# Fuzzing

**Piotr Łaskawiec**
**J2EE Developer/Pentester**

**Metrosoft (www.metrosoft.com)**
piotr.laskawiec@gmail.com

**OWASP**
14.01.2010



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

# Abstract

- Methods of testing the application
- Ensuring application security
- Fuzzing – definition
- Use of fuzzing
- Types of fuzzers
- Fuzzing and SDLC
- Who uses fuzzers?
- Examples of fuzzers
- Web application fuzzing
- Summary

# Application testing

- Popular tests:
  - Unit testing
  - Functional testing
  - Regression testing
  - Performance testing
  - Usability tests
- Other classification:
  - Whitebox, Graybox, Blackbox
- What about security?
  - Security on level of design, implementation, testing and deployment.

# Ensuring application security

- Response vs prevention
- Response:
  - Firewall
  - IDS/IPS
  - Antivirus
  - Authentication mechanisms
  - Vulnerability scanners (Nessus, Nikto, etc.)
  - Etc.
- Prevention:
  - Fuzzing!
  - Code audit/RE

# Fuzzing - definition

- Fuzzing is a method of testing software to find security holes and unexpected behavior of an application, using semirandom data.
- Fuzzing most frequently is fully automated process - „run and wait for result”.

# Fuzzing – what does it mean in practice?

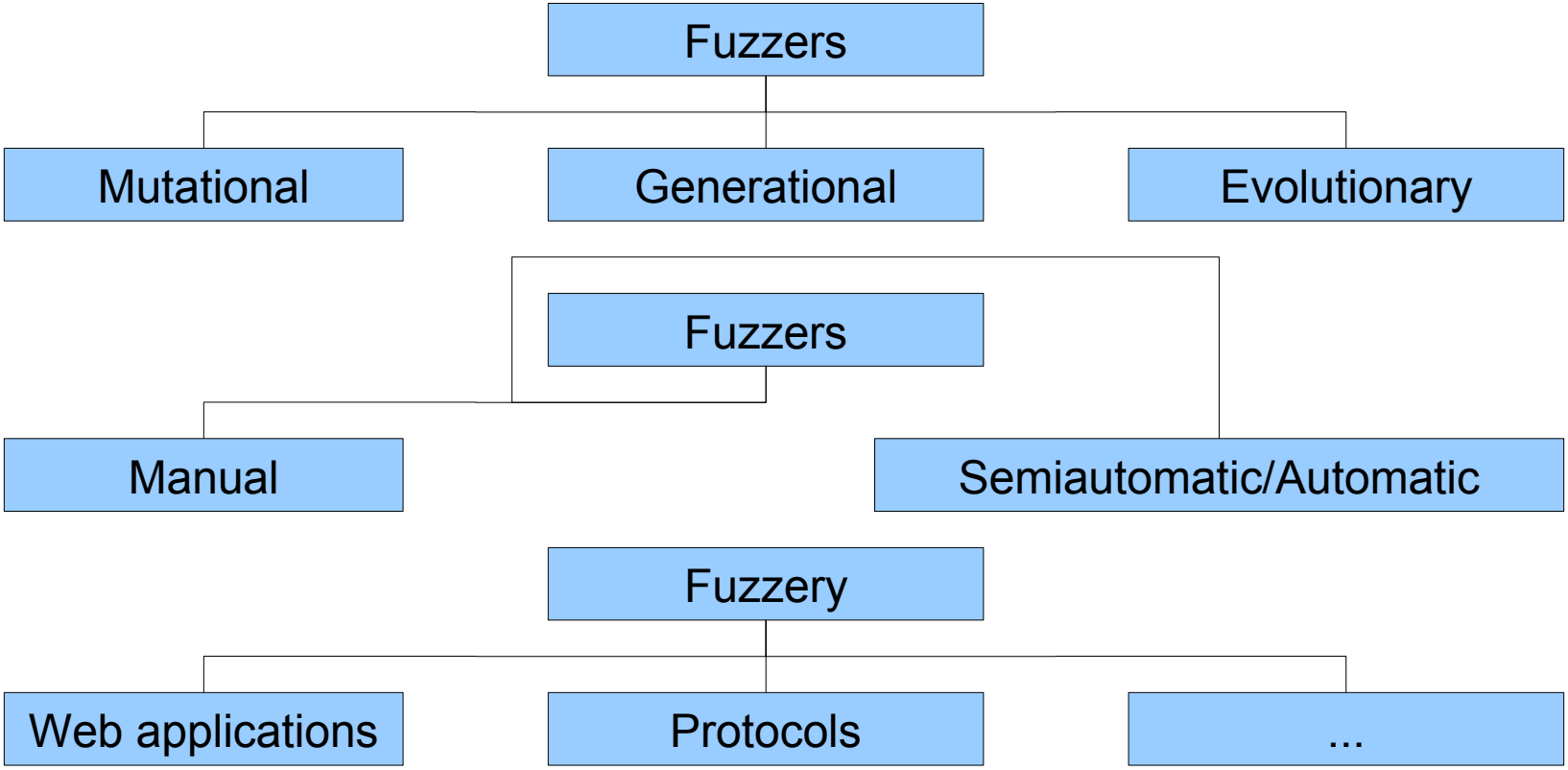
- Fuzzing == Negative testing
- The aim of the fuzzer is to send invalid data to application (too long strings, improper encoding, bad file format, bad sequence of communicates).
- We hope that the application will accept the data and an atypical reaction of the program will occur – DoS, error message, fast-growing demand for resources.
- Our target is to break the application!

