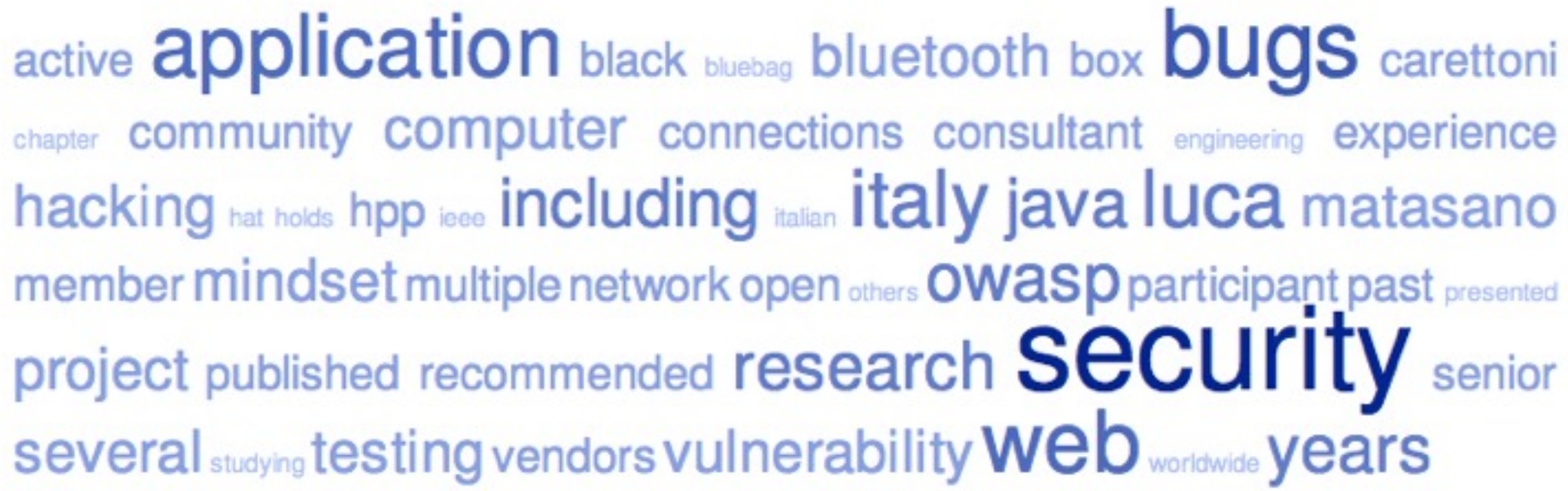
AMF Testing Made Easy!

About me

- Luca Carettoni - luca@matasano.com



Agenda

- **Context**
- **AMF specification, BlazeDS**
- **Current techniques and tools**
- **Drawbacks and limitations**
- **Blazer**
  - architecture, core techniques, heuristics
- **Testing with Blazer**
  - Objects generation and fuzzing DEMO
- **Finding vulnerabilities with Blazer**
  - Unauthenticated operations DEMO
  - SQL Injection DEMO
- **Introducing a gray-box testing methodology**
- **Conclusion**

Introduction and context

⌘ Adobe Flex

- Framework for building **R**ich-**I**nternet-**A**pplications based on Adobe Flash

⌘ ActionScript

- ActionScript is an object-oriented programming language commonly used within Adobe Flash applications

⌘ Action Message Format (AMF)

