XSSer: "The mosquito"

THSF- 05/2011

/ Cross Site "Scripter" /

- *Current version: XSSer.v1.5: "Swarm Edition"*

- *Main Website: http://xsser.sf.net*

Where XSSer come from?

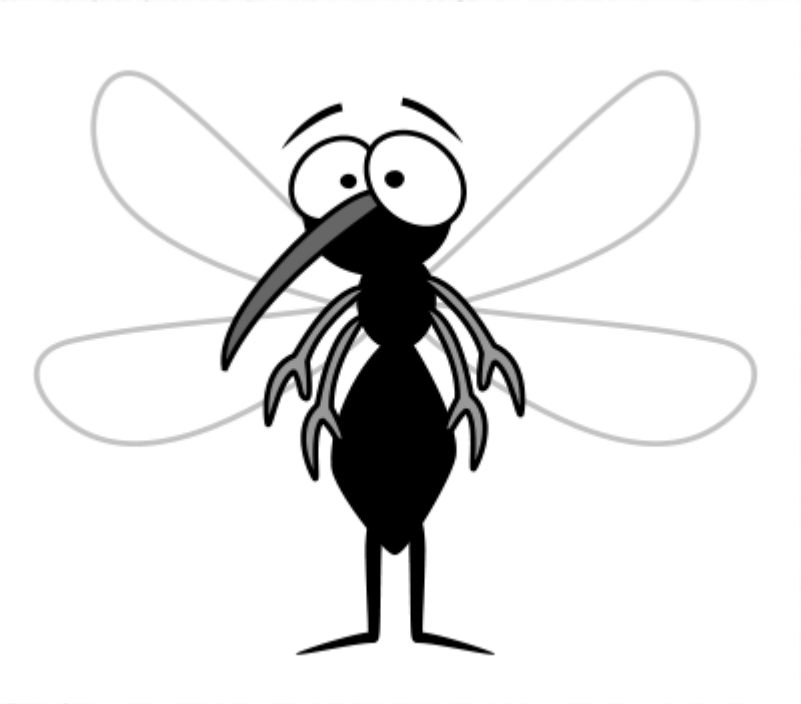
How it works?

Features & Technicques

"Flying mosquito/s"

Next steps ...

XSSer Community!



/ Where XSSer come from? /

- Wikipedia Definition: Cross Site Scripting (XSS)

- Top 10 Application Security Risks: OWASP 2010

Cross Site "Scripter" (aka XSSer) is an automatic -framework- to:

- Detect XSS flaws in web-based aplications.
- Exploit -local/remote- code “on wild”.
- Report found vulnerabilities in *real time* to community.

OWASP: Open Web Application Security Project:

- XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping.
- XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

Top 10 2010-A2-Cross-Site Scripting (XSS)

/ Where XSSer come from? /

- PDF: "XSS for fun and profit": EN / SP

- Website - NLNet Foundation

Paper – SCG'09 : "*XSS for fun and profit*"

Complete PDF (174 pages) about XSS with detailed examples:

- Attack Types
- Evading filters
- PoC Examples
- Attack technicques
- XSS Cheats / Fuzz vectors
- Tools / Links / Bibliography
- [...]

NLNet Awards: Next Deadline 01/06/2011

NLNet Awards: "XSSer Winner April 2010"

- Start: 01/07/2010 – End: 01/02/2011
- It was 5 milestones (steps)

- XSSer project recieve: 3.000 €

/ How it works? /

- XSSer "official" pre-compiled packages:

Debian / Ubuntu
ArchLinux

XSSer is a *pentesting tool* written in *python* under the *GNU License v.3*

+xsser-public (v1.5: "Swarm Edition")

+docs (documentation)
+Makefile (to create packages)
+gtk
+xsser (executable)
+XSSer:

- crawler.py – Crawler connected to XSSer
- curlcontrol.py – Curl class to manage connections
- encdec.py – Encoder/Decoder injections to bypass filters
- main.py – Main core code (kernel)
- options.py – User interface options
- imagexss.py – ImageXSS auto-builder
- publish.py – Social networking publisher connected to XSSer
- gtkcontroller.py – GTK interface main controller
- globalmap.py – Globalgeomap plotting system
- theadpool.py – Threads manager system for working proccess
- twsupport.py – Twisted technology handler
- mozchecker.py – Mozilla embed checker
- dork.py – Dorker connected to XSSer
- tokenhub.py – Final token checker requester
- reporter.py – Central reporting point
- heuristic.py – Heuristic checkers

+post:
- shorter.py – Shorteners conneted to XSSer
- xml_exporter.py – AML/XML exporters

+fuzzing:
- DCP.py - Data Control Protocol fuzzing plugin
- DOM.py – Document Object Model fuzzing pugin
- Vectors.py – XSS vectors and browsers
- HTTPsr.py – HTTP Splitting vectors

Latest development version from the -subversion- repository:

```
$ svn co https://xsser.svn.sourceforge.net/svnroot/xsser xsser
```

/ How it works? /

- *Videos of "old" versions of XSSer:*

Simple automatic payloading vectors
Server side Apache logging
Dorking injections

From Shell:

$ python xsser [OPTIONS] [-u |-i |-d] [-g |-p |-c] [Request(s)] [Vector(s)]
[Bypasser(s)] [Technique(s)] [Final Injection(s)]

- List available commands: $ python xsser -h / --help ()

** Simple injection from URL:*

$ python xsser.py -u "http://host.com"

** Multiple injections from URL, with automatic payloading, using tor proxy, injecting on payloads character encoding in "Hexadecimal", with verbose output and saving results to file (XSSlist.dat):*

$ python xsser.py -u "http://host.com" --proxy "http://127.0.0.1:8118" --auto --Hex --verbose -w

- More examples: http://xsser.sourceforge.net/#examples

/ How it works? /

- Video XSSer:"Swarm Edition!"

