e67 4d65 |sages.Re motingMe|
0x00000040: 7373 6167 6509 626f 6479 1163 6c69 656e |ssage.bo dy.clien|
0x00000050: 7449 6417 6465 7374 696e 6174 696f 6e0f |tId.dest ination.|
0x00000060: 6865 6164 6572 7313 6d65 7373 6167 6549 |headers. messageI|
0x00000070: 6413 6f70 6572 6174 696f 6e0d 736f 7572 |d.operat ion.sour|
0x00000080: 6365 1574 696d 6554 6f4c 6976 6513 7469 |ce.timeT oLive.ti|
0x00000090: 6d65 7374 616d 7009 0101 0106 0f70 726f |mestamp. ....pro|
0x000000A0: 6475 6374 0a0b 0109 4453 4964 0649 3832 |duct.... DSId.I82|
0x000000B0: 3330 4432 3531 2d37 4231 432d 3444 3646 |30D251-7 B1C-4D6F|
0x000000C0: 2d39 3343 452d 4530 3041 3341 4237 3746 |-93CE-E0 0A3AB77F|
0x000000D0: 3441 1544 5345 6e64 706f 696e 7406 0d6d |4A.DSEnd point..m|
0x000000E0: 792d 616d 6601 0649 4533 3839 4245 4541 |y-amf..I E389BEEA|
0x000000F0: 2d46 4532 452d 3443 3745 2d42 3144 302d |-FE2E-4C 7E-B1D0-|
0x00000100: 3733 3143 4644 3130 4641 3632 0617 6765 |731CFD10 FA62..ge|
0x00000110: 7450 726f 6475 6374 7301 0101 |tProduct s... |
```

Identifying message properties

The screenshot displays the Burp Suite Professional interface. The top section shows a list of HTTP requests. The bottom section shows the details of an AMF message request and response.

#	method	URL	params	mod	status	length
16	GET	/samples/testdrive-remoteobject/index.html			200	4497
17	GET	/samples/testdrive-remoteobject/AC_OETags.js			200	8853
18	POST	/samples/messagebroker/amf			200	537
19	POST	/samples/messagebroker/amf;sessionId=18A6C95...			200	4631

type	value
AMF version 3	
body	
a target	string null
a response	string /1
a response method	string null
[] data	array
[0]	RemotingMessage
Body	array
a Operation	string getProducts
Source	null
RemoteUsername	null
RemotePassword	null
MessageId	string B218DE47-5EA7-8525-39FF-DEC3C6F5E646
ClientId	null
headers	map
a	string 56DBF984-571B-380F-89F2-79BBBF8A704F
a DSEndpoint	string my-amf
TimeToLive	number 0
a Destination	string product
1 Timestamp	number 0

The endpoint

The operation called

The destination service

The channel id



AMF RemotingMessage

Send RPC's to remote service methods

- Contain the following attributes
 - ▶ body
 - ▶ destination
 - ▶ operation
 - ▶ and more...

Flex Remoting Services

Send complex data structures to services

- Data types and object are preserved from client to server
- Client side Flash ValueObjects interact with backend POJOs

Complex Data Structures

body is an array of objects

- `body[0] = map {`
 - ▶ `String sex = "Male";`
 - ▶ `String eyeColor = "Blue";`
 - ▶ `Number age = 30;`
 - ▶ `String name = "James Bond";``}`

The screenshot displays the Burp Suite Professional interface. The top menu bar includes 'burp', 'intruder', 'repeater', 'window', and 'help'. Below the menu, there are tabs for 'target', 'proxy', 'spider', 'scanner', 'intruder', 'repeater', 'sequencer', 'decoder', 'comparer', 'options', and 'alerts'. The main window shows a table of intercepted requests with columns for '#', 'host', 'method', 'URL', 'params', 'mod', 'status', and 'length'. The selected request (ID 396) is a POST to 'http://dev.themidnight.../tourdeflex/dotnet/weborb.aspx'. Below the table, the 'response' tab is active, showing the raw AMF response. The response is an AMF version 3 object with a 'body' array. The first element of the body array is a 'RemotingMessage' object containing a 'Body' array. The 'Body' array contains a 'map' object with the following properties: 'sex' (string, 'Male'), 'eyeColor' (string, 'Blue'), 'age' (number, 30), and 'name' (string, 'James Bond'). Other properties of the 'RemotingMessage' object include 'Source' (string, 'Weborb.Examples.IdentityService'), 'Operation' (string, 'HideIdentity'), 'Timestamp' (number, 0), 'Headers' (map), 'Destination' (string, 'GenericDestination'), and 'MessageId' (string, 'F258CAB3-4579-126E-773A-ED8FF614F019').

Pentesting Adobe Flex Applications

RECONNAISSANCE

Enumerating Services and Methods

Inspect the traffic through an HTTP proxy

- Burp Suite, WebScarab, Charles, Wireshark
- Identify the
 - ▶ Destination service
 - ▶ Operation
 - ▶ Endpoint
- How many parameters (and type) are passed?

Decompiling SWFs

The beauty of having client-side code

- AS and MXML is compiled to bytecode
- Developers expose all sorts of good stuff
 - ▶ Usernames and passwords
 - ▶ URLs and connection strings
 - ▶ Hidden functionality
 - ▶ and other sensitive data

Decompiling SWFs

Common strings to look for in decompiled code

- RemoteObject | WebService | HTTPService
- .destination | .operation | .useProxy
- get | set | add | remove | create | delete

Local SharedObjects

Persistent “cookies” that reside on filesystem

- Often used to save UI preferences
- Sometimes find cool stuff
 - ▶ Session IDs
 - ▶ User/Role information
 - ▶ Sensitive data

Pentesting Adobe Flex Applications

ATTACKING REMOTING SERVICES

Calling Remoting Services

Go ahead, start calling remote methods

- What does the method do?
- How many parameters does it accept?
 - ▶ What type?
- Are there admin/hidden methods?
 - ▶ DeBlaze

Remoting Services

DEMO

Custom ValueObjects

The server complains about invalid types. WTF?