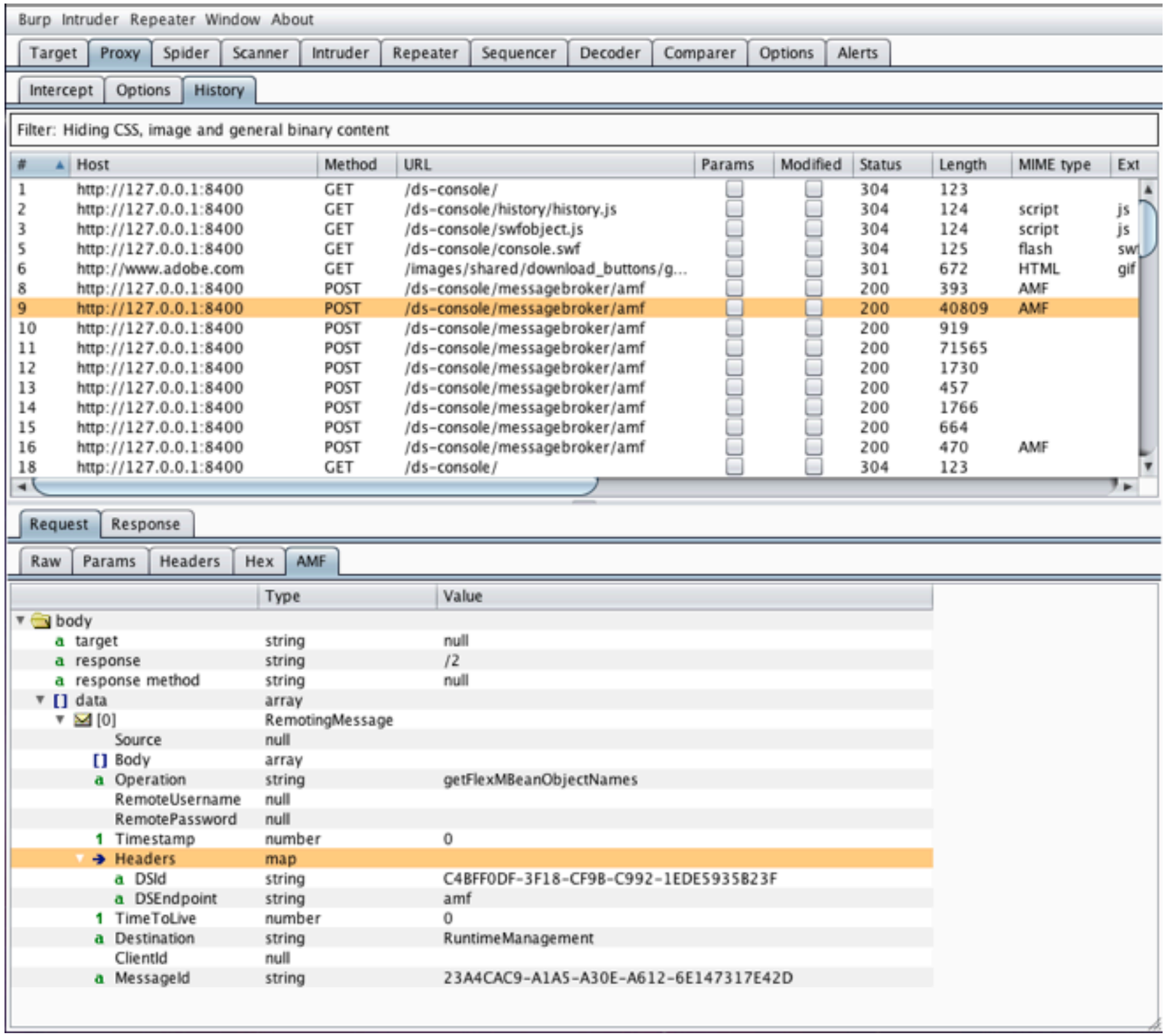
- Introduced with Flash Player 6
- Compact binary format to serialize ActionScript objects
- Fast data transfer, comparing to text-based protocols
- An efficient mechanism to:
  - **Save and retrieve application resources**
  - **Exchange strongly typed data between client-server**

AMF for end-users

AMF for old-school hackers

```
0x0230:  0d0a 0003 0000 0001 0004 6e75 6c6c 0003  ..........null..
0x0240:  2f34 3200 0002 220a 0000 0001 110a 8113  /42..."".........
0x0250:  4f66 6c65 782e 6d65 7373 6167 696e 672e  Oflex.messaging.
0x0260:  6d65 7373 6167 6573 2e52 656d 6f74 696e  messages.Remotin
0x0270:  674d 6573 7361 6765 0d73 6f75 7263 6513  gMessage.source.
0x0280:  6f70 6572 6174 696f 6e09 626f 6479 1163  operation.body.c
0x0290:  6c69 656e 7449 6417 6465 7374 696e 6174  lientId.destinat
0x02a0:  696f 6e0f 6865 6164 6572 7313 6d65 7373  ion.headers.mess
0x02b0:  6167 6549 6415 7469 6d65 546f 4c69 7665  ageId.timeToLive
0x02c0:  1374 696d 6573 7461 6d70 0106 1b67 6574  .timestamp...get
0x02d0:  4174 7472 6962 7574 6573 0905 0106 8231  Attributes.....1
0x02e0:  666c 6578 2e72 756e 7469 6d65 2e42 6c61  flex.runtime.Bla
0x02f0:  7a65 4453 3a48 5454 5050 726f 7879 5365  zeDS:HTTPProxySe
0x0300:  7276 6963 653d 7072 6f78 792d 7365 7276  rvice=proxy-serv
0x0310:  6963 652c 4d65 7373 6167 6542 726f 6b65  ice,MessageBroke
0x0320:  723d 4d65 7373 6167 6542 726f 6b65 7231  r=MessageBroker1
0x0330:  2c69 643d 4465 6661 756c 7448 5454 502c  ,id=DefaultHTTP,
0x0340:  7479 7065 3d4d 6573 7361 6765 4272 6f6b  type=MessageBrok
0x0350:  6572 2e48 5454 5050 726f 7879 5365 7276  er.HTTPProxyServ
0x0360:  6963 652e 4854 5450 5072 6f78 7944 6573  ice.HTTPProxyDes
0x0370:  7469 6e61 7469 6f6e 0909 0106 1f49 6e76  tination.....Inv
0x0380:  6f6b 6548 5454 5043 6f75 6e74 061f 496e  okeHTTPCount..In
0x0390:  766f 6b65 534f 4150 436f 756e 7406 2749  vokeSOAPCount.'I
0x03a0:  6e76 6f6b 6548 5454 5046 7265 7175 656e  nvokeHTTPFrequen
0x03b0:  6379 0627 496e 766f 6b65 534f 4150 4672  cy.'InvokeSOAPFr
0x03c0:  6571 7565 6e63 7906 4943 3443 3936 4541  equency.IC4C96EA
0x03d0:  392d 3944 3039 2d33 4243 332d 3238 4532  9-9D09-3BC3-28E2
0x03e0:  2d35 3233 3539 3734 3344 3735 3706 2352  -52359743D757.#R
0x03f0:  756e 7469 6d65 4d61 6e61 6765 6d65 6e74  untimeManagement
0x0400:  0a0b 0109 4453 4964 0649 4334 4339 3634  ....DSId.IC4C964
0x0410:  4535 2d39 4431 382d 3637 4435 2d31 4133  E5-9D18-67D5-1A3
0x0420:  442d 4132 3144 3642 4534 3933 3535 1544  D-A21D6BE49355.D
0x0430:  5345 6e64 706f 696e 7406 0761 6d66 0106  SEndpoint..amf..
0x0440:  4941 3739 3032 3034 452d 3842 3538 2d30  IA790204E-8B58-0
0x0450:  3244 452d 3432 3838 2d36 4531 3931 3534  2DE-4288-6E19154
0x0460:  4634 4636 3404 0004 00                   F4F64....
```

