



# How to Break Web Application Security

Erwin Geirnaert  
Director European Operations  
Security Innovation  
[egeirnaert@securityinnovation.com](mailto:egeirnaert@securityinnovation.com)  
+32478289466

## OWASP

Copyright © 2004 - The OWASP Foundation  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License.

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

# Agenda

- Security Test Checklist
- Threat Modeling
- Tools
- Some examples

# Security Test Checklist

- You need an EXPERIENCED TESTER
- Create a threat model and a test plan
- Web application testing <> penetration testing
- Do not rely ONLY on automated web application security scanners
- Source code of the web application HELPS
- Have a Security Tester Toolbox
- Log everything

# Threat modeling

**“You cannot test a system until you understand the threats”**

Threat modeling is the design activity to discover the threats that your application is susceptible to.

Threat modeling yields both threats and vulnerabilities and provides ways to perform security testing in order to prioritize the security fixes needed.

# Threat modeling - Definitions

Threats are possible attacks. Vulnerabilities are security related software errors:

- A *threat* is what an attacker might try to do to an asset or through an entry point
- A *vulnerability* is a specific security exploit due to an unmitigated threat path

# Threat modeling - STRIDE

Threats can be classified using the STRIDE classification:

- **S**poofing – lying about identity
  - **T**ampering – Destroying data
  - **R**epudiation – Cleaning the steps of an attack/Denying a transaction
  - **I**nformation Disclosure – Stealing valuable private data
  - **D**enial of Service – Stopping an application from providing its basic functionality
  - **E**scalation of Privileges – Executing code with stolen high privileges
- 
- Whenever discovering threats the analyst will always think about STRIDE elements

# Threat modeling – Example of threats

## Some threats for Web Video Recoding System

- Attacker **tampers** with central video storage
- Attacker **sends** malicious input to overrun the video recording client
- Attacker **deletes** temporary recordings
- Attacker remotely **executes** code in Video web service box

# Threat modeling - DREAD

## DREAD:

- Damage potential – what's the extent of the damage if this vulnerability was to be exploited
  - Reproducibility – how well can the finder reproduce the issue
  - Exploitability – difficulty of taking advantage of the flaw for malicious purpose
  - Affected users – how many or what type of users are affected by the flaw
  - Discoverability – how fast can it be publicly be discovered
- 
- DREAD is used to analyze the risk of discovered **vulnerabilities**

# Attack vectors for web applications

- ➔ Parameter Tampering
- ➔ Cookie Tampering
- ➔ Cross-site Scripting
- ➔ SQL Injection
- ➔ Script Injection
- ➔ Command Injection
- ➔ Encoding Attacks
- ➔ Buffer Overflows
- ➔ Format-string attacks
- ➔ Harvesting User IDs
- ➔ Brute-forcing Accounts
- ➔ Path Truncation Attacks
- ➔ Hidden Path Discovery
- ➔ Application Directory and File Mapping
- ➔ Forceful Browsing
- ➔ Source Code Disclosure
- ➔ Web server vulnerability exploitation

# Security Tester Toolbox

- Tools are just a way of manipulating web applications
- They are no silver bullet, a lot of false positives can be the result of automated scan
- They can be really expensive
- They can be useful
- You need to learn how to use them and what the limitations are
- Internet Explorer can do the job and for free 😊

# Tools in the past

- 4 years ago, a limited list of free tools:
  - ▶ Achilles: local proxy
  - ▶ @Stake WebProxy: local proxy& fuzzer, in Java 😊
  - ▶ WebSleuth: plugin for IE, raw requests
  - ▶ Whisker: vulnerability scanner
  - ▶ Nikto: vulnerability scanner
  - ▶ Nessus: didn't include web vulnerabilities yet
- ▶ But they did the job, only it required more time....