# Using fuzzers

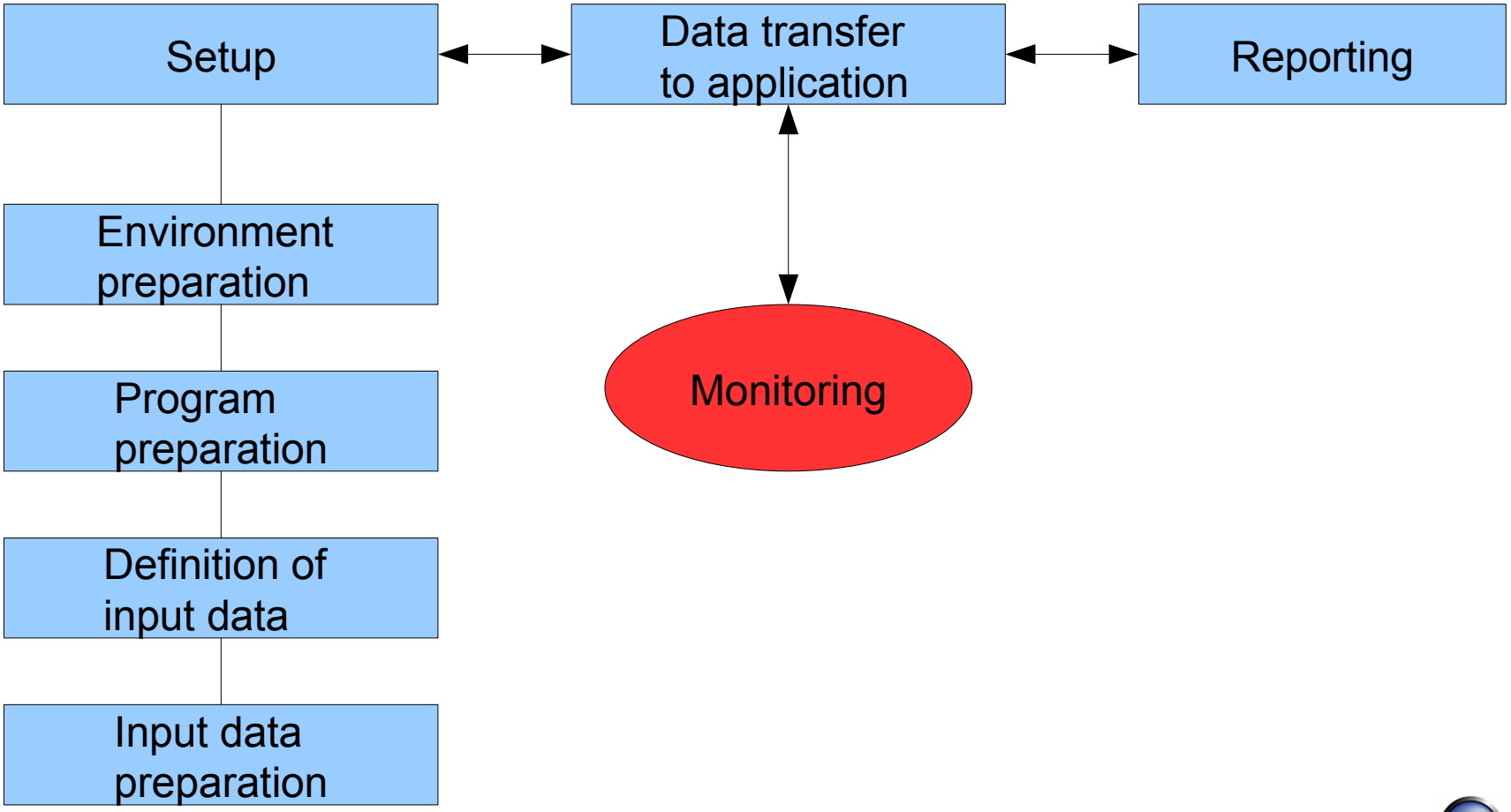
- Local applications
- Web applications
- WebServices
- Network applications
- ActiveX controls
- Files
- Libraries
- …