AMF for web hackers



AMFv0 versus AMFv3

- Flash Player 6
- Object instances can be sent by reference
- Support for ActionScript 1.0

- Flash Player 9
- Object instances, traits and strings can be sent by reference
- Support for new ActionScript 3.0 data types
- Support for `flash.utils.IExternalizable`
- Variable length encoding scheme for integers

Adobe BlazeDS

- Server-side Java Remoting/Messaging technology
- Using Flex Remoting, any Flex client or AIR application can communicate with remote services and inter-exchange data
- In practice, clients invoke Java methods from classes deployed within a traditional J2EE application server (e.g. Apache Tomcat)
- A widely deployed implementation
- Multiple alternatives exist:
  - Java: Adobe LiveCycle Data Service, Granite, ...
  - Others: RubyAMF, FluorineFX, amfPHP, ...

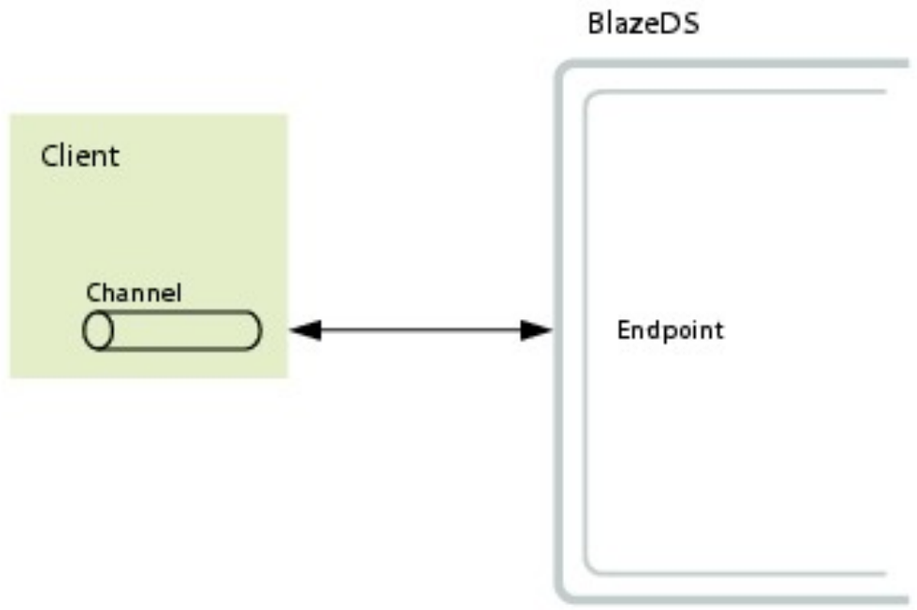
Action Message Format (AMF)

⌘ AMF request/response types:

- CommandMessage
- **RemotingMessage**
- AcknowledgeMessage
- ErrorMessage
- HTTPMessage / SOAPMessage

⌘ Client-Server communication through channels:

- Endpoint
  - **e.g. http://<>/messagebroker/amf**
- Destination Service
  - **e.g. echoService**
- Operation
  - **e.g. String echo(String input)**



State of art (research, tools)

- **Testing Flash Applications, OWASP AppSec 2007**
  - Stefano di Paola
- **Flex, AMF3 And Blazeds - An Assessment, Blackhat USA 2008**
  - Jacob Karlson and Kevin Stadmeyer
- **Deblaze, Defcon 17**
  - Jon Rose
  - Deblaze: remote method enumeration tool for Flex servers
- **Pentesting Adobe Flex Applications, OWASP NY 2010**
  - Marcin Wielgoszewski
  - Blazentoo: a tool to exploit proxy services
- **Starting from Burp Suite 1.2.124**
  - Allows to visualize and tamper AMF requests and responses
- **Other debugging tools**
  - Charles Proxy, WebScarab, Pinta AIR app, ...

Testing remote methods, today

⌘ Traffic inspection and tampering

- Using network packet analyzers
- Using HTTP proxies

⌘ Enumeration (black-box testing)

- Retrieving endpoints, destinations and operations from the traffic
- Decompiling the Flex application
- Brute-forcing endpoint, destination and operation names

Life is pain, highness. Anyone who tells you differently is selling something

W. Goldman

Is this the best we can do?