```
"Cannot convert type java.lang.String with value 'marcin' to an instance of class flex.samples.crm.employee.Employee"
```

- The SWF is binding ActionScript ValueObjects to server-side POJO's
- Simply passing a string, boolean or an integer isn't enough

Reversing a ValueObject

Well then, what do we do now?

- Identify the object's namespace
- Identify the object members that are set
- Reverse the binary object / client-side code
- Read the AMF0/AMF3 specifications, or...

Creating ValueObjects

Use PyAMF or similar API to create a VO

- Define your class and class members
- Register your class or module with a namespace
- Pass object as parameter to method

Creating Custom VO's in Python

```
# contents of mymodule.py
class Employee(object):
    def __init__(self, *args, **kwargs):
        self.firstName = kwargs.get("firstName", None);
        self.lastName = kwargs.get("lastName", None);
    ..snip..

# using mymodule in a custom AMF client
import mymodule

pyamf.register_package(mymodule, 'flex.samples.crm.employee')

employee = Employee(**{'firstName': "Marcin"})
..snip..
```

Passing Custom VO's to methods

Specify a VO instance as a method parameter

- RemotingMessage body is an array of objects
 - ▶ Your VO may be the only one, or one of many
- Remember, you're not just limited to Python
 - ▶ RubyAMF
 - ▶ AMFPHP
 - ▶ AMF::Perl

Custom ValueObjects

DEMO

Pentesting Adobe Flex Applications

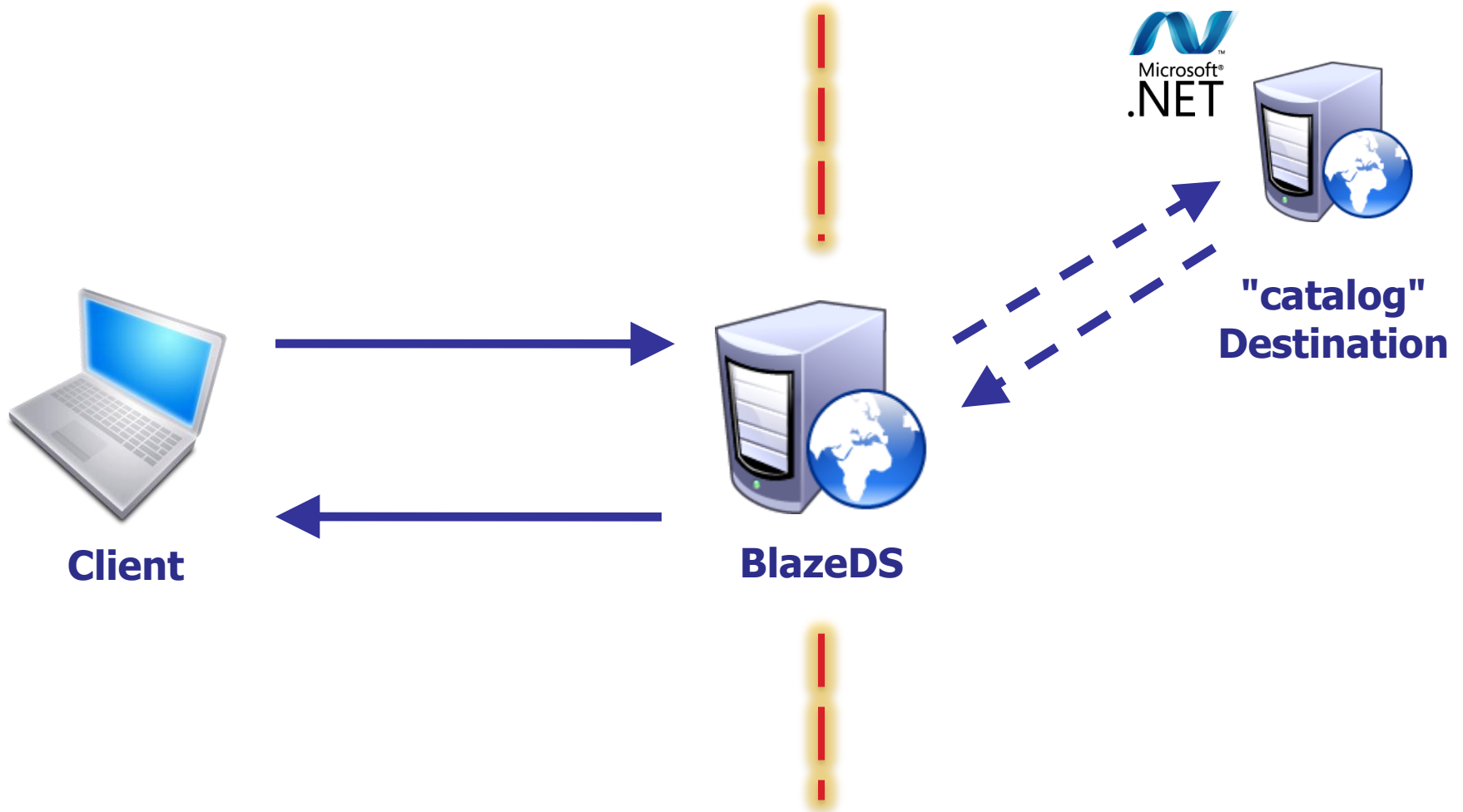
ABUSING PROXY SERVICES

BlazeDS Proxy Services

Connect Flex applications to backend services

- Request resources from another domain
- AMF/X wrapped HTTP/SOAP requests

Proxy Service Architecture



AMF HTTPMessage / SOAPMessage

Client can define a URL for BlazeDS to request

- Get around crossdomain policy restrictions
- Don't want to expose internal service publicly