# Commercial Fault Injection Test Tools

1. **SPI Dynamics WebInspect**
2. **Sanctum now Watchfire AppScan**
3. **Kavado Scando**
4. **AppSecInc AppDetective for Web Apps**
5. **Cenzic Hailstorm**
6. **Security Innovation Holodeck**
7. **NT Objectives NTOSpider**
8. **Acunetix Web Vulnerability Scanner 2**
9. **Compuware DevPartner Fault Simulator**
10. **Fortify Pen Testing Team Tool**
11. **@stake Web Proxy 2.0**
12. **Burp Intruder**
13. **Sandsprite Web Sleuth**
14. **MaxPatrol 7**
15. **Syhunt Sandcat Scanner & Miner**
16. **TrustSecurityConsulting HTTPExplorer**
17. **Ecyware BlueGreen Inspector**
18. **NGS Typhon**
19. **Parasoft WebKing (more QA-type tool)**



# Open Source or Freeware Fault Injection Test Tools

1. **WebScarab (HTTPush, Exodus)**
2. **Paros Proxy**
3. **Burp Spider**
4. **Burp Proxy**
5. **SPIKE Proxy**
6. **SPIKE**
7. **Achilles Proxy**
8. **Odysseus Proxy**
9. **Webstretch Proxy**
10. **Absinthe 1.1 (formerly SQLSqueal)**
11. **NGS SQL Injection Inference Tool (BH Europe 2005)**
12. **Internet Explorer HTMLBar Plugin**
13. **Firefox LiveHTTPHeaders and Developer Tools**
14. **Sensepost Wikto (Google cached fault-finding)**
15. **Foundstone Sitedigger (Google cached fault-finding)**

# OWASP - WebScarab

- Java based: download stand-alone JAR and runtime
- HTTP Proxy
- Client-certificates
- Session analysis
- Raw request
- Spider
- Custom plugins: BeanShell

# OWASP – WebScarab - Interceptor

The screenshot shows the 'Edit Request' window in WebScarab. At the top, there are checkboxes for 'Intercept requests' and 'Intercept responses', both of which are checked. Below this, there are two tabs: 'Parsed' (selected) and 'Raw'. The 'Parsed' view shows the following details:

Method	URL	Version
GET	http://www.owasp.org:80/	HTTP/1.0

Header	Value
Accept	image/gif, image/x-bitmap, image/jpeg, image/pjpeg, applicati...
Accept-Language	nl-be
Cookie	JSESSIONID=E2EF7E517C5EDDE3AEA180B81601BECD
User-Agent	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET ...
Host	www.owasp.org
Proxy-Connection	Keep-Alive

On the right side of the header table, there are 'Insert' and 'Delete' buttons. Below the header table is a 'Hex' section with a 'String' input field. At the bottom of the window, there are four buttons: 'Release all intercepts', 'Cancel edits', 'Accept edits', and 'Abort request'.

# OWASP – WebScarab – Raw Request

The screenshot shows the WebScarab application window. The title bar reads "WebScarab". The menu bar includes "File", "View", "Tools", and "Help". Below the menu bar is a toolbar with buttons for "Summary", "Message log", "Proxy", "Manual Request", "Spider", "SessionID Analysis", "Scripted", and "Fragments". The "Manual Request" button is selected, and the window title is "Manual Request".

Under "Manual Request", there is a "Previous Requests" section showing "1 - GET http://www.owasp.org:80/ 302 Moved Temporarily". Below this is the "Request" section, which is currently displaying the "Raw" view of the request:

```
GET http://www.owasp.org:80/index.jsp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
Accept-Language: nl-be
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: www.owasp.org
Proxy-Connection: Keep-Alive
Cookie: JSESSIONID=E2EF7E517C5EDDE3AEA180B81601BECD
```

Below the request is the "Response" section, which is currently displaying the "Raw" view of the response:

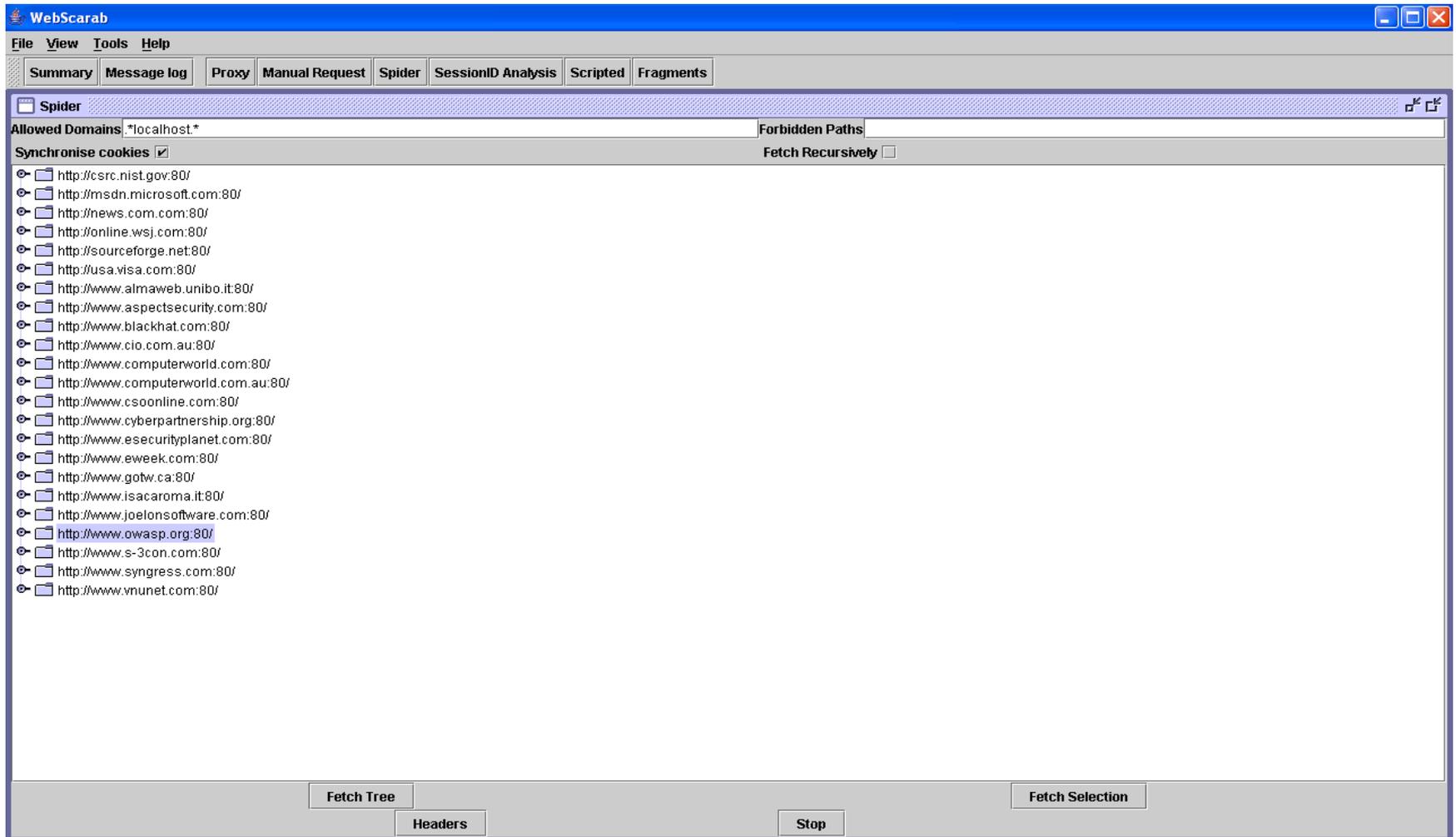
Version	Status	Message
HTTP/1.1	200	OK

Header	Value
Date	Tue, 24 May 2005 13:23:07 GMT
Server	Apache/2.0.50 (Fedora)
Content-Type	text/html; charset=ISO-8859-1
Connection	close

At the bottom of the window, there are three buttons: "Get Cookies", "Fetch Response", and "Update CookieJar".

