Màster interuniversitari en Seguretat de les TIC – MISTIC

ANÀLISI D'ERRORS EN SERVIDORS WEB

Estudiant: Genís Domínguez

Directors: Jordi Duch i Agustí Solanas

Universitat Oberta de Catalunya

Gener 2014

Resum del Treball

Aquest treball es divideix en quatre capítols. El primer capítol és una introducció general a l'obtenció d'informació sensible a partir dels errors que podem trobar en una aplicació web en cadascuna de les seves capes: servidor, aplicació i base de dades. S'explica en què consisteix OWASP, com es pot obtenir informació a partir dels errors, com buscar aquesta informació amb Google i com un atacant podria aprofitar-la. En aquest capítol s'utilitza un cas real amb el Banc Sabadell d'exemple. El segon capítol tracta específicament dels tipus d'errors que podem obtenir d'un servidor HTTP, llistant aquells que podem provocar, com fer-ho i quina informació proporcionen: *time-outs*, mètodes no vàlids, peticions mal formades,.... El tercer capítol conté un estudi detallat del error HTTP 404 – pàgina no trobada, el seu comportament i la configuració en els servidors Apache, IIS i Tomcat. Finalment, en el quart capítol s'explica l'aplicació que s'ha realitzat com a complement del estudi i que és una mostra de com es poden aprofitar o analitzar la informació que proporcionen els errors en servidors web. L'aplicació es divideix en dos blocs: per un costat l'aplicació base amb eines bàsiques que permet desenvolupar diferents mòduls, i per l'altre el primer d'aquests mòduls dedicat a l'anàlisi dels errors HTTP. En resum, l'aplicació realitza un conjunt de peticions HTTP a un grup de servidors i mostra un informe amb tota la informació obtinguda a partir de les capçaleres HTTP i els missatges d'error obtinguts. Aquesta eina pot ser utilitzada per un atacant com a punt de partida per realitzar un atac, o per els mateixos administradors de sistemes i auditors de seguretat per poder llistar aquells servidors que no estan configurats correctament.

This document is divided in four big sections. The first section is a general overview of the information gathering using the errors shown in a web application in each of the different layers: server, application and data base. It explains what OWASP is, how sensible information can be obtained from errors, how to search this information through Google and how attackers cold take advantage of this information. A real case with Banc Sabadell is used in this chapter. The second section contains specific information about the errors we can get from a HTTP server, doing a list of that errors that can be provoked, how to do it and what information do they provide: time-outs, methods not valid, bad requests,…. The third section contains a detailed analysis of the error HTTP 404 – Not found, its behavior and its configuration in the servers Apache, IIS and Tomcat. Finally, the fourth section contains an application that has been developed to demonstrate how to take advantage and analyze the information provided by the errors in the web servers. The application is divided in two parts. Firstly there is a base application with basic tools that developers can use to create new modules, and secondly the first of that modules that will make an analysis of the HTTP errors. To summarize, the application makes several HTTP requests to a group of defined servers and it creates a report with all the information gathered from the HTTP headers and the error messages. This tool can be used by an attacker as a first step to plan his attack, or by the System Administrators and Auditors to list the servers that are not configured correctly.

Taula de Continguts

Introducció

OWASP (Open Web Application Security Project) és una organització internacional sense ànim de lucre que fomenta el desenvolupament d'aplicacions web segures mitjançant la divulgació d'informació.

El projecte amb més renom d'OWASP és el Top 10 que publiquen periòdicament amb les vulnerabilitats més comunes en entorns web. Un altre projecte important que desenvolupa OWASP és la "Testing Guide" on, no només tracten els problemes més populars, sinó que proporcionen informació sobre totes les vulnerabilitats conegudes en entorns web i classificades per categories segons temàtica.

La temàtica escollida per a realitzar aquest Treball de Final de Màster és "4.9.1 Analysis of Error Codes", dins de la categoria "4.9. Error Handling".

Aquesta tècnica consisteix en obtenir informació a partir dels codis d'error que mostra l'aplicació quan aquesta falla. Els errors poden ser casuals o provocats en diferents graus. Tot i que és una categoria diferent, està relacionada amb la compilació d'informació (4.2 Information Gathering) ja que permet obtenir informació sobre l'aplicació web que després s'utilitzarà per executar altres tècniques.

S'ha escollit aquesta temàtica perquè la informació que proporcionen els errors acostuma a ser obviada pels encarregats de configurar els serveis i les aplicacions. Treballar sobre aquesta tècnica permet aprendre sobre els diferents errors que es poden donar en un servei o aplicació, com es poden provocar, quina informació proporcionen i finalment com configurar-los i protegir-se d'ells.

S'han establert dos objectius per aquest treball:

1) Crear els fonaments per l'anàlisi d'errors en aplicacions web, a través d'un estudi general sobre quins errors es poden aprofitar, i una aplicació per poder automatitzar l'anàlisi.
2) A partir de la base anterior, aprofundir en l'estudi d'un dels errors i realitzar el complement de l'aplicació per poder analitzar-ho automàticament.

Per realitzar l'estudi primer es realitzarà una investigació general sobre els tipus d'error dels quals es poden obtenir informació. Tot seguit s'ampliarà l'estudi dels errors que es poden provocar a nivell HTTP i servidor web, i finalment ens centrarem en un d'aquests errors per aprofundir en detall.

L'aplicació consistirà en una base que contemplarà els automatismes de les proves, l'enviament de peticions HTTP, l'obtenció del resultat i l'anàlisi d'aquest. L'aplicació permetrà afegir mòduls que aprofitaran la base per l'estudi d'un error determinat. S'inclourà un mòdul per poder analitzar específicament l'error estudiat.

Per a complementar l'estudi i l'aplicació, s'utilitzaran tres laboratoris amb diferents servidors web per poder diferenciar el comportament i la informació que proporcionen cadascun d'ells i per poder fer proves amb l'aplicació.

TEMA 1 – Introducció als errors web

Hi ha organitzacions com al MITRE i la National Vulnerability Database (NVD) que compilen i gestionen CVEs (*Common Vulnerabilities and Exposures*), bases de dades sobre vulnerabilitats en serveis i sistemes, on afegeixen informació de quines versions estan afectades, com es pot explotar, quin risc suposen,.... A cada vulnerabilitat s'assigna un codi CVE- que s'utilitza com a referència en molts altres entorns, per exemple, en la publicació de les actualitzacions que solucionen aquestes vulnerabilitats, o en els *workarounds*.

Altres organitzacions, com ara Offensive Security (www.exploitdb.com), permeten que a partir del codi CVE- puguem trobar *exploits* que ens permeten aprofitar la vulnerabilitat a la que fan referència.

D'aquesta manera, coneixent una versió determinada d'un software o servidor, podrem fer una cerca de les vulnerabilitats conegudes i, per cada vulnerabilitat, podrem cercar els *exploits* que ens permetin explotar-la.

Si el software està actualitzat a una versió recent, serà més complicat trobar una vulnerabilitat a explotar, a no ser que existeixi un *zero day* actiu. Si el software no està actualitzat a la ultima versió, és més probable que puguem trobar una vulnerabilitat que puguem explotar.

En un entorn web, l'únic pas que ens falta per poder realitzar un atac és conèixer quina és la versió de software que està utilitzant el servidor web. Hi ha diferents tècniques i eines per descobrir-ho i una d'elles és analitzar el comportament d'aquest software quan respon als errors. En alguns casos ens proporcionarà directament la versió del software, en altres, podem crear un catàleg de quines respostes esperem per cada sistema i versió.

Tot i així, no només es tracta de conèixer la versió del software, alguns errors poden donar altra informació addicional important com ara rutes del servidor, informació sobre les taules de les bases de dades, tecnologies utilitzades...

Com s'analitza en els següents capítols, hi ha dues maneres d'obtenir aquesta informació. Per un costat, podem fer una cerca de pàgines que mostrin errors, interpretar-los i extreure informació interessant d'ells. Per l'altre costat, podem provocar els errors per tal que ens proporcionin la informació que estem cercant.

1.1. OWASP i Error Handling

La *OWASP Testing Guide* dedica un capítol a la gestió dels errors: 4.9 Error Handling. La guia explica que és important que l'aplicació web falli de manera segura quan es produeixen errors i que no proporcionin cap tipus d'informació sensible. Això no evitarà que es produeixin atacs, però posarà més dificultats.