PRO

- Ideal for black-box testing, limited knowledge required

CONS

- Time consuming
- Requires to invoke all application functionalities
- What about custom objects?
- What about “hidden” services?
- How to ensure coverage?

Enterprise-grade applications

- Large attack surface
- Custom externalizable classes
- We have seen applications with more than 500 remote invokable methods and more than 600 custom Java objects

Request | Response

Raw | Params | Headers | Hex

```
POST /samples/messagebroker/amf HTTP/1.1
Host: 127.0.0.1:8400
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:13.0) Gecko/20100101 Fi
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Cookie: JSESSIONID=ABD7C9C82A4A1CAD9615C63C59580110
Referer: http://127.0.0.1:8400/samples/inventory/inventory.swf/[[DYNAMIC]]/6
Content-type: application/x-amf
Content-Length: 466

 null /4æ

Å Oflex.messaging.messages.RemotingMessage source operation      body clientId heade
update
s9flex.samples.product.Product price productId category description qtyInStock image
Italy!   merlot.jpg      Wine ID4D74179-370A-21E5-AD9C-24FB6A2889B8
 DSEndpoint  my-amf      DSId ID4D7373D-9619-76FB-1C1F-87D8A45EA531 I0647C66C-9265-4
```

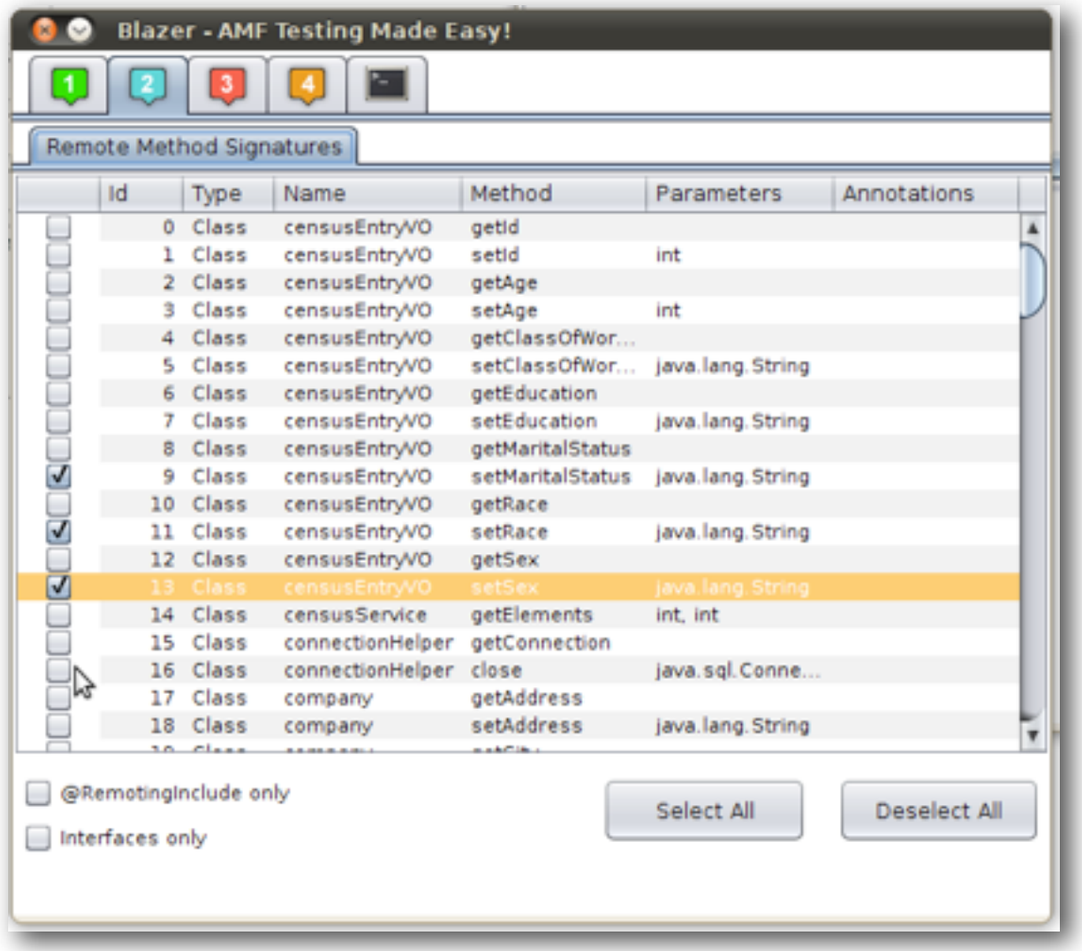
Security Testing Areas

- **Authentication**
  - Missing authentication
  - Authentication bypass flaws
  - ...
- **Authorization and Access Control**
  - Direct object reference bugs
  - Horizontal and vertical escalation bugs
  - ...
- **Error Handling**
  - Information disclosure via stack traces
  - ....
- **Input Validation**
  - SQL Injection vulnerabilities
  - ...
- **Output Encoding**