# OWASP – WebScarab - Spider



# OWASP – WebScarab – SessionID Analysis

The screenshot shows the WebScarab application window with the 'SessionID Analysis' tab selected. The 'Collection' sub-tab is active, showing a list of previous requests. The selected request is a GET request to http://www.owasp.org:80/index.jsp with a 200 OK status. The request details are shown in the 'Request' section, and the response details are shown in the 'Response' section. The response includes a Set-Cookie header with the value 'JSESSIONID=E3CCA5AA0CDCDB674C5515C9C871D421; Path=/'.

**WebScarab**

File View Tools Help

Summary Message log Proxy Manual Request Spider SessionID Analysis Scripted Fragments

SessionID Analysis

Collection Analysis Visualisation

Previous Requests : 2 - GET http://www.owasp.org:80/index.jsp 200 OK

From message body  Name

Regex

Request

Parsed Raw

GET http://www.owasp.org:80/index.jsp HTTP/1.0  
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\*  
Accept-Language: nl-be  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)  
Host: www.owasp.org  
Proxy-Connection: Keep-Alive

Response

Parsed Raw

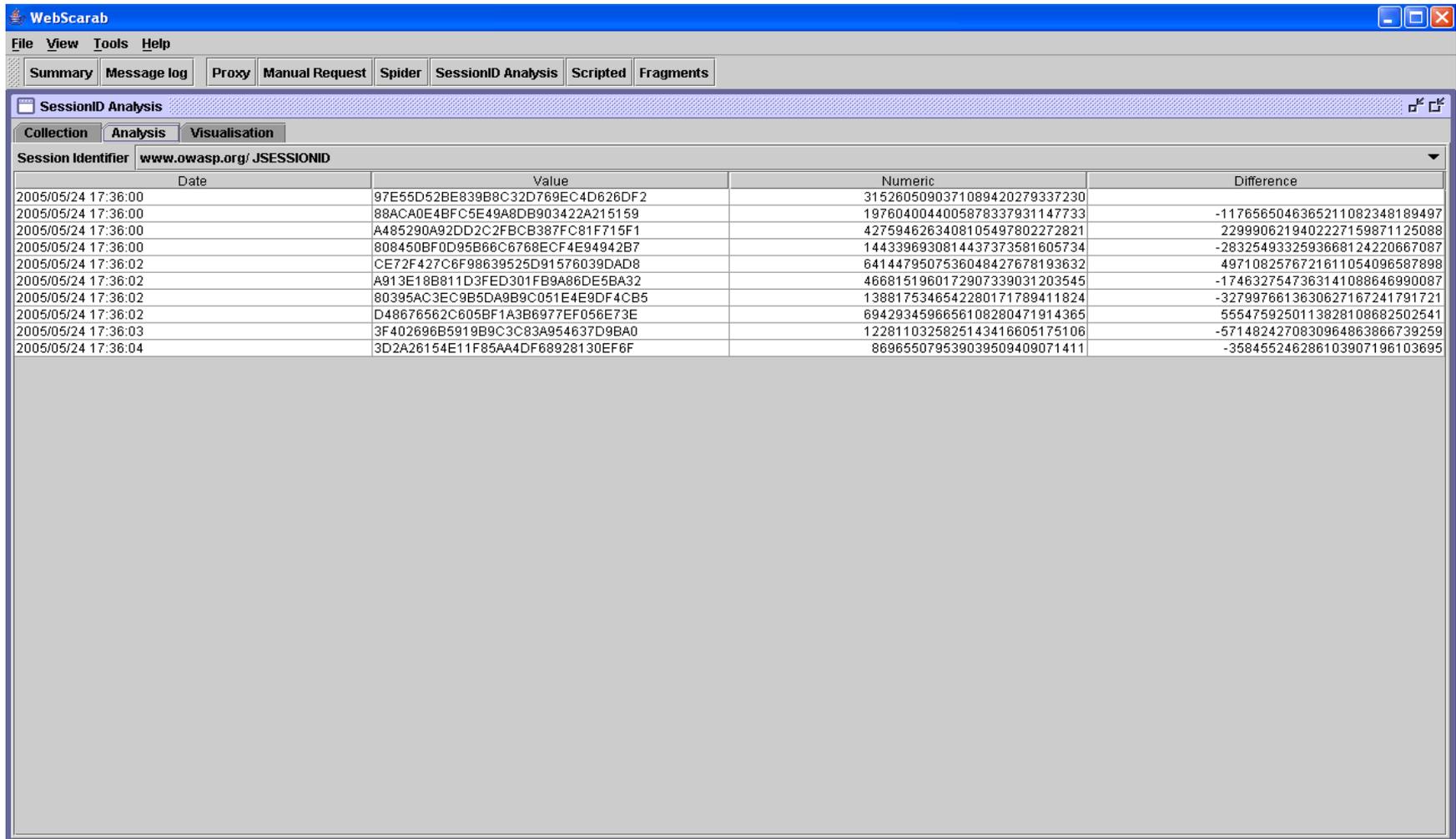
Version	Status	Message
HTTP/1.1	200	OK

Header	Value
Set-Cookie	JSESSIONID=E3CCA5AA0CDCDB674C5515C9C871D421; Path=/
Content-Type	text/html; charset=ISO-8859-1
Connection	close

HTML XML Text Hex

Test Samples 10 Fetch

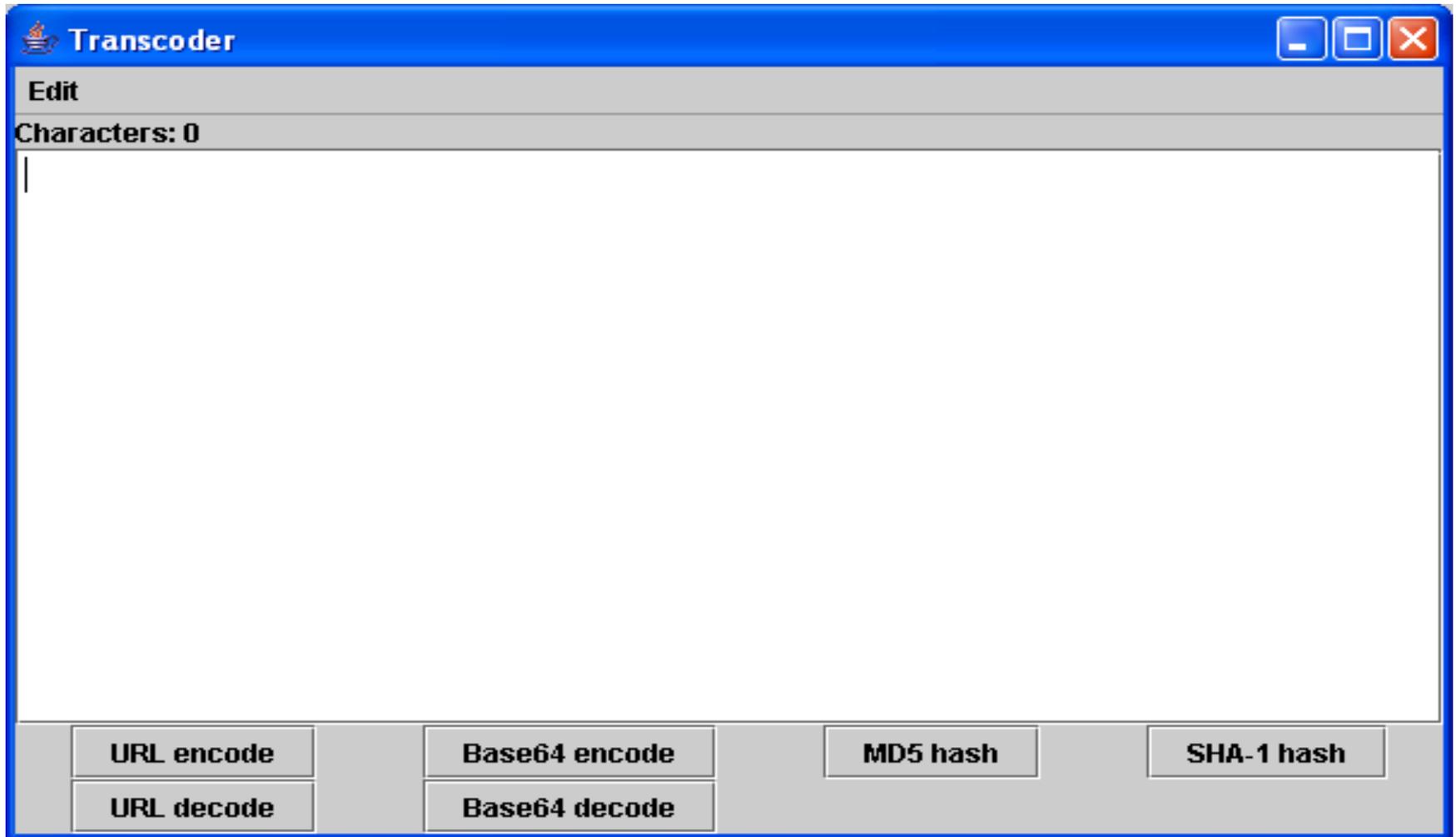
# OWASP – WebScarab – SessionID Analysis