Aquest capítol consta de dos subapartats:

- 4.9.1 Analysis of Error Codes (OTG-ERR-001)
- 4.9.2 Analysis of Stack Traces (OTG-ERR-002)

El primer dels apartats classifica els errors en tres categories: Servidors Web, Aplicacions de servidor i Bases de dades (aquesta és la classificació que s'utilitzarà en el proper capítol per descriure en detall cadascuna d'elles). El segon apartat tracta de les *Stack Traces*, que són errors que proporcionen alguns llenguatges de programació per informar del motiu i la localització del error.

Els dos capítols de la guia estan en l'estat "Esborrany" perquè hi ha informació encara pendent de publicar.

1.2. Un exemple real: el Banc Sabadell

Com a exemple de la importància de gestionar els errors, utilitzarem la pàgina web del Banc Sabadell. Al ser la pàgina web d'un banc, les mesures de seguretat que s'apliquen a ella acostumen a ser altes. Si intentem forçar l'error 404 en la web accedint a la URL https://www.bancsabadell.com/test obtenim el resultat de la figura 1.

Figura 1. Pàgina no trobada – Error 404 del Banc Sabadell

Però si utilitzem la URL a la que ens redirigeix la pàgina d'inici i afegim un text qualsevol al final: https://www.bancsabadell.com/cs/Satellite/SabAtl/test sí que ens mostrarà un error d'Apache Tomcat – JBOSS on es proporciona informació sensible: utilitzen la versió de JBOSS 2.0.0.

Figura 2. Pàgina no trobada – Error 400 del Banc Sabadell utilitzant la URL /cs/Staellite/SabAtl/

Per altra banda, durant una incidència en els centres de dades del Banc Sabadell el 19 de novembre del 2013, la pàgina web del Banc Sabadell va estar no disponible durant uns minuts. La pàgina que es mostrava per defecte a tots els clients en entrar a la web va ser un error 404 que mostrava també la versió de JBOSS i uns minuts més tard un error 502 que en aquest cas no mostrava informació de la versió.

Figura 3. Pàgina no trobada – Error 404 del Banc Sabadell durant una incidència en el centre de dades

Figura 4. Proxy Error – Error 502 del Banc Sabadell durant una incidència en el centre de dades

Amb aquest error el Banc Sabadell ens està donant informació de que està utilitzant la versió 2.0.0 de JBOSS Web que és antiga. La ultima versió publicada és la 2.0.10 d'agost del 2011 i actualment ja forma

part del paquet JBOSS Application Server. Sobre aquesta versió existeixen tres vulnerabilitats conegudes i identificades amb els següents CVE:

- CVE-2013-1976: usuaris locals poden canviar de propietari d'arxius mitjançant un atac *symlink*
- CVE-2007-6433: permet executar comandes Enterprise Java Beans – Query Language mitjançant les ordres de paràmetres
- CVE-2005-2158: permet executar codi arbitrari a atacants remots

Tot això no vol dir que la pàgina web sigui vulnerable a atacs, ja que podrien haver aplicat *patches* específics per aquestes vulnerabilitats, però és un punt de partida important perquè un atacant ho pugui utilitzar per iniciar l'atac.

1.3. Google Hacking

Una tècnica molt utilitzada recentment és el Google Hacking, que consisteix en utilitzar la potencia cercadora i de filtratge de Google per trobar aplicacions web vulnerables.

Un exemple molt senzill de cerca és *intitle:index.of "parent directory"* que, tot i que donarà molts falsos positius, mostra milions de llocs web que tenen el *directory listing* del servidor web actiu i per tant es pot navegar a través de l'estructura d'arxius d'aquests servidors.

Google Hacking es pot utilitzar també per realitzar cerques d'errors en servidors, aplicacions o bases de dades. En el cas de tenir un objectiu definit, es pot limitar la cerca de Google amb el filtre *site:www.lloc_atacat.com* i cercar diferents patrons que segueixin els errors. Per altra banda, si no es té un objectiu definit però es coneix un patró d'error determinat, es pot fer una cerca a Google d'aquest patró per trobar ells llocs que seran vulnerables o que ens proporcionin la informació que estem cercant.

Moltes cerques són útils només si es limita per un *site*, ja que són missatges d'error comuns i hi ha centenars de resultats de gent que pregunta sobre aquest mateix error en fòrums. Per fer cerques globals s'haurà de filtrar per *title* o *filetype* per poder obtenir resultats útils.

A continuació es mostren alguns exemples de cerques que aprofiten els errors mostrats per l'aplicació web o el servidor i que proporcionen informació sobre tecnologies i versions i, fins i tot, la ruta d'instal·lació o taules i camps de la base de dades.

Cerca: "intitle:"Object not found!" intext:"Apache/2.0.* (Linux/SuSE)"

Figura 5. Pàgina web regioart.net que mostra un Error 404 degut a un objecte no trobat. L'error mostra la versió d'Apache/2.0.54 i el sistema operatiu Linux/Suse

Cerca: intext:"404 Object Not Found" Microsoft-IIS/5.0

Figura 6. Pàgina web gitlactik.com que mostra un Error 404 degut a un objecte no trobat. L'error mostra que el servidor utilitzà la versió 5.0 de IIS

Cerca: "A syntax error has occurred" filetype:ihtml

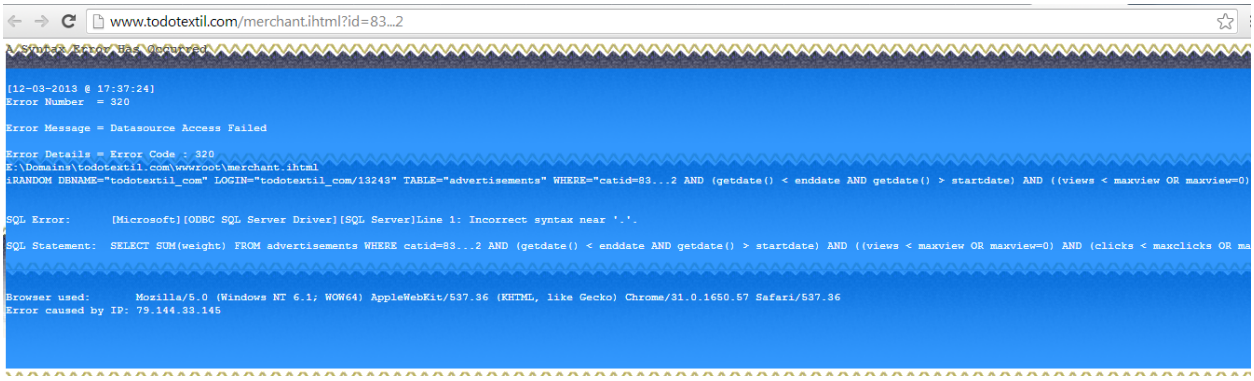

Figura 7. Pàgina web todotextil.com que mostra un error de SQL on es mostra la ruta del arxiu, el login, taules i elements de la base de dades.

Tot i que Google es el cercador mes utilitzat, també es poden utilitzar altres proveïdors com ara Bing. S'ha de tenir en compte que cada cercador es comporta d'una manera diferent i conté una base de dades diferent, per tant s'ha de considerar altres cercadors com a fonts d'informació alternatives.

1.4. Altres tècniques per obtenir informació

Existeixen moltes altres tècniques per obtenir informació d'aplicacions web en qualsevol de les diferents capes (servidor, aplicació, base de dades). Algunes d'aquestes tècniques són les que utilitzen eines com Nmap (http://nmap.org) o Shodan (http://www.shodanhq.com) i consisteixen en fer una anàlisi del *fingerprint* del servidor mitjançant peticions a la capta d'IP i transport i l'anàlisi de les respostes obtingudes.

Aquestes tècniques es poden considerar més fiables en certes situacions, però en qualsevol cas són complementàries. En general, podem obtenir informació dels errors que els *banners* o *fingerprints* no ens proporcionen. En molts casos podrem obtenir el mateix tipus d'informació, però en determinats casos ens serà útil disposar de tècniques alternatives quan una o altra falli.

TEMA 2 – Estat de l'art del problema

2.1. Tipus d'errors web

La *OWASP Testing Guide* classifica els errors web en tres categories: errors en servidors web, errors en aplicacions i errors en bases de dades.

Cadascun d'aquests grups requereix una especialització diferent i els professionals que gestionen aquests entorns pertanyen a branques diferents de la informàtica: administradors de sistemes, programadors i administradors de bases de dades.