# Fuzzer's classification

- There are many criteria for classification
- Examples:



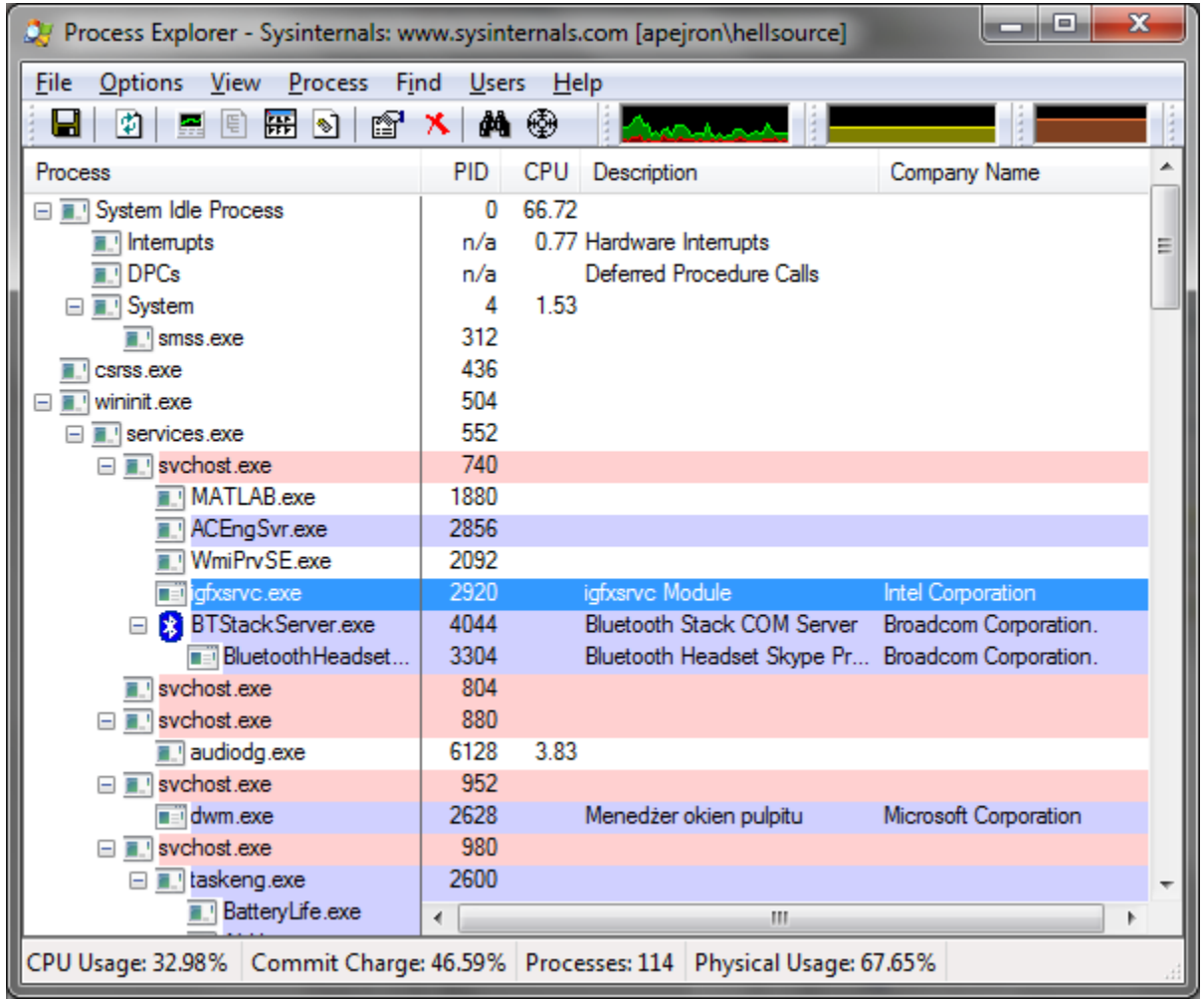
# Fuzzing process



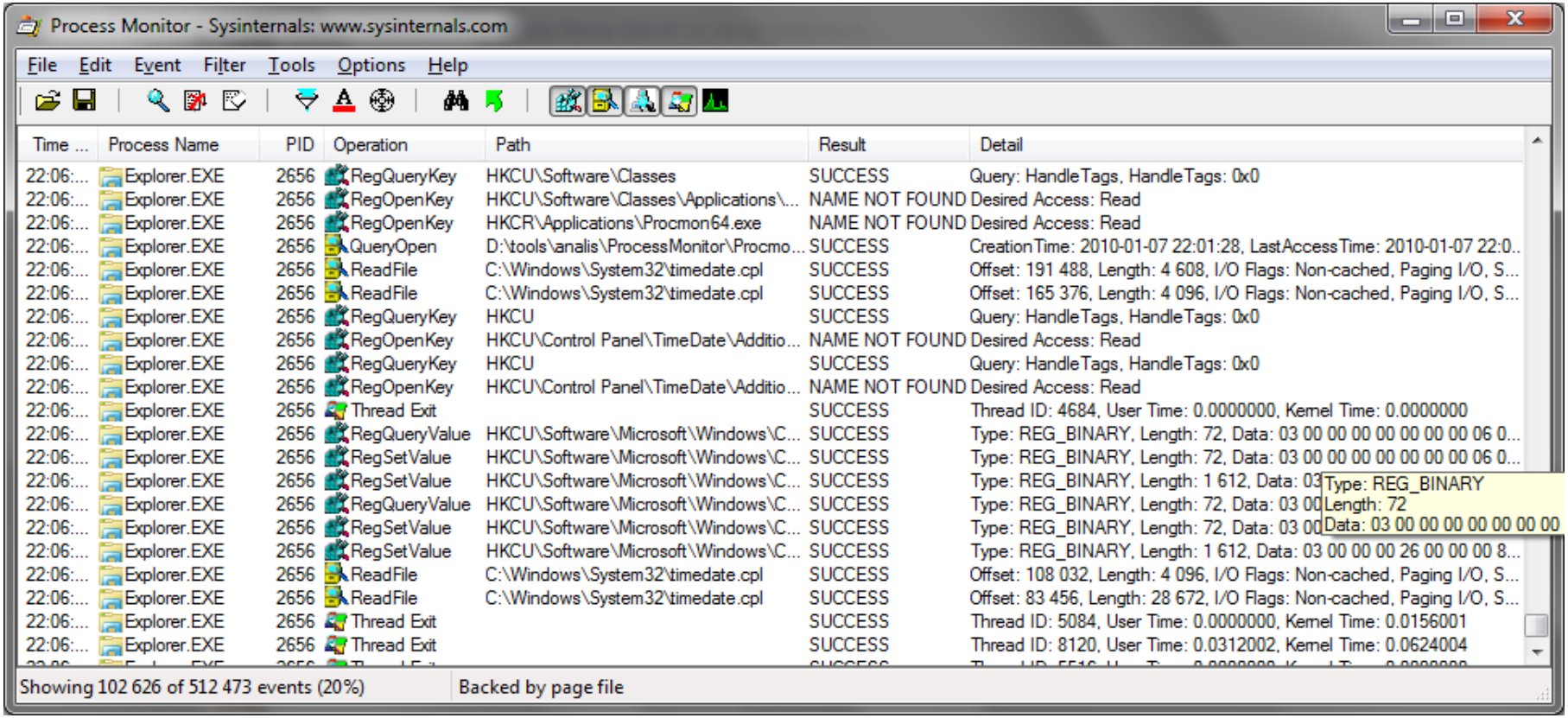
# Monitoring

- Observation of program behavior
- Logs
- Debuggers (!exploitable...)
- Files, processes and network monitors
- Virtualization (VMWare)
- Source code modifications (breakpoints)
- Additional techniques (Valgrind, Guard Malloc)
- Combined techniques