The screenshot shows the WebScarab application window with the 'SessionID Analysis' tab selected. The window title is 'WebScarab' and the menu bar includes 'File', 'View', 'Tools', and 'Help'. The toolbar contains buttons for 'Summary', 'Message log', 'Proxy', 'Manual Request', 'Spider', 'SessionID Analysis', 'Scripted', and 'Fragments'. The 'SessionID Analysis' window has three tabs: 'Collection', 'Analysis', and 'Visualisation'. The 'Collection' tab is active, showing a table with the following data:

Session Identifier	www.owasp.org/ JSESSIONID		
Date	Value	Numeric	Difference
2005/05/24 17:36:00	97E55D52BE839B8C32D769EC4D626DF2	3152605090371089420279337230	
2005/05/24 17:36:00	88ACA0E4BFC5E49A8DB903422A215159	1976040044005878337931147733	-1176565046365211082348189497
2005/05/24 17:36:00	A485290A92DD2C2FBCB387FC81F715F1	4275946263408105497802272821	2299906219402227159871125088
2005/05/24 17:36:00	808450BF0D95B66C6768ECF4E94942B7	1443396930814437373581605734	-2832549332593668124220667087
2005/05/24 17:36:02	CE72F427C6F98639525D91576039DAD8	6414479507536048427678193632	4971082576721611054096587898
2005/05/24 17:36:02	A913E18B811D3FED301FB9A86DE5BA32	4668151960172907339031203545	-1746327547363141088646990087
2005/05/24 17:36:02	80395AC3EC9B5DA9B9C051E4E9DF4CB5	1388175346542280171789411824	-3279976613630627167241791721
2005/05/24 17:36:02	D48676562C605BF1A3B6977EF056E73E	6942934596656108280471914365	5554759250113828108682502541
2005/05/24 17:36:03	3F402696B5919B9C3C83A954637D9BA0	1228110325825143416605175106	-5714824270830964863866739259
2005/05/24 17:36:04	3D2A26154E11F85AA4DF68928130EF6F	869655079539039509409071411	-358455246286103907196103695

# OWASP – WebScarab – Transcoder



# Some examples

- Parameter tampering
- Cross-site-scripting
- Hidden fields
- SQL Injection
- Error messages
- Google 😊

# That's it...

- Any Questions?

## Thank you!