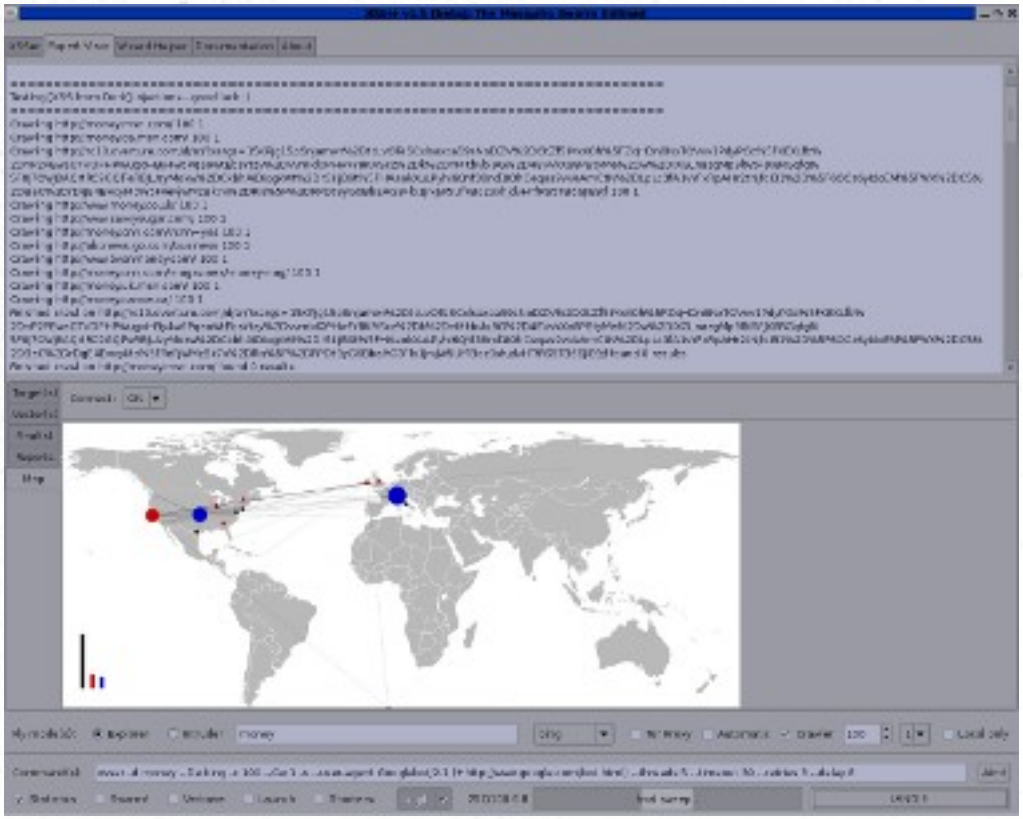
GTK version: http://xsser.blip.tv

From Interface:

$ python xsser --gtk (--silent)

_* Intuitive navigation
_* Command(s) autocompleter
_* Wizard helper
_* Expert visor
_* Target(s) geolocation
_* Documentation
_* […]

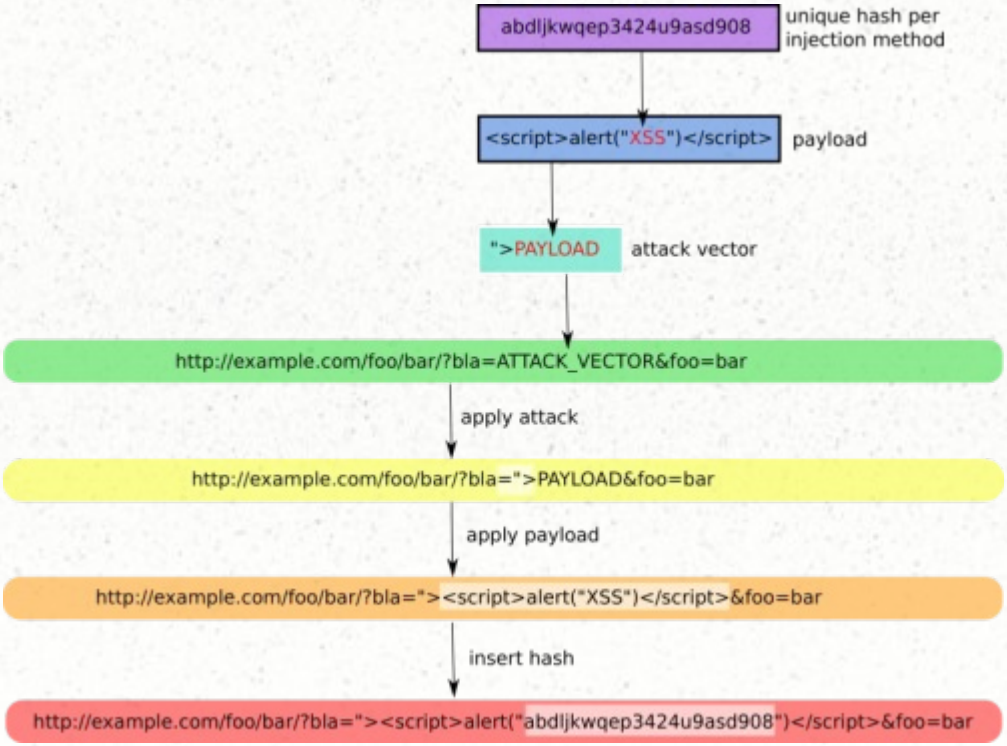
"Fly your mosquito(s) faster..."



/ How it works? /

$ man xsser

XSSer URL Generation Schema:



"With every injection with a different -hash-, is more difficult to block the tool."