- HTTP methods supported
 - ▶ GET, POST, HEAD, OPTIONS, TRACE, DELETE

Pivoting Intranets through BlazeDS

Proxy Services have inherent risks

- Proxy Services often configured insecurely
- Expose internal/Intranet apps to world
- Culprit? wildcards in *proxy-config.xml*
 - ▶ `<dynamic-url>*</dynamic-url>`
 - ▶ `<soap>*</soap>`

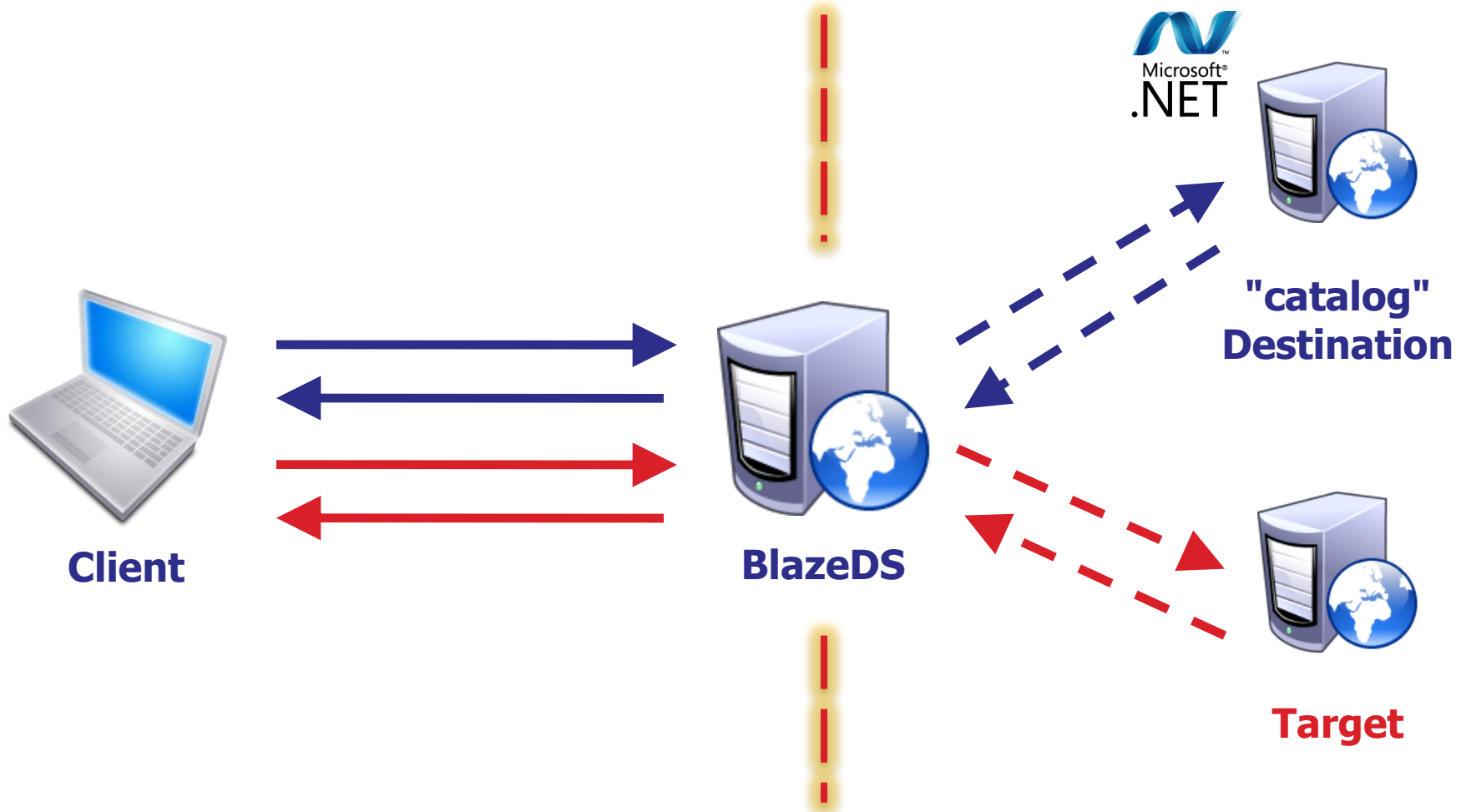
WEB-INF\flex\proxy-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<service id="proxy-service" class="flex.messaging.services.HTTPProxyService">  
..snip..
```

```
    <destination id="catalog">  
      <properties>  
        <dynamic-url>*</dynamic-url>  
      </properties>  
    </destination>
```

```
    <destination id="ws-catalog">  
      <properties>  
        <wsdl>http://livecycledata.org/services/ProductWS?wsdl</wsdl>  
        <soap>*</soap>  
      </properties>  
      <adapter ref="soap-proxy"/>  
    </destination>
```