En aquest apartat es realitza una descripció i visió general de cadascuna d'aquestes categories, detallant la informació que poden proporcionar i quins són els sistemes afectats.

2.1.1. Errors en servidors web

Els errors en servidors web són la primera de les capes en la que podem trobar errors i aquella més propera als sistemes.

Els servidors web, a un nivell molt bàsic, s'encarreguen de respondre a les peticions HTTP dels usuaris. HTTP és un protocol que té tres versions 1.0, 1.1 i 2, definides en els RFC 1945, 2616 i 2774 respectivament. Actualment s'utilitza la versió 1.1, ja que la versió 2 està en fase experimental.

Aquest protocol defineix un conjunt de codis d'estat, identificats per un número. Per exemple, els codis 100 són d'informació i els 200 indiquen que la petició s'ha realitzar correctament. Els codis 400 són aquells que s'utilitzen per informar d'un error en el client i els codis 500 indiquen errors en el servidor.

Les respostes dels diferents servidors a cadascun d'aquests codis d'errors són les que podem utilitzar per obtenir informació sensible.

Els objectius d'aquests tipus d'errors són els servidors web. Apache, IIS i Tomcat són els més populars, tot i que hi ha molts altres tipus de servidor web que també s'han de considerar, com ara JBOSS, basat en Tomcat.

En la gran majoria dels casos, la informació sensible que podrem obtenir dels errors de servidor són únicament la versió del propi servidor, el sistema operatiu i els components i versions instal·lats en el servidor.

En l'apartat 2.2 d'aquest capítol s'amplia la informació sobre els errors en els servidors web.

2.1.2. Errors en aplicacions

Per sobre del servidor web s'instal·len les aplicacions. Aquestes poden ser de molts tipus i utilitzar moltes tecnologies diferents, des del llenguatge de programació utilitzat fins els diferents complements o mòduls que es poden instal·lar en un servidor.

Habitualment, una aplicació web estarà feta amb HTML, disposarà de CSS i probablement JavaScript. En alguns casos podem trobar també XML.

Addicionalment, una aplicació pot utilitzar un llenguatge de programació per incrementar la seva funcionalitat, que podria ser ASP, PHP o JSP. També es poden utilitzar programes desenvolupats de manera més tradicional a través de CGIs que poden utilitzar C, C++, Perl o Python. IIS també permet l'execució de programes en Visual Basic i .Net, i Tomcat permet executar programes en Java.

Finalment hi ha desenes de mòduls addicionals que es poden afegir en els servidors web per realitzar funcions concretes. Aquests mòduls en molts casos estan lligats a l'aplicació o al servidor.

Qualsevol d'aquestes tecnologies pot causar un error en l'aplicació i el mètode habitual per mostrar l'error al programador o usuari és mostrar un missatge per pantalla. Algunes d'elles no proporcionaran informació útil per un atacant, però d'altres sí que mostraran informació sensible que pot ser utilitzada per iniciar un atac.

Cada una d'aquestes tecnologies mereix un estudi independent sobre els errors que pot generar, els que es poden provocar, quins proporcionen informació sensible i quines són les bones pràctiques per evitar mostrar aquesta informació.

Dins d'aquesta categoria també s'han d'incloure els paquets d'aplicacions instal·lables: gestors de continguts (Wordpress, Drupal...), eines empresarials (SugarCRM, OpenERP...), fòrums (vBulletin, phpBB)...

Cadascun d'aquests paquets està escrit en un llenguatge determinat dels comentat prèviament i per tant els errors que pugui generar estan relacionats amb ells, però també tenen els seus propis errors que poden proporcionar informació sensible i ajudar a identificar aspectes de l'aplicació que després es podran utilitzar per trobar vulnerabilitats.

El tipus d'informació que podem obtenir dels errors en les aplicacions són: tecnologies utilitzades, rutes del servidor i noms d'arxius.

2.1.3. Errors en bases de dades

Les bases de dades estan molt relacionades amb l'aplicació i és aquesta qui hi accedeix i qui mostrarà els errors. Tot i així es pot dedicar un estudi complet a parlar dels errors que pot mostrar cadascun dels sistemes de bases de dades.

Els sistemes més coneguts són MySQL, PostgreSQL, Access, Microsoft SQL i Oracle.

Com a característica afegida en aquesta categoria tenim que un dels atacs més comuns en aplicacions web consisteix en les injeccions SQL. Això fa que existeixi molta documentació sobre com realitzar injeccions i molt interès en extreure informació de la base de dades a partir dels errors.

La informació sensible que podem obtenir dels errors en bases de dades són: tecnologies utilitzades, noms de les bases de dades, taules i columnes, servidors de bases de dades, usuaris i fins i tot contrasenyes.

MySQL ha documentat públicament tots els errors possibles per part de servidor i client i es poden veure recopilats en l'annex "Inventari d'errors". Només aquesta tecnologia disposa de 948 errors, molts dels quals no proporcionaran informació sensible i molts altres no podran ser provocats, però tot i així hi haurà alguns que sí proporcionin informació i estaran indexats a Google i altres podran ser provocats i en podrem treure informació.

2.2. Errors en servidors web

Com s'ha explicat en el capítol anterior, els errors en servidor web es limiten a les respostes del protocol HTTP que rep el client quan es produeix un error.

Hi ha dos grups de codis de resposta dedicats als errors:

1- 400, indica que la petició conté sintaxi incorrecta
2- 500, indica que el servidor ha fallat intentant respondre una petició aparentment correcta

2.2.1. Llistat d'errors

El RFC 2616 explica el funcionament del protocol HTTP 1.1 i allí es descriuen els possibles errors 4xx i 5xx, que són els següents:

- 400 Bad Request
- 401 Unauthorized
- 402 Payment Required
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Time-out
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URI Too Large
- 415 Unsupported Media Type
- 416 Requested range not satisfiable
- 417 Expectation Failed
- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Time-out
- 505 HTTP Version not supported extension-code

Altres RFCs han afegit errors addicionals i certes implementacions de servidor han creat els seus propis codis d'error. L'annex "Inventari d'errors" conté el llistat de 60 codis de resposta 400 i 500 en HTTP.

El protocol HTTP també defineix que els clients han de considerar qualsevol error desconegut 4xx o 5xx de manera genèrica com a 400 o 500.

L'error més senzill d'explotar i que sovint proporciona informació útil és l'error 404 i s'explicarà en detall en el Capítol 3.

2.2.2. Exemples d'explotació d'errors

Telnet s'utilitza per enviar peticions HTTP específicament formades per causar errors i poder rebre els missatges que el servidor retorna de manera íntegra.

Per fer-ho, executem un Telnet cap el servidor de prova utilitzant el port 80

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
```

Figura 8. Establiment de la connexió Telnet contra el servidor del laboratori.

Tot seguit realitzarem la petició HTTP, que té la següent estructura:

GET /index.htm HTTP/1.1
host: 192.168.195.145

Un cop introduït aquest text, obtindrem la resposta del servidor, tant les capçaleres HTTP com les dades enviades com a codi HTML.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /index.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 200 OK
Date: Fri, 06 Dec 2013 20:12:50 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Sat, 30 Nov 2013 13:23:00 GMT
ETag: "a1fec-b1-4ec64d884e138"
Accept-Ranges: bytes
Content-Length: 177
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
Connection closed by foreign host.
genis@MATRIX:~$
```

Figura 9. Exemple de petició HTTP satisfactòria mitjançant Telnet

El codi HTML que es retorna com a resposta de la petició és la pàgina "It works!" d'Apache.

En les capçaleres de la resposta HTTP de la figura 9 podem observar com s'anuncia el servidor "Apache/2.2.22". Aquesta és la informació que ve incorporada en el *banner* i que algunes eines utilitzen per identificar el servidor, per exemple Shodan, tal com s'ha explicat a la introducció del treball. És important diferenciar aquest *banner* dels missatges que proporcionen les respostes d'error ja que són dues tècniques diferents.