/ Features and Technicques /

- Definitions:

GET / POST
WebCrawler

Main Features (v1.5: “Swarm Edition!”:

Options:

* -s	Show advanced statistics
* -v	Verbose mode

Special features:

* --imx=IMX	Create a false image with code XSS embedded
* --fla=FLASH	Create a false .swf with code XSS embedded

Select target(s):

* -u URL, --url=URL	Enter target(s) to audit
* -i READFILE	Read target URLs from a file
* -d DORK	Process search engine dork results as target urls
* --De=DORK_ENGINE	Search engine to use for dorking (dork.py)

Select type of HTTP/HTTPS Connection(s):

* -g GETDATA	Enter payload to audit using GET. (ex: '/menu.php?q=')
* -p POSTDATA	Enter payload to audit using POST. (ex: 'foo=1&bar=')
* -c CRAWLING	Number of urls to crawl on target(s): 1-99999
* --Cw=CRAWLING_WIDTH	Deeping level of crawler: 1-5
* --Cl	Crawl only local target(s) urls (default TRUE)

/ Features and Technicques /

- *XSSer automatic injections (--auto):*

List of "exploitable": reported vectors

Configure Request(s):

* --cookie=COOKIE	Change your HTTP Cookie header
* --user-agent=AGENT	Change your HTTP User-Agent header (default SPOOFED)
* --referer=REFERER	Use another HTTP Referer header (default NONE)
* --headers=HEADERS	Extra HTTP headers newline separated
* --auth-type=ATYPE	HTTP Authentication type (value Basic or Digest)
* --auth-cred=ACRED	HTTP Authentication credentials (value name:password)
* --proxy=PROXY	Use proxy server (tor: http://localhost:8118)
* --timeout=TIMEOUT	Select your Timeout (default 30)
* --delay=DELAY	Delay in seconds between each HTTP request (default 8)
* --threads=THREADS	Maximum number of concurrent HTTP requests (default 5)
* --retries=RETRIES	Retries when the connection timeouts (default 3)

Select Vector(s):

* --payload=SCRIPT	OWN - Insert your XSS construction -manually-
* --auto	AUTO - Insert XSSer 'reported' vectors from file

Select Bypasser(s):

* --Str	Use method String.FromCharCode()	* --Hes	Use Hexadecimal encoding, with semicolons
* --Une	Use function Unescape()	* --Dwo	Encode vectors IP addresses in DWORD
* --Mix	Mix String.FromCharCode() and Unescape()	* --Doo	Encode vectors IP addresses in Octal
* --Dec	Use Decimal encoding	* --Cem	Try -manually- different Character Encoding (ex:'Mix,Une,Str,Hex')
* --Hex	Use Hexadecimal encoding		

/ Features and Technicques /

- More info about all the technicques:

PDF: "XSS for fun and profit": EN / SP

Special Technique(s):

* --Coo	Cross Site Scripting Cookie injection
* --Xsa	Cross Site Agent Scripting
* --Xsr	Cross Site Referer Scripting
* --Dcp	Data Control Protocol injections
* --Dom	Use Anchor Stealth (DOM shadows!)
* --Ind	HTTP Response Splitting Induced code
* --Anchor	Use Anchor Stealth payloader (DOM shadows!)

Select Final injection(s):

* --Fp=FINALPAYLOAD	Insert your final code to inject -manually-
* --Fr=FINALREMOTE	Insert your final code to inject -remotelly-
* --Doss	XSS Denial of service (server) injection
* --Dos	XSS Denial of service (client) injection
* --B64	Base64 code encoding in META tag (rfc2397)

Special Final injection(s):

* --Onm	Use onMouseMove() event to inject code
* --Ifr	Use "iframe" source tag to inject code

Special Final injection(s):

* --publish	Output 'positives' to Social Networks (identi.ca)
* --short=SHORTURLS	Display -final code- shortered (tinyurl, is.gd)

/ " Flying the mosquito " /

ZzzzzzzzzzZZZZZZz...