Proxy Service Architecture

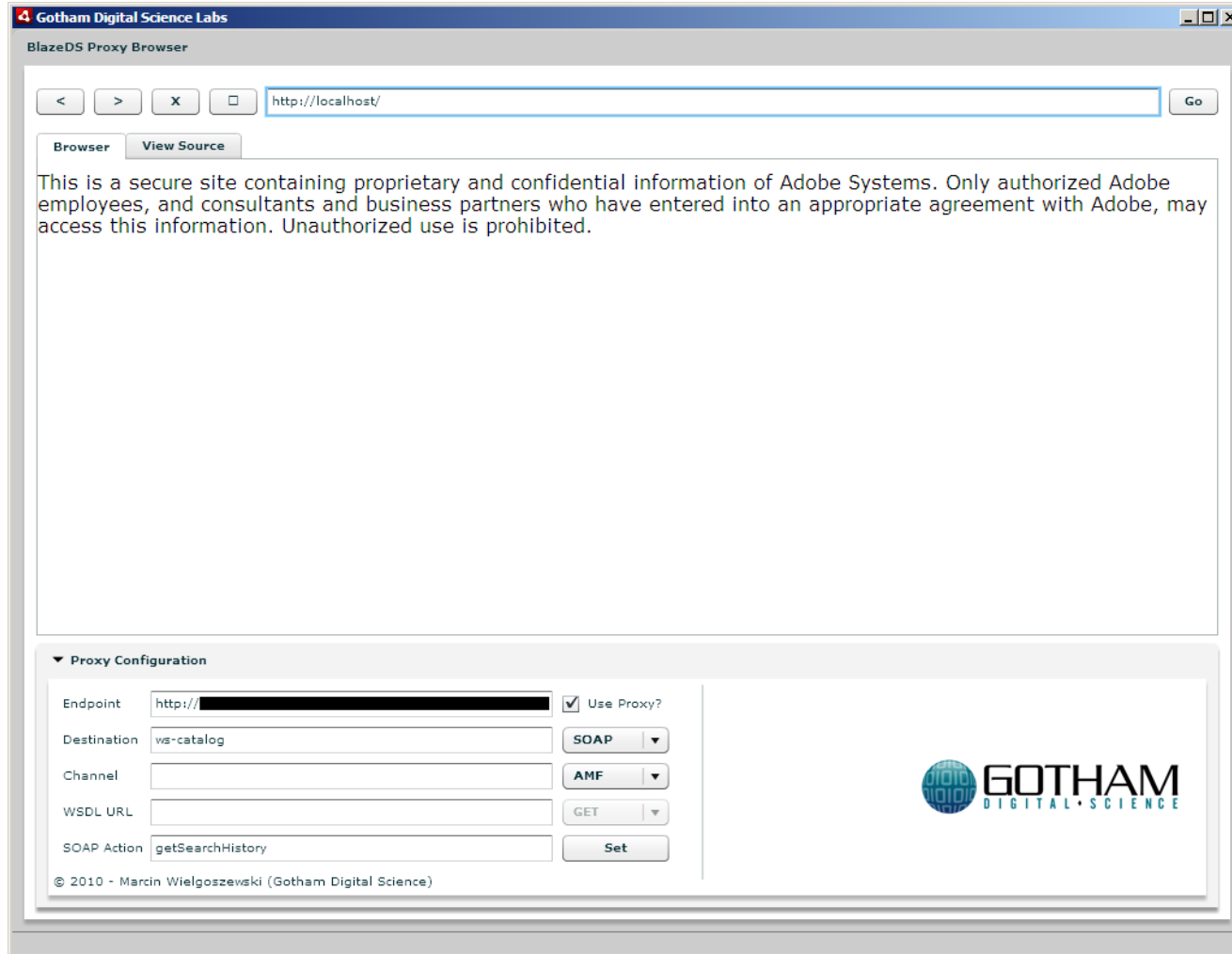


Blazentoo

A tool to exploit Proxy Services

- Browse websites reachable from server
 - ▶ Hello Intranet applications!
- Can also be a crude port scanner
 - ▶ Just specify a different port
 - ▶ Connection might be refused, reset or stay open...

Blazentoo



Blazentoo

DEMO

Flex Assessment Methodology

Let's recap:

- Passively analyze traffic
- Decompile SWF and identify stored secrets
- Enumerate services, methods & endpoints
 - ▶ Input validation, fuzzing, etc
 - ▶ Check enforcement of AuthN and AuthZ controls
- Exploit insecure configurations

References

Articles

BlazeDS Developer Guide - http://livedocs.adobe.com/blazeds/1/blazeds_devguide/

GDS Security Blog - <http://www.gdssecurity.com/l/b/>

Tools

Burp Suite - <http://portswigger.net/suite/>

Charles Proxy - <http://www.charles.com/>

DeBlaze - <http://deblaze-tool.appspot.com/>

Flex Libraries/APIs

PyAMF - <http://www.pyamf.org/>

RubyAMF - <http://rubyamf.org/>

AMF::Perl - <http://www.simonf.com/flap/>



Marcin Wielgoszewski
Gotham Digital Science
<http://www.gdssecurity.com>
labs@gdssecurity.com

QUESTIONS?