Life is not #000000 and #ffffff

Say hello to Blazer

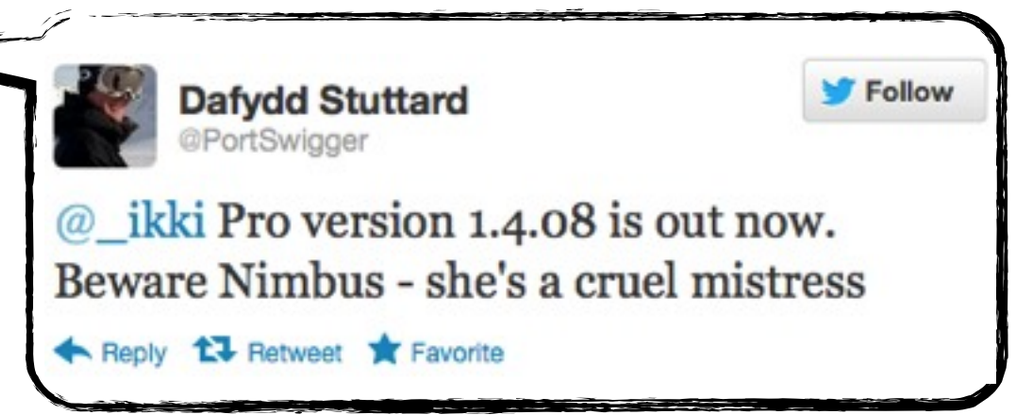
- Custom AMF message generator with fuzzing capabilities
- Method signatures and Java reflection are used to generate dynamically valid objects



Blazer v0.2

⌘ GUI-based Burp Suite plugin

- Well-integrated so you won't need to leave your favorite tool
- Burp Free and Pro
- With Nimbus look'n'feel too

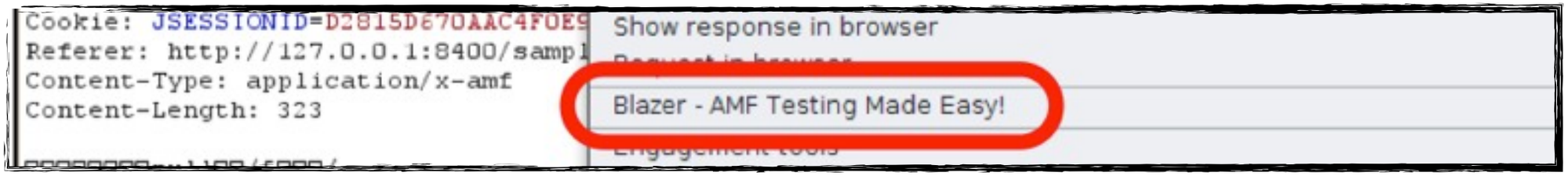


⌘ GNU GPL software

⌘ Start Burp with

- java -classpath burp.jar:Blazer_v0.2.jar burp.StartBurp

and launch Blazer from the context menu



Blazer - Architecture 1/2

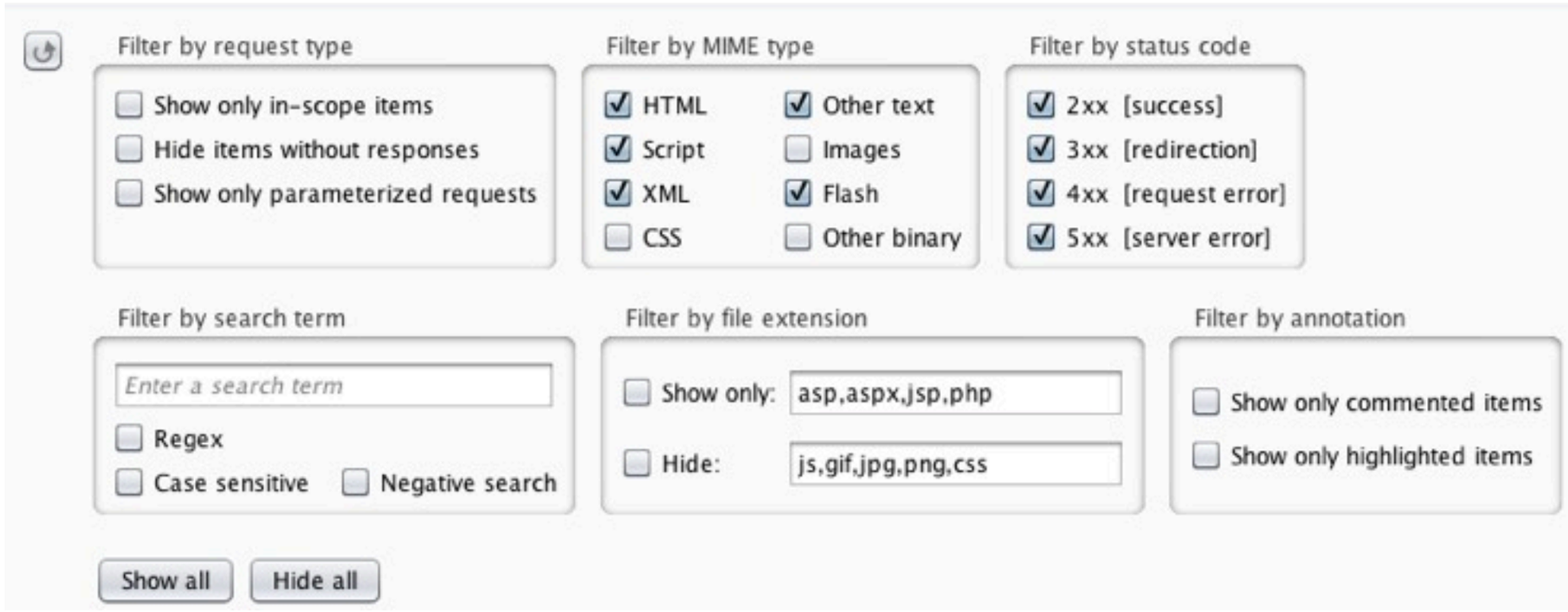
- **A packet generator**
  - based on Adobe AMF OpenSource libraries
- **An object generator**
  - to build valid application objects using “best-fit” heuristics
- **A lightweight fuzzing infrastructure**
  - to generate attack vectors, insert payloads within objects, manage multiple threads and monitor the progress