Un factor important és que cada error HTTP té les seves característiques en quant a explotació:

3- Errors explotables des d'un navegador web: 400, 403, 404 (són aquells que podrem cercar a través de Google)
4- Errors que requereixen una configuració determinada en el servidor: 403, 500
5- Errors que requereixen modificar la petició HTTP: 400, 408, 501
6- Errors que poden ser provocats: 400, 403, 403, 408, 501

També s'ha de tenir en compte que alguns navegadors com el Internet Explorer modifiquen la resposta que retorna el servidor. Si aquest navegador detecta que la pàgina de resposta és menor a una certa quantitat de bytes, interpreta que no ha estat personalitzada i proporcionarà a l'usuari una resposta pròpia del navegador:

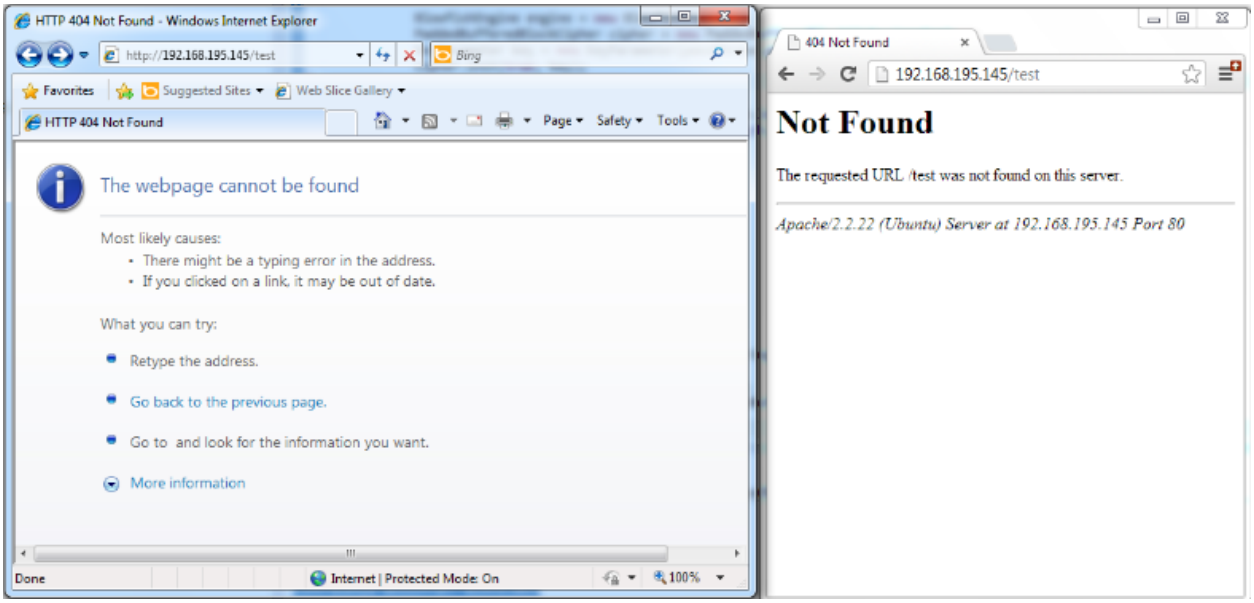

Figura 10. Diferència del mateix error mostrat en els navegadors Internet Explorer i Chrome.

Els següents exemples s'han realitzat utilitzant un servidor Apache. Per analitzar a fons cadascun dels errors s'hauria d'estudiar el comportament en diferents entorns i servidors web.

400 Bad Request:

Per provocar aquest error necessitem enviar una petició mal formada. En aquest cas s'ha generat una petició sense el camp "host".

GET /index.html HTTP/1.1

Podem observar en la figura 11 que el servidor respon amb el codi HTTP 400 i un HTML on s'inclou la versió del servidor.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /index.html HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Fri, 06 Dec 2013 23:57:53 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.22 (Ubuntu) Server at 127.0.1.1 Port 80</address>
</body></html>
Connection closed by foreign host.
genis@MATRIX:~$
```

Figura 11. Petició GET mal formada i resposta 400 del servidor Apache del laboratori

Tot i que aquesta informació pot semblar molt limitada, això és degut a que s'estan realitzant les proves sobre una màquina instal·lada expressament amb només el servidor Apache. Si es disposa de més mòduls instal·lats es pot obtenir molta més informació. És el cas, per exemple, del servidor que allotja el domini www.cmeducation.co.uk:

```
genis@MATRIX:~$ telnet www.cmeducation.co.uk 80
Trying 80.13.223.162...
Connected to www.cmeducation.co.uk.
Escape character is '^]'.
GET /index.html HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Sat, 07 Dec 2013 00:17:31 GMT
Server: Apache/2.2.11 (Ubuntu) mod-xslt/1.3.9 PHP/5.2.6-3ubuntu4.6 with Suhosin-Patch
Vary: Accept-Encoding
Content-Length: 356
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.11 (Ubuntu) mod-xslt/1.3.9 PHP/5.2.6-3ubuntu4.6 with Suhosin-Patc
h Server at 127.0.0.1 Port 80</address>
</body></html>
Connection closed by foreign host.
genis@MATRIX:~$
```

Figura 12. Petició GET mal formada i resposta 400 del servidor Apache de la web cmeducation.co.uk

Observem en la figura 12 que el servidor utilitza Apache 2.2.11 sobre Ubuntu amb el modul xslt 1.3.9, la versió de PHP 5.2.6-3 i disposa del Suhosin Patch.

Aquest no és un tipus d'error que habitualment puguem cercar a Google, ja que requereix una modificació de les capçaleres HTTP. Tot i així hi ha un altre motiu que genera el mateix error que es dóna quan s'intenta accedir a un servidor HTTPS mitjançant una petició HTTP al port 443. El servidor també retornarà l'error 400.

Podem realitzar una cerca a Google amb la següent cadena per trobar llocs que estiguin reportant aquest error i observar quin tipus d'informació addicional proporcionen.

Cerca: "Your browser sent a request that this server could not understand" intitle:"400 Bad Request"

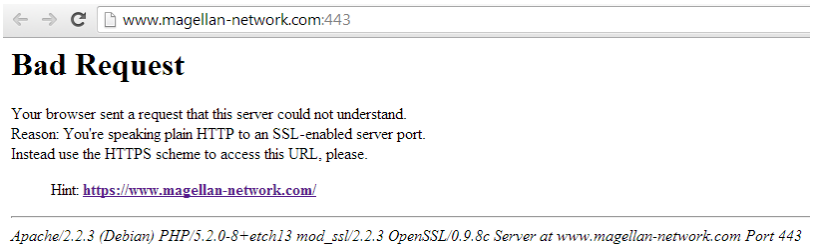

Figura 13. Error 400 - Bad Request de la pàgina magellan-network en accedir per el port 443 però utilitzant el protocol HTTP. Podem observar qeu es mostra la versió del servidor, PHP, mod_ssl i OpenSSL

Si especifiquem una versió determinada que sigui vulnerable, trobarem tots els llocs que pateixin aquest error:

Figura 14. Resultat de la cerca del error “Your browser sent a request...” especificant la versió 5.2.0 de PHP

Limitant la cerca amb el text “PHP 5.2.0” ens apareixen 234 resultats a Google de pàgines que mostren l’error 400 *Bad Request* i tenen la versió de PHP 5.2 instal·lada.

403 Forbidden

HTTP disposa d’un estat d’error per quan l’accés al recurs que es sol·licita és denegat. No podem forçar aquest estat per client, només quan el servidor tingui el recurs protegit.

En aquest cas s’ha modificat un arxiu del servidor de manera que no disposi de permisos pels usuaris amb la comanda *“chmod 600 index.html”*

Fem una petició HTTP a través de Telnet sol·licitant el recurs:

GET /testforbidden/index.htm HTTP/1.1
host: 192.168.195.145

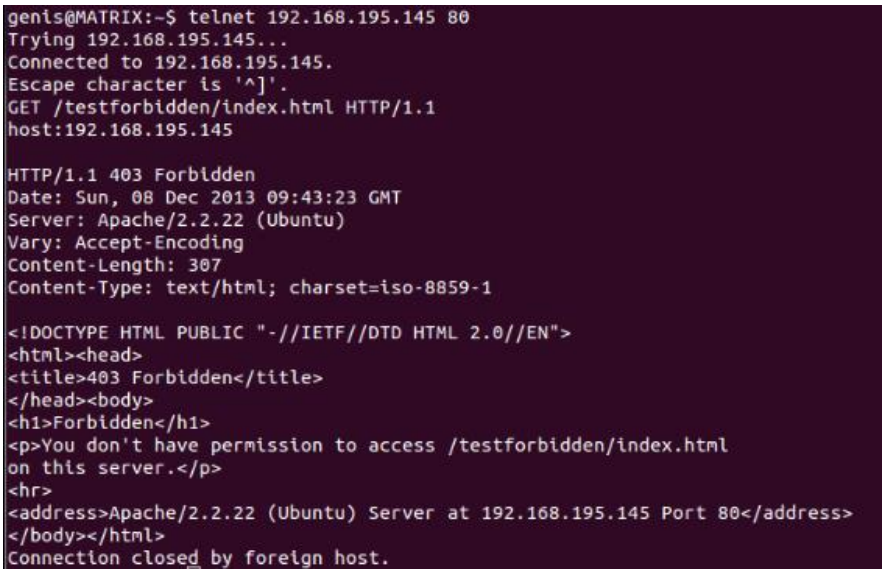

Figura 15. Petició GET contra un recurs protegit i resposta 403 Forbidden del servidor Apache

Observem que la resposta a aquest error conté la versió del servidor Apache tant en la capçalera HTTP com en el missatge HTML de resposta.

405 Method Not Allowed

HTTP proporciona diferents mètodes o comandes que es poden enviar al servidor web. Les més comuns són GET i POST, que serveixen per demanar les pàgines al servidor i enviar informació al servidor a través de formularis o URLs. També existeixen altres mètodes més delicats com són el PUT o DELETE.

DELETE permet eliminar arxius del servidor mentre que PUT permet enviar arxius al servidor. La configuració per defecte i recomanada d'un servidor web és que aquests dos mètodes estiguin deshabilitats, per tant, quan s'intenten executar, el servidor retornarà l'error 405.

Per enviar aquest mètode a través de Telnet utilitzem la següent petició:

DELETE /index.html HTTP/1.1
host: 192.168.195.145

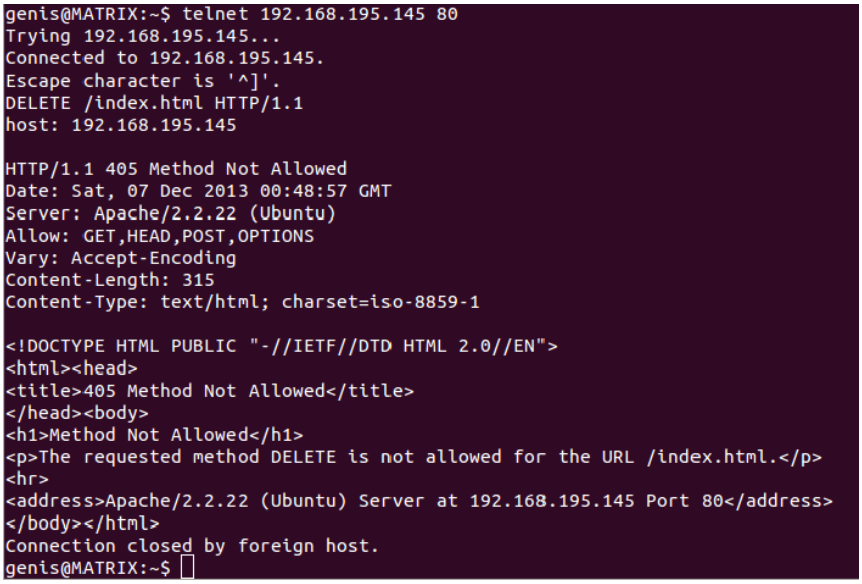

Figura 16. Petició DELETE al servidor Apache i resposta 405 Method Not Allowed

Observem que la resposta del servidor és un codi HTML que proporciona de nou la versió del servidor. La capçalera HTTP informa dels mètodes permesos: GET, HEAD, POST i OPTIONS.

408 Request Time-out

Per forçar un Time-out s'ha d'establir la connexió amb el servidor mitjançant Telnet i llençar la primera part de la petició, sense enviar la part de la comanda que inclou el camp host:

GET /index.html HTTP/1.1

Esperem fins que el servidor superi el temps d'espera i aparegui el missatge d'error.

Figura 17. Resposta d'error 408 Time-out quan només s'envia una part de la petició i el servidor queda a l'espera de la resta d'informació.

Podem observar que la resposta és exactament la mateixa que en els apartats anteriors. Apache retorna un HTML amb el missatge d'error i inclou el nom i la versió de servidor.

501 Method not implemented

Aquest exemple és molt semblat al Method Not Allowed, però enlloc de sol·licitar un mètode que no està permès, sol·licitem un mètode que no existeix en el servidor. En aquest cas no es considera error de la petició del client (4xx) sinó que es considera un error del servidor (5xx)

RENAME /index.html HTTP/1.1
Host: 192.168.195.145

```
Connected to 192.168.195.145.
Escape character is '^]'.
RENAME /index.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 501 Method Not Implemented
Date: Sun, 08 Dec 2013 09:59:32 GMT
Server: Apache/2.2.22 (Ubuntu)
Allow: GET,HEAD,POST,OPTIONS
Vary: Accept-Encoding
Content-Length: 299
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Method Not Implemented</title>
</head><body>
<h1>Method Not Implemented</h1>
<p>RENAME to /index.html not supported.<br />
</p>
<hr>
<address>Apache/2.2.22 (Ubuntu) Server at 192.168.195.145 Port 80</address>
</body></html>
Connection closed by foreign host.
genis@MATRIX:~$
```

Figura 18. Resposta d'error 501 Method not Implemented quan s'envia una petició RENAME

2.3. Un altre exemple real – loteriadecatalunya.cat

En el Capítol 1 s'ha pogut veure com la informació proporcionada per els servidors pot ser utilitzada per els atacants amb l'exemple del Banc Sabadell. Ara aprofitarem els coneixements del Capítol 2 per veure un altre exemple d'exposició d'informació sensible degut als errors. Es tracta de la pàgina web de Loteria de Catalunya.

Si realitzem una petició HTTP correcta, obtenim les següents capçaleres, on no podem veure la versió d'Apache que utilitza l'organització. Només apareix el terme "Apache".

```
genis@MATRIX:~$ telnet www.loteriadecatalunya.cat 80
Trying 81.88.48.71...
Connected to loteriadecatalunya.cat.
Escape character is '^]'.
GET / HTTP/1.1
host: www.loteriadecatalunya.cat