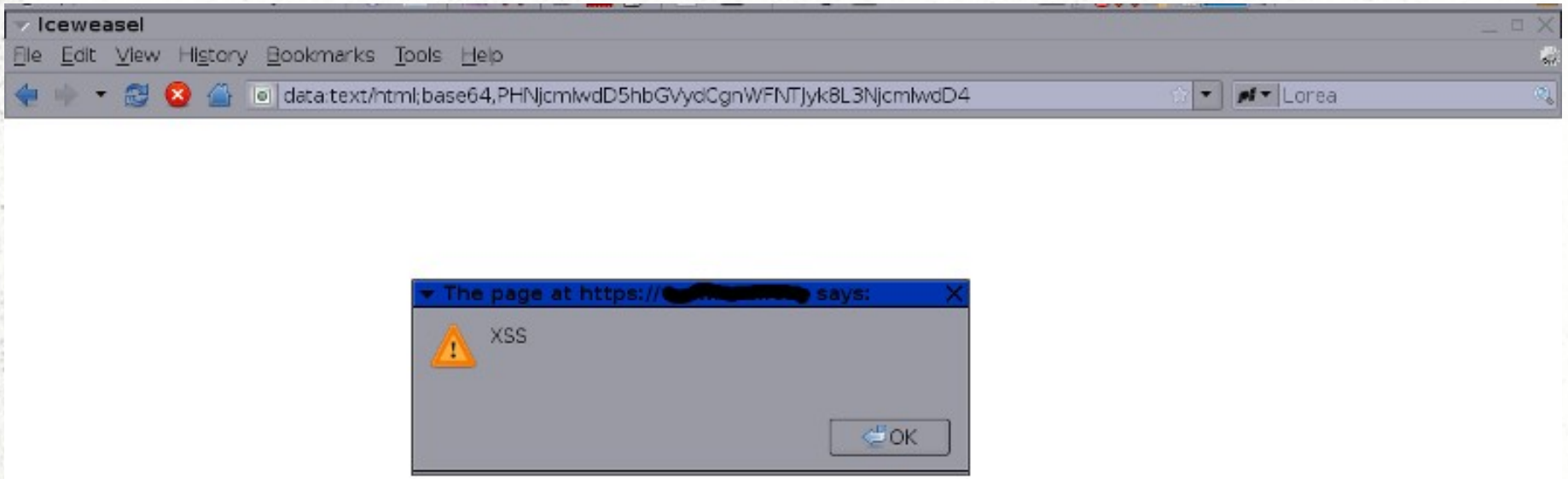
Comparison of 43 web application vulnerability scanners:

"The best HTTP GET XSS detection ratio (while considering the low amount of false positives) of open source tools belongs to XSSer."

Security Tools Benchmarking (25-01-2011).