- **Blazer’s core classes are:**
  - com.mtso.blazer.ObjectGenerator
  - com.mtso.blazer.MessageGenerator
  - com.mtso.blazer.TaskManager
  - com.mtso.blazer.MessageTask

Blazer - Architecture 2/2

⌘ By default, Blazer uses Burp Proxy to record requests and responses

- so that you can benefit from all Burp's built-in tools available



⌘ Users can configure their custom proxy:

- Another instance of Burp to avoid a "collapse" during fuzzing
- A better tool for displaying AMF messages

DEMO 1

⌘ General usage

1. Application Libraries
2. Remote Method Signatures
3. General Options and Data Pools
4. Status
5. (Optional) BeanShell

⌘ Objects generation

⌘ Manual testing using BeanShell

```
$ MessageGenerator myGen = new
MessageGenerator("127.0.0.1","8080",ENDPOINT,"");
$ MessageSkeleton message = new MessageSkeleton(SERVICE, OPERATION);
$ message.addPar("BH");
$ myGen.send(message);
```

DEMO 2

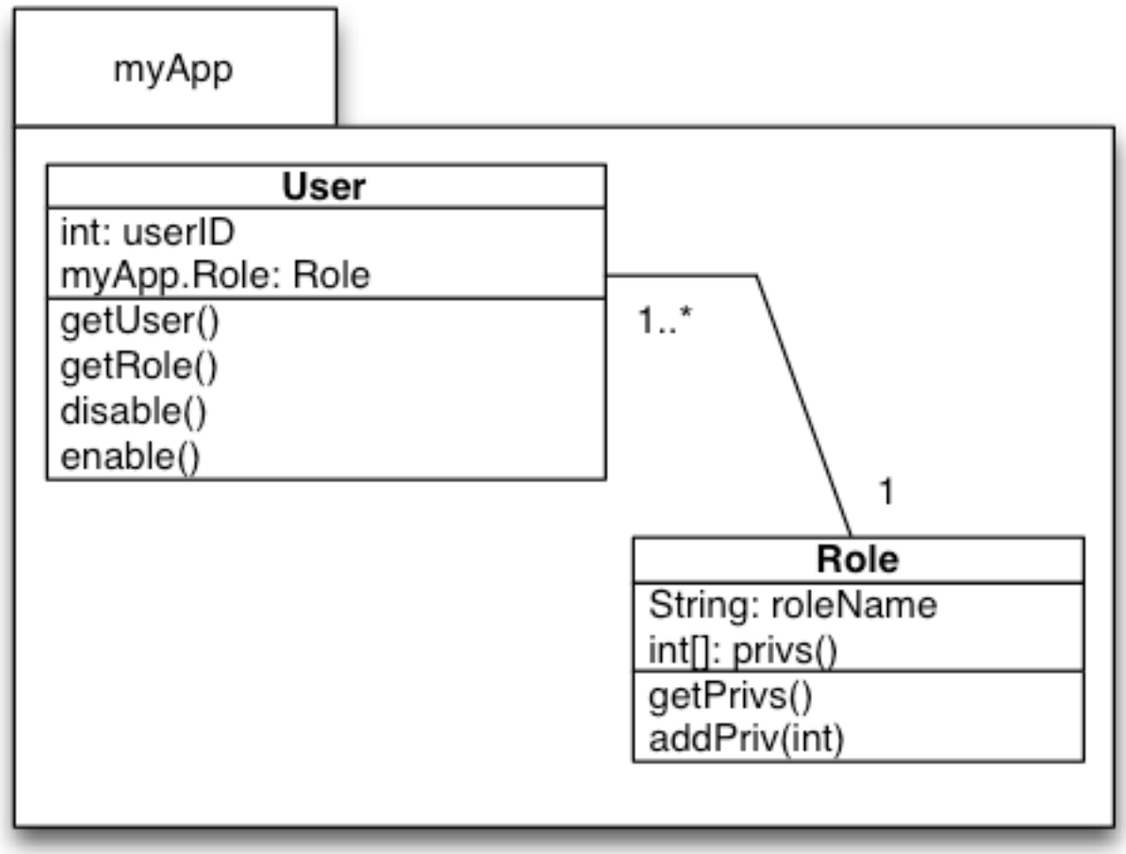
⌘ Finding vulnerabilities with Blazer

- Test case: Discover unauthenticated and exposed operations

Blazer - Core techniques 1/3

⌘ Objects generation

- Java reflection
- “Best-fit” heuristics
- Randomness and permutations



Blazer - Core techniques 2/3

⌘ Data Pools

- Containers for “good” user-supplied input
- Allow to instantiate objects and invoke methods with semantically valid data
- Available for all primitive types and String
- Require to be customized for the target