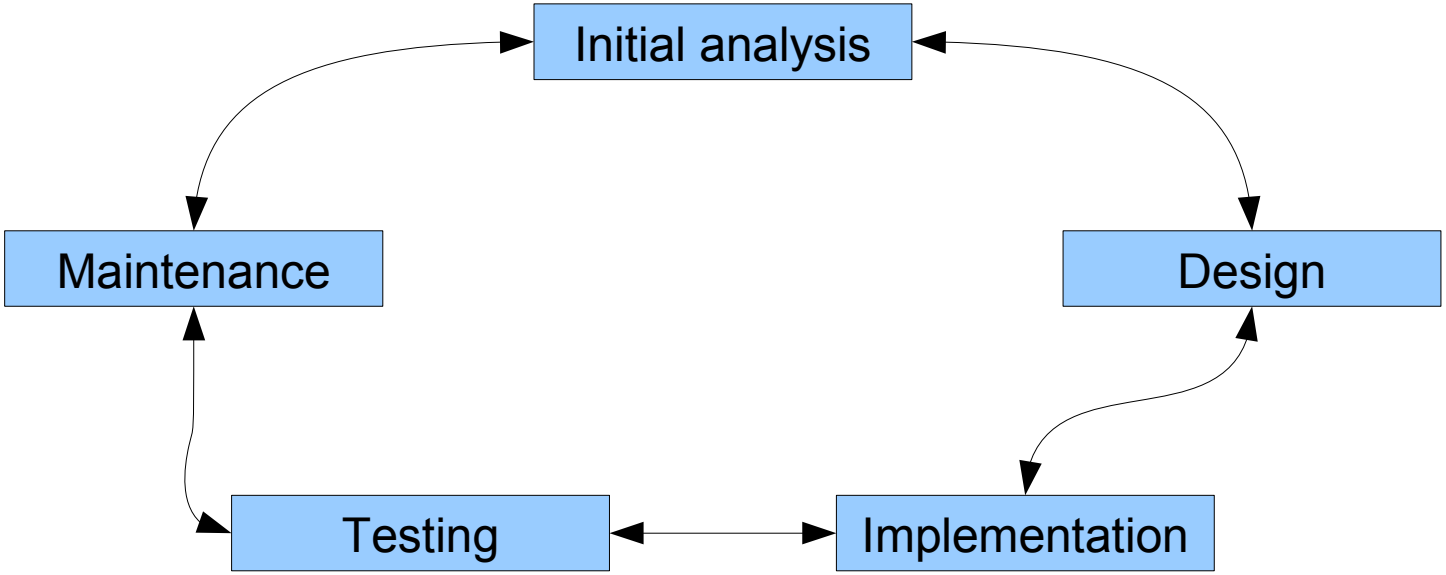
# Process Explorer



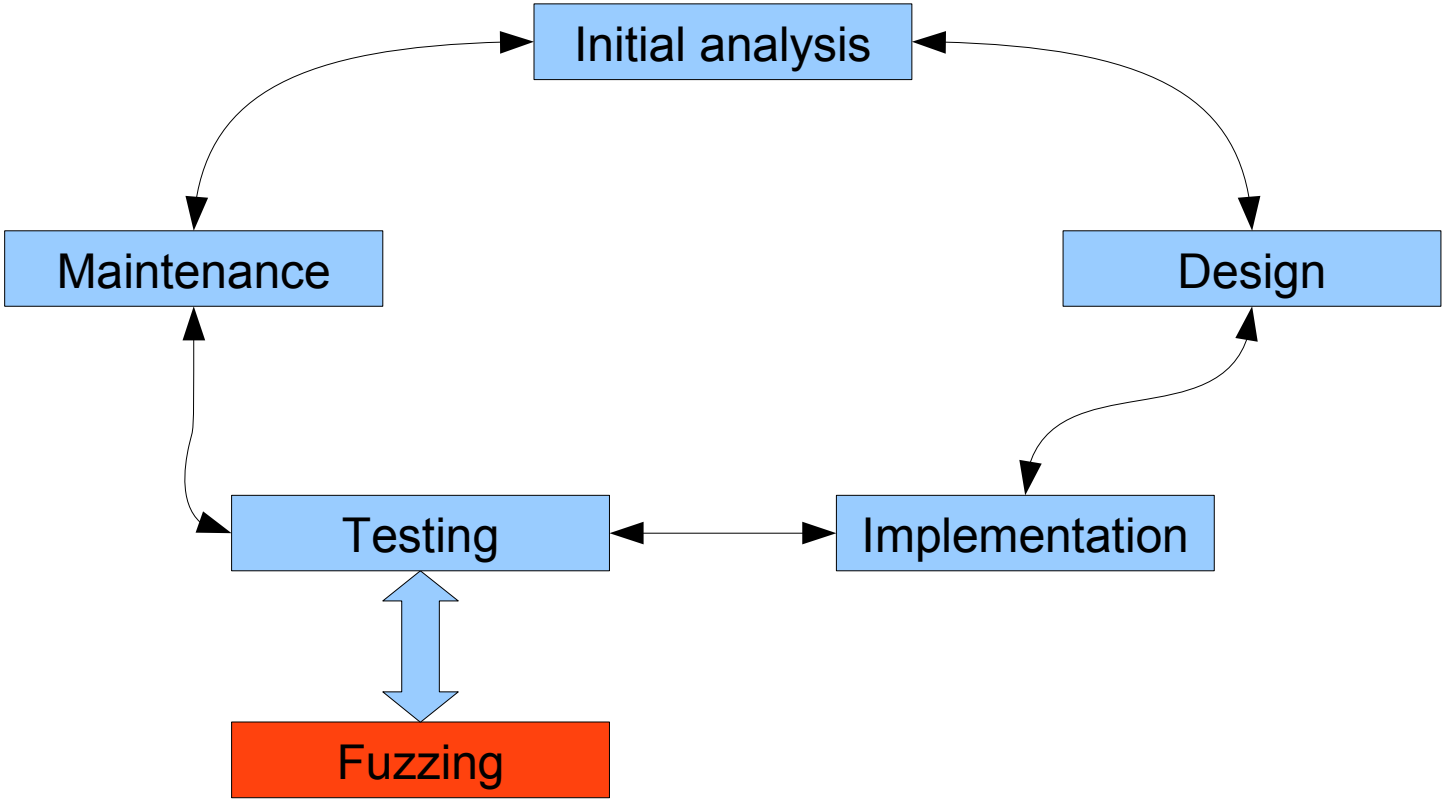
# Process Monitor



# Fuzzing and SDLC



# Fuzzing and SDLC



# Fuzzing and SDLC

- After publication of the new version, application is tested by a previously prepared fuzzer.