HTTP/1.1 200 OK
Date: Wed, 01 Jan 2014 20:08:30 GMT
Server: Apache
Last-Modified: Tue, 04 Dec 2007 11:16:33 GMT
Accept-Ranges: bytes
Content-Length: 708
Content-Type: text/html
Content-Language: es
```

Figura 19. Petició GET satisfactòria 200 OK a la pàgina web de loteriadecatalunya.cat

Podem realitzar una petició a una pàgina no existent en el servidor per obtenir el següent error, on tampoc es mostra cap informació sobre la versió del servidor.

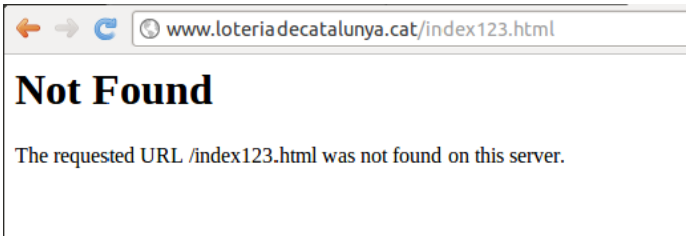

Figura 20. Error 404 Not found a la pàgina web de loteriadecatalunya.cat

Realitzant aquestes proves sembla que la pàgina web està configurada de manera apropiada, però durant el sorteig de la Grossa el 31 de Desembre, possiblement degut a problemes de capacitat, es mostrava el següent error "502" en accedir a la pàgina web principal. A la figura 21 podem veure que el servidor està utilitzant Apache 1.3.12.

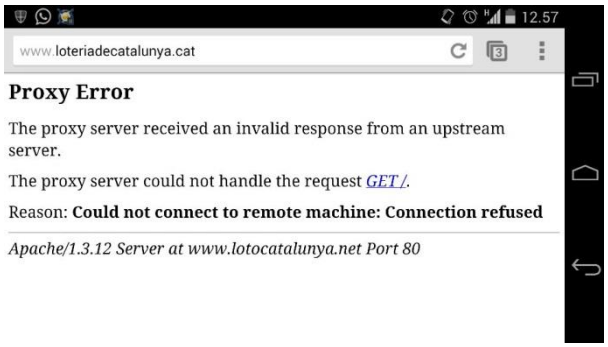

Figura 21. Resposta 502 Proxy Error a la pàgina web de loteriadecatalunya.cat

La versió d'Apache 1.3.12 és una versió molt antiga i sense suport d'Apache, publicada el 25 de febrer del any 2000 i que probablement és vulnerable a atacs.

L'error 502 – Bad Gateway o Proxy Error es un error de servidor que no pot ser provocat sense utilitzar tècniques de denegació de servei, per tant, de manera externa, no podem comprovar si està correctament configurat o no.

2.4. Aplicacions que aprofiten errors web i eines disponibles

S'ha vist en els apartats anteriors que hi hauria dos maneres d'aprofitar el fet que els missatges d'error proporcionin informació sensible:

1- Cerca d'errors existents per obtenir informació.
2- Forçar errors per obtenir informació.

No hi ha eines disponibles a internet específiques per realitzar la recopilació d'informació a partir dels errors en aplicacions web. La única que ho utilitza és la Foca. Però hi ha altres eines que ens poden ser útils per poder aprofitar la informació dels errors web.

FOCA

La FOCA és una eina dedicada al *information gathering*. Utilitza desenes de tècniques per obtenir informació d'un objectiu. No està documentat quin és el funcionament de l'eina, però en la presentació que van fer a la Defcon 18 es pot observar com les tècniques 22 i 23 del llistat on presentaven l'algorisme de descobriment de xarxa utilitzen els errors 404 i els missatges d'errors en aplicacions.

Tot i així, no apareix en els registres de l'eina ni en la configuració cap referència a aquestes tècniques.

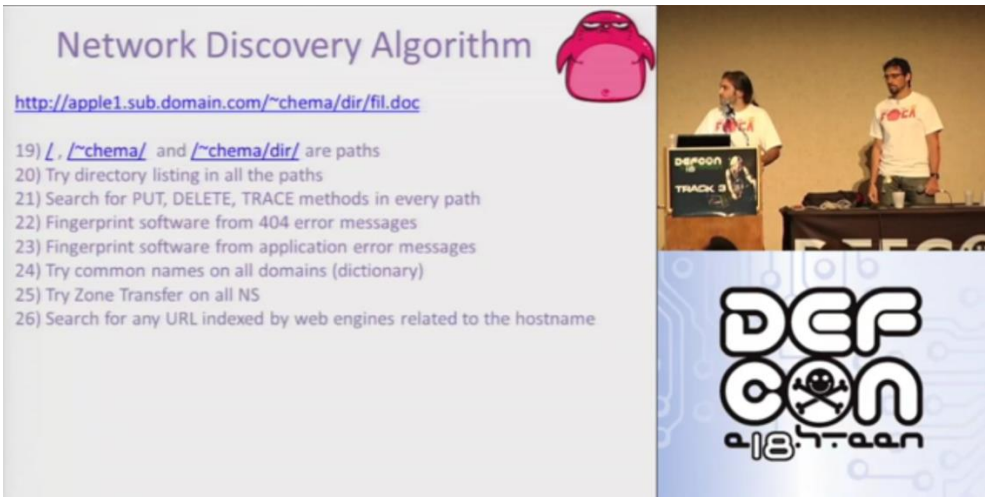

Figura 22. Captura de la conferencia on s'explica la utilització d'errors 404 i errors en aplicacions per obtenir informació sensible en FOCA

Google

En aquest treball s'han utilitzat tècniques de cerques en Google per trobar errors en pàgines web. No és una eina dissenyada per aquesta funció, però la seva capacitat de cerca i les funcionalitats avançades per restringir cerques com ara *site*: i *intitle*:, ens permeten trobar pàgines que continguin errors.

Es especialment útil si no tenim un objectiu en concret o si coneixem el comportament d'un error i podem utilitzar filtres per descartar els falsos positius.

Per altra banda, Google no conté l'estat real de la pàgina, sinó que conté una *cache* del que els seus robots han trobat i registrat, per tant el resultats de les cerques poden no correspondre a la realitat. En alguns casos, Google elimina de la seva *cache* les pàgines que mostrin errors.

Com a avantatge, és pràcticament impossible que hi hagi cap altra eina amb la potència de cerca de la que disposa Google o altres cercadors de grans empreses (Bing o Yahoo també poden utilitzar-se).

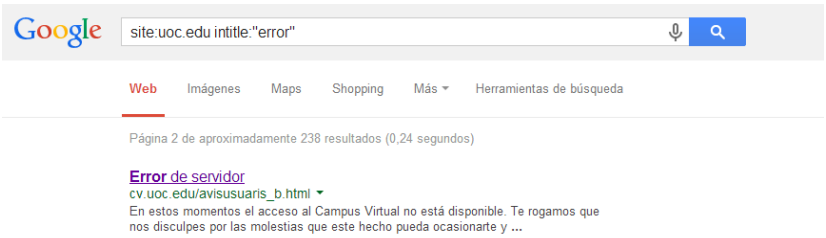

Figura 23. Exemple de cerca a Google restringida al *site* de la UOC amb la paraula Error en el títol

Web Scrapers