AMFX

Uses an HTTPChannel/HTTPEndpoint

- AMF objects are serialized to **XML**
- Usually provided as a fallback channel
- Different channel == different endpoint
 - ▶ URL for AMFX endpoint will differ from AMF

Message serialized to AMFX

```
<amfx ver="3"  
  xmlns="http://www.macromedia.com/2005/amfx">  
  <body>  
    <object type="flex.messaging.messages.HTTPMessage">  
      <traits>  
        <string>body</string>  
        <string>clientId</string>  
        ..snip..  
      </object>  
    </body>  
</amfx>
```

AMF CommandMessage

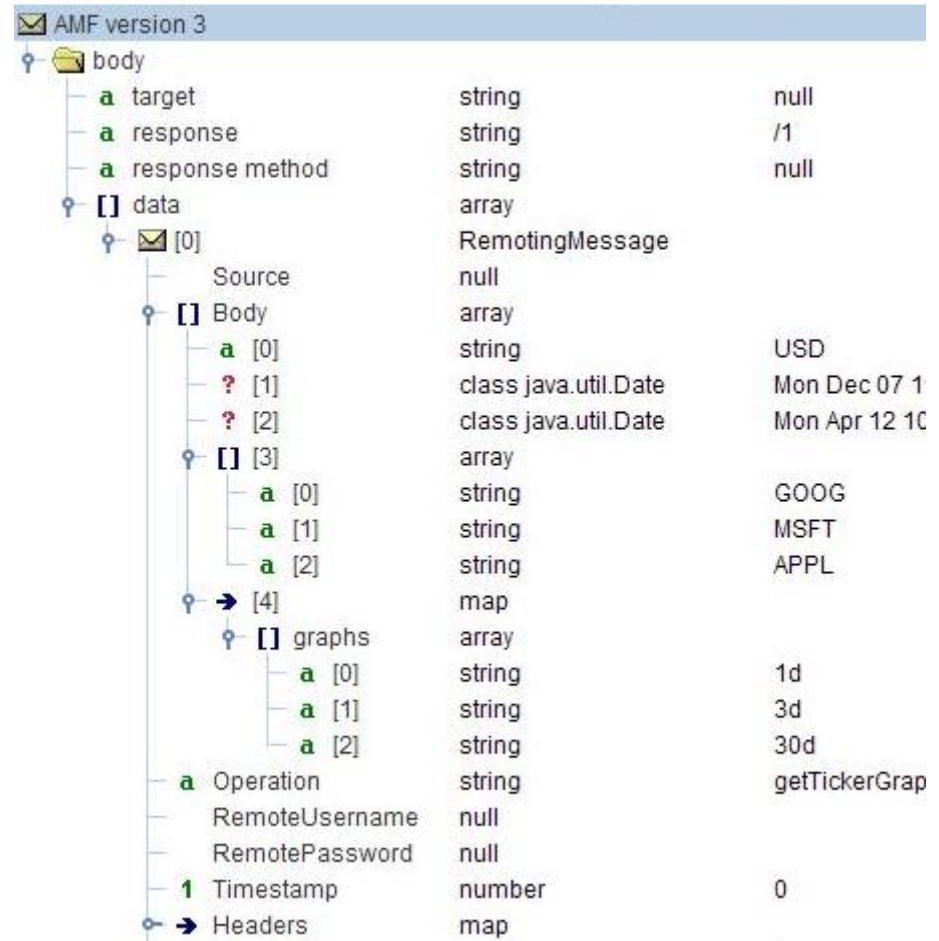
is used to... send commands!

- Mechanism for sending commands related to publish/subscribe, ping, cluster operations
 - ▶ Ping
 - ▶ Login / Logout
 - ▶ Subscribe / Unsubscribe
 - ▶ and more..

Complex Data Structures Revisited

body is an array of objects

- `body[0]` = string
- `body[1]` = `java.util.Date`
- `body[2]` = `java.util.Date`
- `body[3]` = array [
 - ▶ string, string, string]
- `body[4]` = map {
 - ▶ string, string, string }



Complex Data Structures Revisited

Check your API's language type mapping

- Python datetime = date
- Python int/float/long = number
- Python list/tuple = array
- Python dict = map