/ “ Flying the mosquito” /

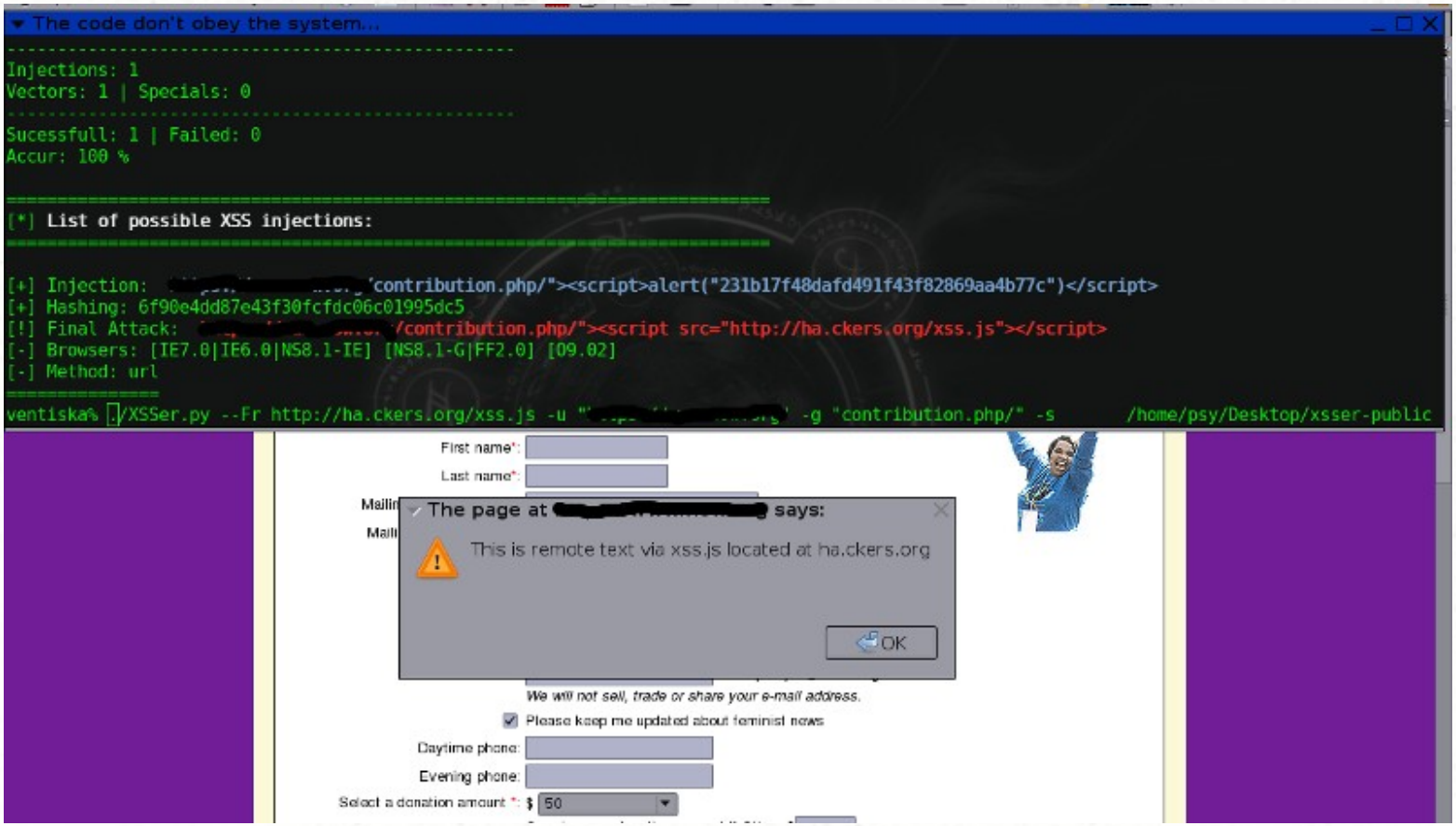
ZzzzzzzzzZZZZZZz...



Example of DCP Injection

/ “ Flying the mosquito” /

ZzzzzzzzzZZZZZZz...