- Test results are verified by testers. Next, they are sent to programmers.
- If any errors occur, programmers must fix the application.
- New build once again must pass the fuzzing process.

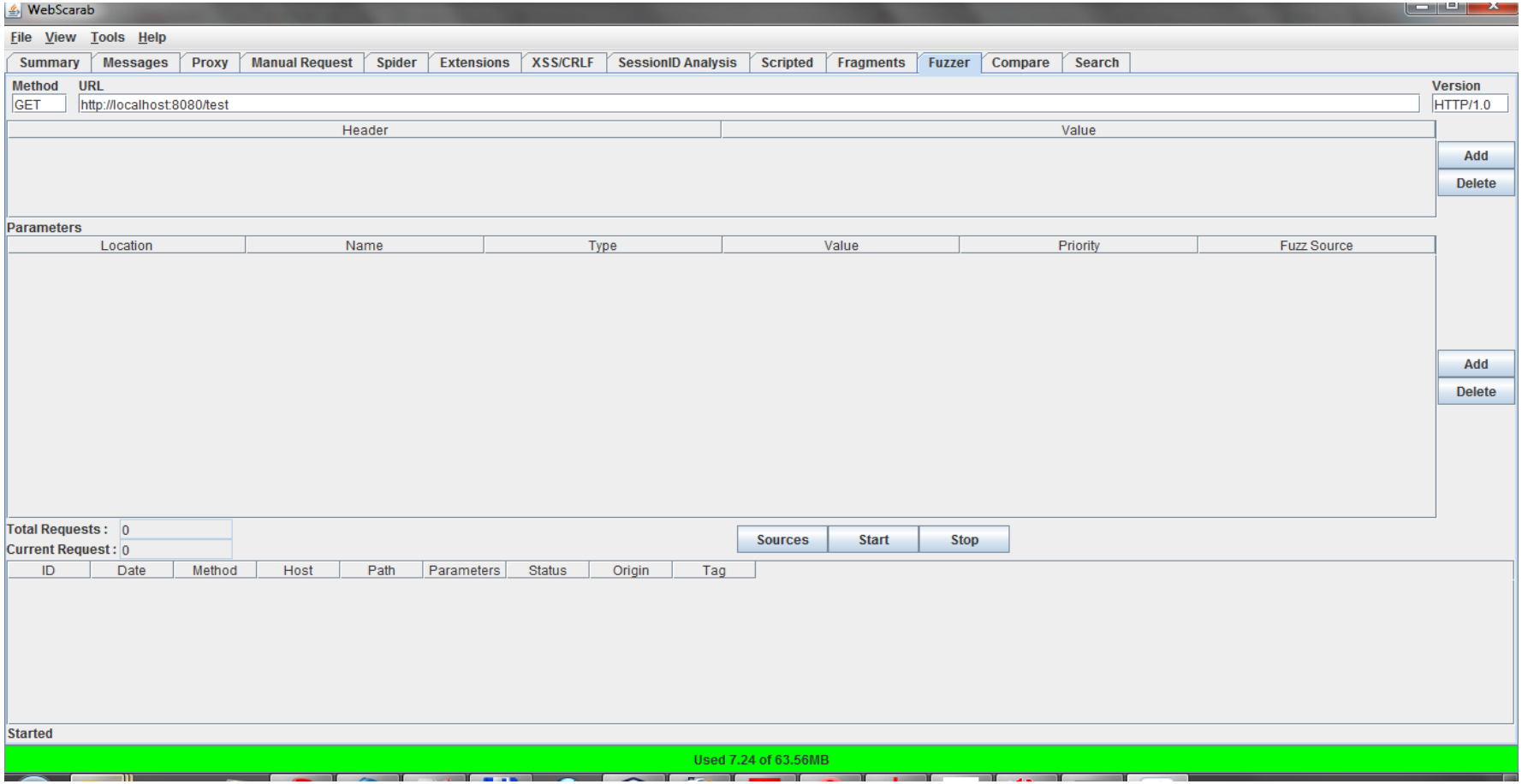
# Examples:

- Adobe
  - http://blogs.adobe.com/asset/2009/12/fuzzing_reader_-_lessons_learned.html
- Bugs in IIS
  - http://pentestit.com/2009/12/28/microsoft-iis-day-open/
- Mozilla JavaScript fuzzer
  - http://blog.mozilla.com/security/2007/08/02/javascript-fuzzer-available/
- Microsoft MiniFuzz
  - http://www.microsoft.com/downloads/details.aspx?FamilyID=b2307ca4-638f-4641-9946-dc0a5abe8513&displaylang=en
- Google Flayer
  - http://code.google.com/p/flayer/

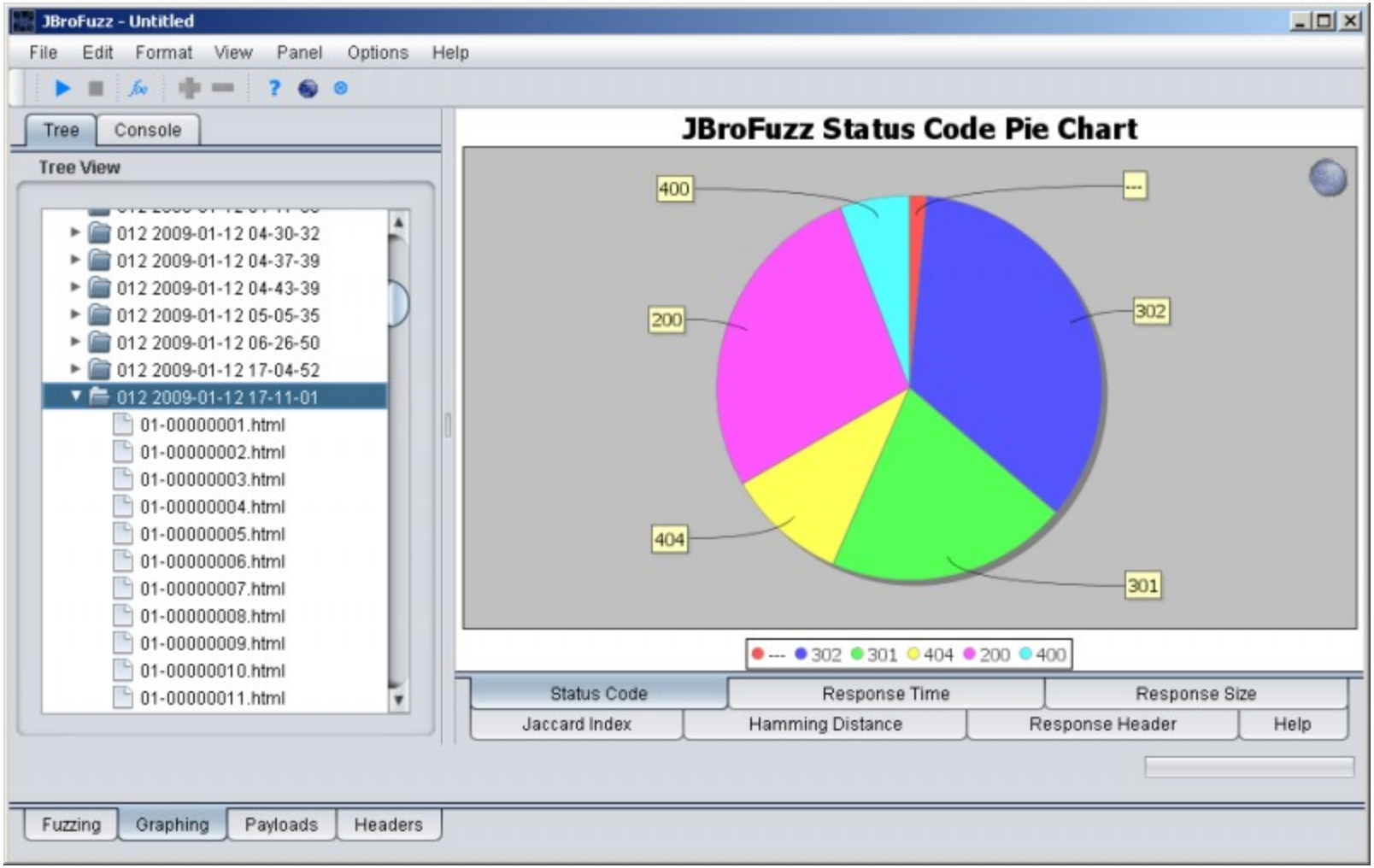
# Examples of fuzzers:

- Frameworks:
  - Peach (http://peachfuzzer.com/)
  - Sulley
- Specialized fuzzers:
  - JBroFuzz (OWASP)
  - WSFuzzer (OWASP)
  - TAOF
  - Wfuzz
  - Spike Proxy
  - WebFuzz
- Custom solutions

# WebScarab Fuzzer plugin



# JBroFuzz


JBroFuzz - Untitled
File Edit Format View Panel Options Help
Tree Console
Tree View
012 2009-01-12 04-30-32
012 2009-01-12 04-37-39
012 2009-01-12 04-43-39
012 2009-01-12 05-05-35
012 2009-01-12 06-26-50
012 2009-01-12 17-04-52
012 2009-01-12 17-11-01
01-00000001.html
01-00000002.html
01-00000003.html
01-00000004.html
01-00000005.html
01-00000006.html
01-00000007.html
01-00000008.html
01-00000009.html
01-00000010.html
01-00000011.html
JBroFuzz Status Code Pie Chart
400
---
200
302
404
301
--- 302 301 404 200 400
Status Code
Response Time
Response Size
Jaccard Index
Hamming Distance
Response Header
Help
Fuzzing
Graphing
Payloads
Headers

# Web application fuzzing

- Problems:
  - Identification of input points
    - HTTP communication analysis
    - Webspidering
    - Search engines
  - Generation of test data
    - Payloads hardcoded in fuzzers
    - Bruteforce
    - Payloads based on patterns
  - Error identification

# Error identification

- HTTP response codes
- Analysis of website content
- Comparision of website internal structure
- Time attacks
- Multiple requests
- Analysis of unique data identifying website.
- Logs

# Anti-fuzzing

- We can't directly defend against fuzzing!
- Generic defense:
  - Validation of input data
  - Application of good programming practices
  - Ensuring security through all phases of SDLC

# Summary

# Fuzzing advantages

- Full automatization (in most cases)
- Fuzzers find real vulnerabilities
- Ability to identify bugs which are hard to find by manual testing
- Ability to quickly obtain satisfactory results (first bug)

# Fuzzing disadvantages

- Inability to find logical bugs
- Inability to find complex bugs
- Time required for performing test is very hard to specify

# Additional information

- Talks:
  - PyCON 2008
  - SEConference 2009
- Sites:
  - fuzzing.eu
  - fuzzing.org
  - krakowlabs.com/lof.html


**09-10.04.2010**

www.seconference.pl

# Questions

# Thanks for your attention!