Aquest tipus d'utilitats no són eines que aprofiten els codis d'error per obtenir informació, però sí són eines que es poden utilitzar per detectar errors en pàgines web. Bàsicament el que fan és navegar a través del lloc web seleccionat per extreure informació i analitzar-la. Tot i que Google ens fa una funció semblant, no tenen per què produir els mateixos resultats. Un *Web Scraper* obtindrà la informació real que està mostrant la pàgina, mentre que Google mostrarà la informació en la seva cache i pot haver descartat errors. Alguns *Web Scrapers* són Inspyder (http://www.inspyder.com/), FMiner (http://www.fminer.com), Deixto (http://deixto.com/) o Mozenda (http://www.mozenda.com), tot i que hi ha desenes d'eines amb funcionalitat similar.

Web Server Fingerprint

La tècnica del *Fingerprint* consisteix en identificar el servidor a través de diferents propietats i característiques de les respostes HTTP. La OWASP Testing Guide dedica un capítol a aquesta tècnica: Fingerprint Web Server (OTG-INFO-002).

Un dels mètodes amb el que es realitza el *fingerprint* és el que s'ha vist en les proves d'aquest capítol on el valor del camp *Server* de la capçalera HTTP proporciona la versió del servidor. En el proper capítol veurem com aquesta informació està relacionada amb la informació proporcionada pels missatges d'error.

Cada servidor web té la seva implementació del protocol HTTP i respon d'una manera o una altre a les peticions HTTP. Com que la capçalera Server es pot modificar i falsificar, aquests programes també utilitzen aquest comportament per identificar la versió del servidor i corroborar la versió proporcionada per la capçalera Server de HTTP.

HTTPrint (http://net-square.com/httprint.html), NetCraft (http://www.netcraft.com) i HTTPrecon (http://www.computec.ch/projekte/httprecon/) són algunes aplicacions que realitzen el *Web Server Fingerprinting*.

TEMA 3 – El cas específic de l'error 404

3.1. Introducció a l'error 404

La resposta d'error amb el codi 404 d'HTTP és la més popular i més senzilla de provocar dins dels errors a nivell de servidor web. Aquest codi d'error es retorna quan el servidor no troba la pàgina que el client està demanant. Quan es vol provocar aquest error tan sols és necessari modificar la URL del navegador i posar un conjunt de caràcters aleatori.

Per exemple, podem veure a la figura 24 la resposta 404 que envia la pàgina web www.todocoleccion.com, mitjançant la següent petició: www.todocoleccion.com/test123.html

Figura 24. Resposta 404 not found a todocoleccion.net

Com que és l'error més senzill de provocar, també és pel que hi ha més conscienciació sobre que s'ha de protegir, per un costat per evitar proporcionar informació sensible, i per l'altre per donar millor imatge en quan l'usuari troba un error.

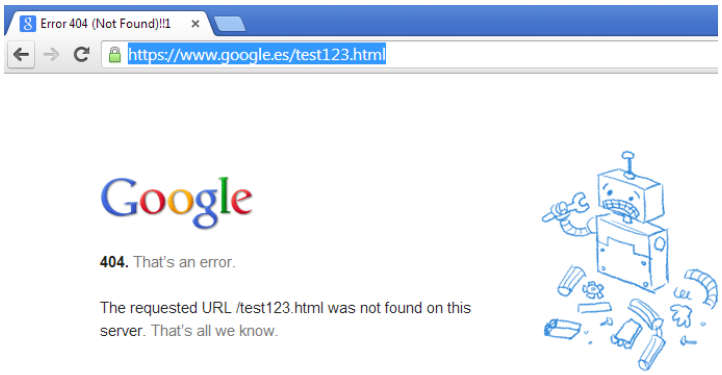

Figura 25. Resposta 404 not found a google.es

En alguns casos, les pàgines web proporcionen eines de cerca o el mapa de navegació del lloc web, per tal que l'usuari pugui fer una cerca a la web de la informació que està demanant.

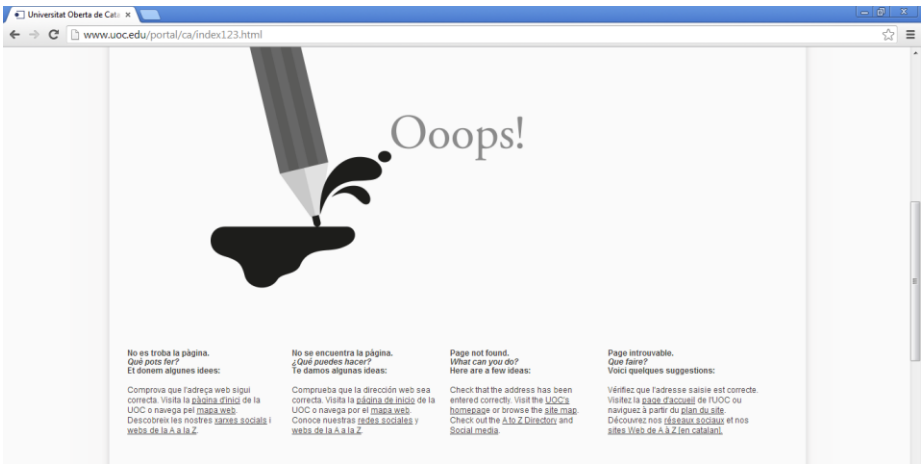

Figura 26. Resposta 404 not found a uoc.edu

Finalment, hi ha casos en que aquest error pot proporcionar informació sobre la versió del servidor i els mòduls utilitzats.

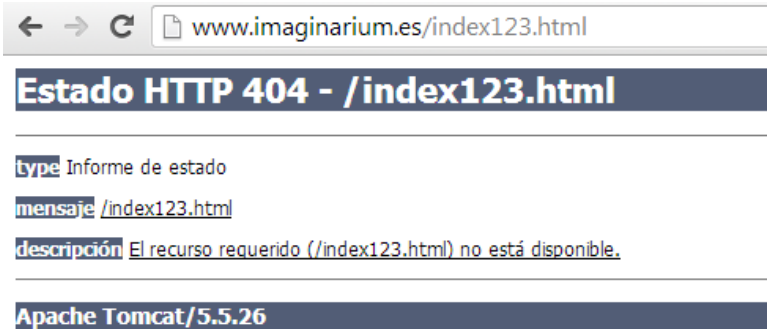

Figura 27. Resposta 404 not found a imaginarium.es. La versió actual de Tomcat és la 7

3.2. L'error 404 en diferents servidors

En aquest apartat estudiem el comportament per defecte de cadascun dels servidors, quina informació proporcionen i quina és la configuració necessària per evitar mostrar aquesta informació.

3.2.1. L'error 404 en IIS

IIS 8 és el servidor web que forma part del Microsoft Windows Server 2012 R2. IIS per defecte mostra la pàgina d'error que es mostra en la figura 28.

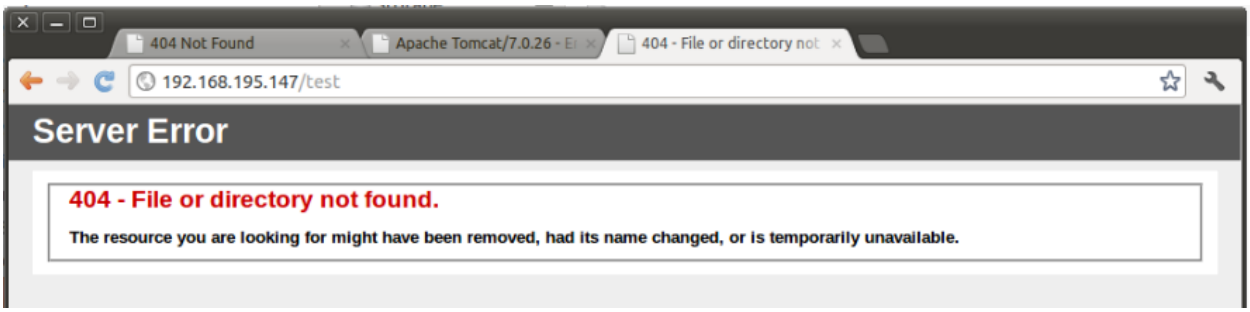

Figura 28. Pàgina 404 per defecte en un servidor IIS

Si realitzem una petició HTTP mitjançant Telnet obtenim la següent informació.