⌘ Attack vectors

- Relevant for String objects only
- Attack vector’s probability allows to unbalance the String data pool with attack vectors

Blazer - Core techniques 3/3

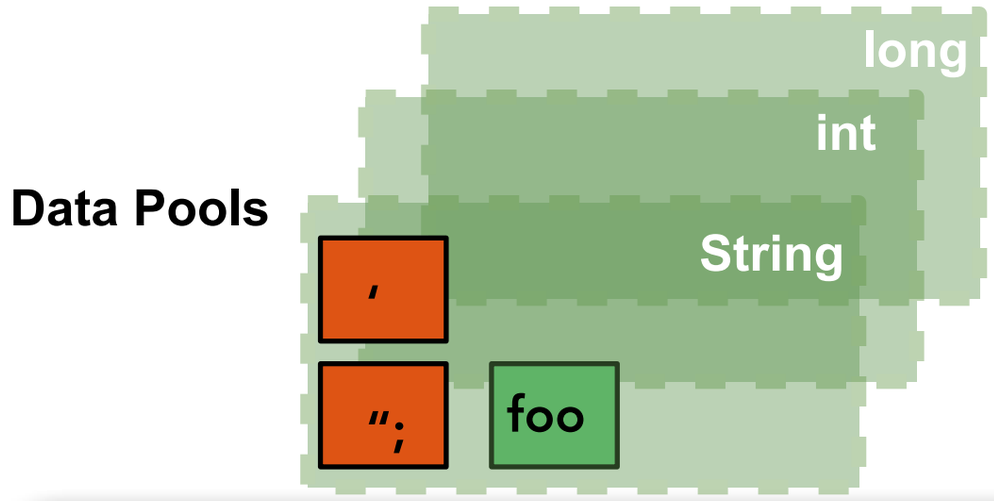
∀ Attack vector

∀ Destination (classes)

∀ Operation (methods)

```
while (numPerm < maxPerm){
   generateObject(signature);
   sendObject();
}
```

Thread




```
Object generate(String signature){
  if ( int ){
     getIntFromPool();
  } else if ( java.lang.String ){
     getStringFromPool();
  }
  ... else {
    //Build the obj
    obj = fc.newInstance();
    //Populate obj using internal methods
    //Call recursively generate(newSign)
}}
```

Test case: SQL injection

```
public List getProductsByHash(HashMap paramHashMap)
    throws DAOException
  {
    String str = (String)paramHashMap.get("key");

    ArrayList localArrayList = new ArrayList();
    Connection localConnection = null;
    try
    {
      localConnection = ConnectionHelper.getConnection();
      PreparedStatement localPreparedStatement =
localConnection.prepareStatement("SELECT * FROM product WHERE UPPER("+str+")");
      ...
```

Blazer - “Best-fit” heuristics 1/2

- For example, let’s build a HashMap

```
//Generation only
ObjectGenerator tCObj = new ObjectGenerator(task, null);
tCObj.generate(“java.util.HashMap”);
```

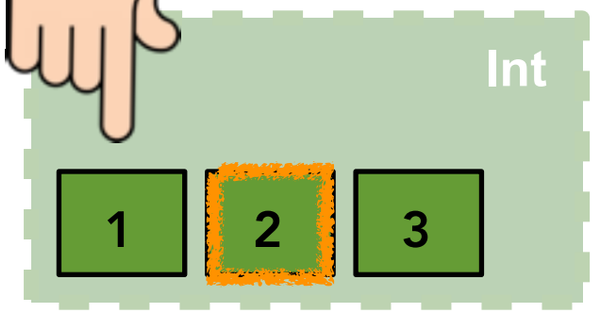
Constructor Summary

HashMap()
Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).

HashMap(int initialCapacity)
Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).

HashMap(int initialCapacity, float loadFactor)
Constructs an empty HashMap with the specified initial capacity and load factor.

HashMap(Map m)
Constructs a new HashMap with the same mappings as the specified Map.