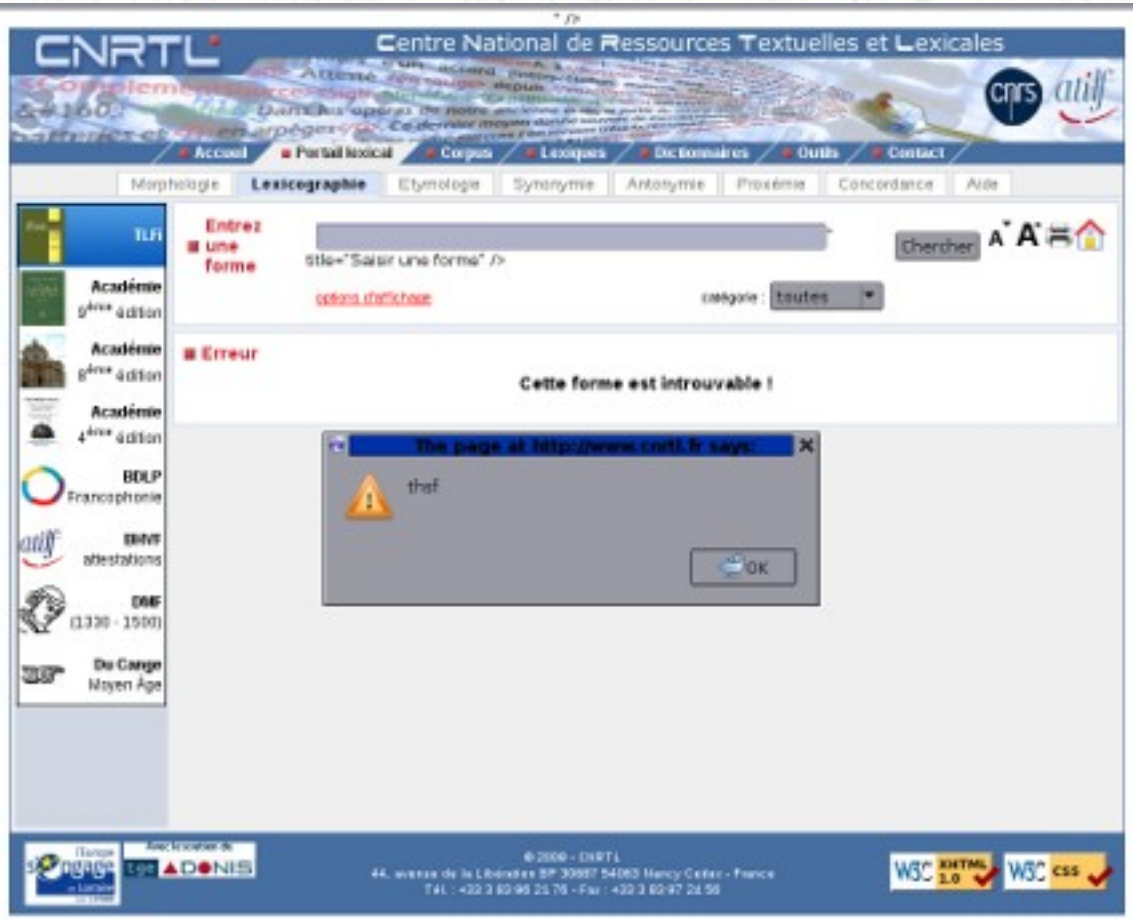
Example “remote” code injection

/ “ Flying the mosquito” /

ZzzzzzzzzZZZZZZz...

/ “ Flying the mosquito” /

ZzzzzzzzzZZZZZZz...



/ “Flying the mosquito”/

Ouch!!!!...

HBGary Email Viewer

aaron@hbgary.com

Tweet

Original file:	1297008273.M48490P14101Q5582.cybercom
click here to show this e-mail with HTML markup	
From:	Ted Vera <ted@hbgary.com>
To:	mark@hbgary.com, Barr Aaron <aaron@hbgary.com>
Date:	Sat, 21 Aug 2010 00:16:51 -0600
Subject:	Oracle Exploit Tools
click here to show full headers	
Attachments:	This e-mail does not have any attachments.

```
http://www.youtube.com/watch?v=euujmKDxmC4


Another Tool:

Just updated today, for web-apps.

http://xsser.sourceforge.net/
http://xsser.svn.sourceforge.net/svnroot/xsser/

--
Ted Vera  |  President  |  HBGary Federal
Office 916-459-4727x118  | Mobile 719-237-8623
www.hbgary.com  |  ted@hbgary.com
```

/ Next steps /

“All kind of help is welcome!!” ;-)

Active Tasks:

* Testing
* Documentation
* Bugfixing
* Refactoring
* Community growth
* Diffusion
* [...]

Under research (26/05/2011):

* Increase accurance of 'browser checked' payloads
* XSS Shell + XSS Tunneling
* XSS Worms auto-payloaders
* XSSer P2P engine
* [...]

Join the workgroup!!! : https://n-1.cc/pg/groups/15466/xsser/

/ Community /

irc.freenode.net / channel: #xsser

If you are interesting in follow last news about XSSer, you can join #xsser-community on many different places:

* Mailing list:

  https://lists.sourceforge.net/lists/listinfo/xsser-users

* Microblogging (tag: #xsser):

  http://identi.ca/group/xsser
  http://identi.ca/psy

* Social networking (XSSer Workgroup):

  https://n-1.cc/pg/groups/15466/xsser/

/ ZZZzzzzzzzzzzz..... /

http://lordepsylon.net
epsylon@riseup.net
0xB18E792B

#spanishrevolution -> #worldrevolution ;-)