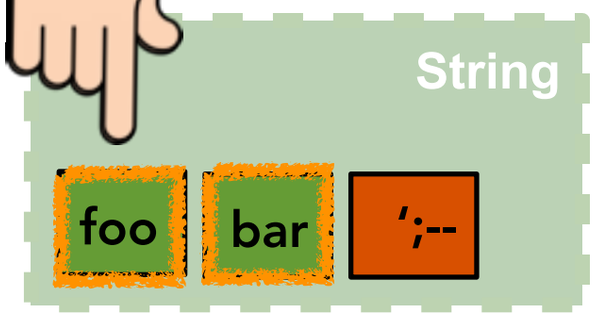


Blazer - “Best-fit” heuristics 2/2

{foo=bar,null}

Method Summary

void	**clear**() Removes all mappings from this map.
Object	**clone**() Returns a shallow copy of this `HashMap` instance: the keys and values themselves are not cloned.
boolean	**containsKey**(Object key) Returns `true` if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) Returns `true` if this map maps one or more keys to the specified value.
Set	**entrySet**() Returns a collection view of the mappings contained in this map.
Object	**get**(Object key) Returns the value to which the specified key is mapped in this identity hash map, or `null` if the map contains no mapping for this key.
boolean	**isEmpty**() Returns `true` if this map contains no key-value mappings.
Set	**keySet**() Returns a set view of the keys contained in this map.
Object	**put**(Object key, Object value) Associates the specified value with the specified key in this map.



DEMO 3

- **Finding vulnerabilities with Blazer**
  - Test case: SQL injection

Coverage and Scalability

- **With unlimited time, you could get theoretically close to 99.9% coverage**
- **In practice, Blazer and target setup are crucial**
  - Optimize the number of permutations, depending on method's arguments complexity
  - Balance "good" and "bad" attack vectors
- **During big assessments, time matters!**
- **Let's do some math:**
  - Application with **~500** exposed operations
  - **45** attack vectors (Burp's default fuzzing list in Intruder)
  - **35** permutations (average for big apps, experimentally determined)
  - ~500 x 45 x 35 = **~787500 reqs**
  - Let's assume a conservative **~80 reqs/sec**
  - It's **~2.7** hours for a full run with a relatively small wordlist

AMF Security Testing with Blazer

⌘ Authentication

- Generate valid messages and verify that all operations require a valid session token
- ...

⌘ Authorization and Access Control

- Generate valid messages with different session tokens
- Customize the data pools, generate objects with multiple permutations and look for DOR bugs
- ....

⌘ Error Handling, Input Validation

- Perform fuzzing with your favorite wordlists and manually validate the results:
  - **Sorting by response size**
  - **Filtering by search term with regex**
  - **Exporting responses and using your custom “grep” tool**
  - **...**

Conclusions

- During real-life assessment, the approach has been proven to increase coverage and effectiveness
- Blazer was designed to make AMF testing easy, and yet allows researchers to control fully the entire security testing process
- From 0 to message generation and fuzzing in just few clicks
- Using Blazer's internal classes, it is possible to build and send AMF messages from your favorite scripting language

Future improvements

⌘ Upcoming:

- Allow import of source code and classes from entire directories
- Sandbox **`com.mtso.blazer.ObjectGenerator`** to avoid dangerous methods execution while generating custom objects

⌘ Wish-list:

- Embed an utility for displaying complex AMF messages
- Auto-selection of remote method signatures for requests already present in Burp History
- Auto-save of recent Blazer's configurations

That's all folks!

- **Questions? Critics? Suggestions?**
  - Now!
  - By email: **luca@matasano.com**
  - By Twitter: **@_ikki**

- **Please complete the Speaker Feedback Surveys**

References 1/2

- **AMF 3 Specification, Adobe Systems Inc.**
  http://download.macromedia.com/pub/labs/amf/amf3_spec_121207.pdf
- **Adobe BlazeDS Developer Guide, Adobe Systems Inc.**
  http://livedocs.adobe.com/blazeds/1/blazeds_devguide/index.html
- **BlazeDS Java AMF Client, Adobe Systems Inc.**
  http://sourceforge.net/adobe/blazeds/wiki/Java%20AMF%20Client/
- **Testing Flash Applications, Stefano di Paola**
  http://www.owasp.org/images/8/8c/
  OWASPAppSec2007Milan_TestingFlashApplications.ppt
- **Adobe Flex, AMF 3 and BlazeDS: An Assessment, Jacob Karlson and Kevin Stadmeyer**
  http://www.blackhat.com/presentations/bh-usa-08/Carlson_Stadmeyer/BlackHat-Flex-Carlson _Stadmeyer_vSubmit1.pdf
- **Deblaze, Jon Rose**
  http://deblaze-tool.appspot.com/
- **Pentesting Adobe Flex Applications, Marcin Wielgoszewski**
  http://blog.gdssecurity.com/storage/presentations/
  OWASP_NYNJMetro_Pentesting_Flex.pdf

References 2/2

- **Burp Suite v1.2.14 Release Note, PortSwigger Ltd.**
  http://releases.portswigger.net/2009/08/v1214.html
- **Burp Suite Extender, PortSwigger Ltd.**
  http://portswigger.net/burp/extender/
- **AMF service test and debug utility, Pinta project members**
  http://code.google.com/p/pinta/

Pictures

- http://www.rialitycheck.com/portfolio.cfm
- http://www.silexlabs.org/amfphp/
- http://cloudfront.qualtrics.com/blog/wp-content/uploads/2010/05/thumbs-up-thumbs-down_orange.jpg
- http://livedocs.adobe.com/blazeds/1/blazeds_devguide/index.html
- http://1.bp.blogspot.com/_zMthNE3rsTA/TQjjurmc-tI/AAAAAAAAAL8/fmfG0QP6ODo/s1600/Disappointed_by_taleb83.jpg
- http://www.clker.com/clipart-pointer-finger.html