```
genis@MATRIX:~$ telnet 192.168.195.147 80
Trying 192.168.195.147...
Connected to 192.168.195.147.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.147

HTTP/1.1 404 Not Found
Content-Type: text/html
Server: Microsoft-IIS/8.5
Date: Wed, 18 Dec 2013 20:41:52 GMT
Content-Length: 1245

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/x
html1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>404 - File or directory not found.</title>
<style type="text/css">
<!--
body{margin:0;font-size:.7em;font-family:Verdana, Arial, Helvetica, sans-serif;b
ackground:#EEEEEE;}
fieldset{padding:0 15px 10px 15px;}
h1{font-size:2.4em;margin:0;color:#FFF;}
h2{font-size:1.7em;margin:0;color:#CC0000;}
h3{font-size:1.2em;margin:10px 0 0 0;color:#000000;}
#header{width:96%;margin:0 0 0 0;padding:6px 2% 6px 2%;font-family:"trebuchet MS
", Verdana, sans-serif;color:#FFF;
background-color:#555555;}
#content{margin:0 0 0 2%;position:relative;}
.content-container{background:#FFF;width:96%;margin-top:8px;padding:10px;positio
n:relative;}
-->
</style>
</head>
<body>
<div id="header"><h1>Server Error</h1></div>
<div id="content">
 <div class="content-container"><fieldset>
  <h2>404 - File or directory not found.</h2>
  <h3>The resource you are looking for might have been removed, had its name cha
nged, or is temporarily unavailable.</h3>
 </fieldset></div>
</div>
</body>
</html>
```

Figura 29. Resposta 404 not found en un servidor IIS

Podem observar que al *banner* de la petició apareix la versió del servidor. Com s'ha explicat en el capítol anterior, aquesta informació no és pròpia del codi d'error 404, sinó que forma part de les capçaleres de la petició HTTP. Realitzant una petició correcta obtenim la mateixa informació.

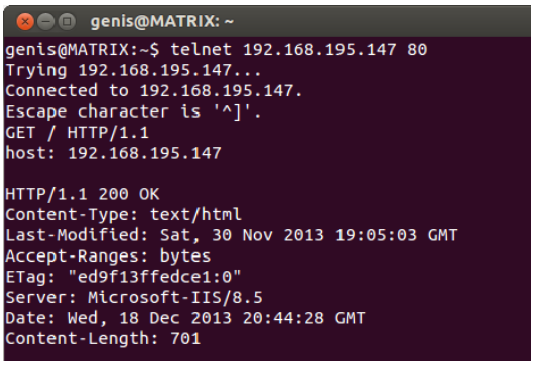

Figura 30. Resposta satisfactòria 200 en un servidor IIS

Segons aquestes proves podem afirmar que IIS 8.5 no proporciona informació sensible a partir dels errors HTTP 404.

Configuració del error 404 a IIS

Com tots els servidors web, IIS permet personalitzar la resposta als errors 404. IIS mostra aquesta opció entre les seves opcions principals en la configuració del *Site*, en l'apartat *Error Pages*.

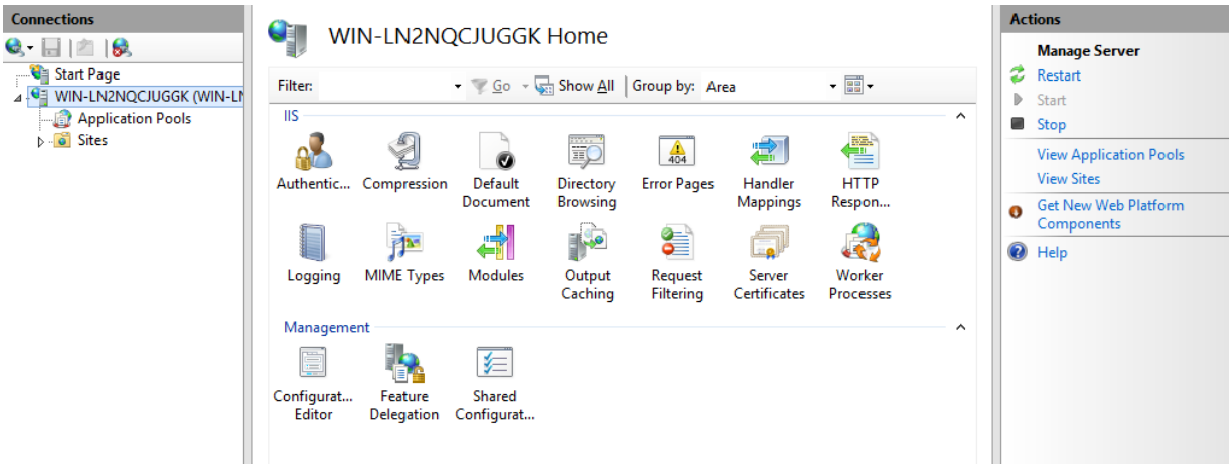

Figura 31. Panell d'administració del servidor IIS

En aquesta secció no només es pot configurar l'error 404, sinó que IIS permet configurar els errors HTTP més comuns, alguns dels que s'ha tractat en el Capítol 2. Cada error fa referència a un arxiu en el disc i es pot optar per apuntar els errors a una pàgina personalitzada o simplement modificar els arxius indicats.

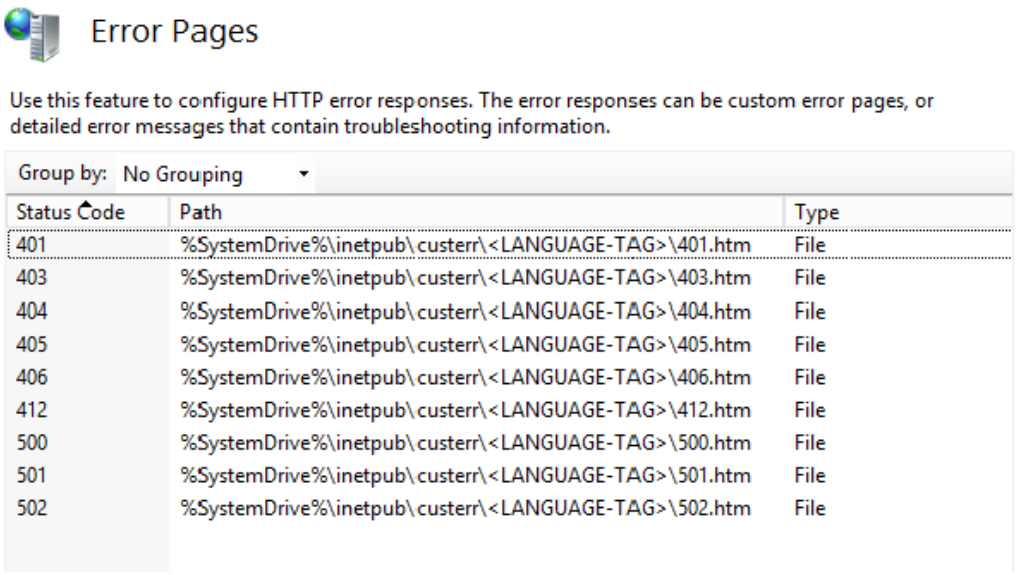

Figura 32. Configuració de les respostes d'error en IIS 8.5

Configuració de la capçalera HTTP a IIS

IIS proporciona una secció a la configuració que permet afegir capçaleres "HTTP Response Headers", però no permet modificar la capçalera Servidor que per defecte dóna informació sobre la seva versió. En versions anteriors a la 8.5 s'ha de crear un mòdul i configurar la seva execució per poder modificar aquesta capçalera. No hi ha documentació sobre com fer-ho en IIS 8.5.

Comportament d'altres versions de IIS

S'ha analitzat també el comportament de l'error 404 en versions prèvies del servidor.

El primer cas és un IIS 6 sobre un Windows 2003 en castellà on obtenim el resultat de la figura 33.

Figura 33. 404 not found en el servidor IIS 6

La pàgina indica que es tracta d'un servidor IIS, però no aporta més informació.

L'error 404 mitjançant Telnet proporciona la següent resposta, on en la capçalera HTTP obtenim la versió de IIS: Microsoft-IIS/6.0

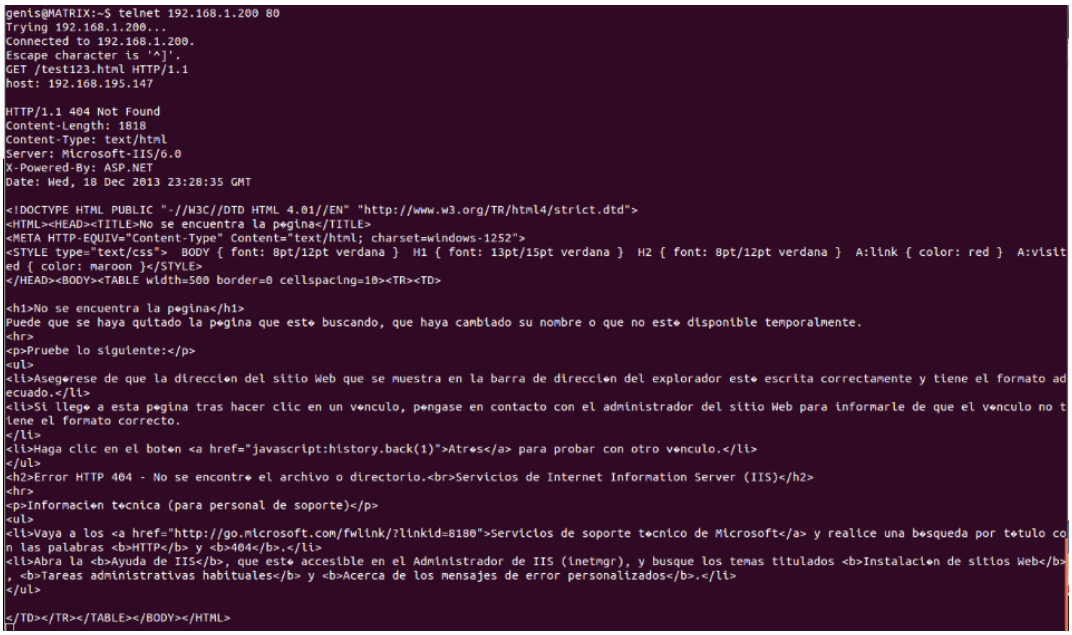

Figura 34. Resposta HTTP al error 404 en el servidor IIS 6

Si fem les mateixes comprovacions sobre un IIS 7.5 en un Windows 2008 obtenim uns resultats semblants. La resposta HTML no proporciona informació sobre la versió ni que es tracta d'un servidor IIS, tot i que el format de la pàgina indica que ho és. La capçalera HTTP sí que indica que es tracta de la versió 7.5.

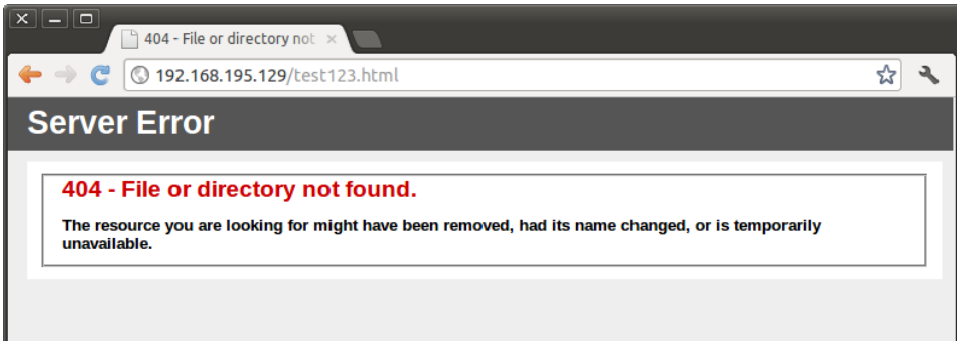

Figura 35. 404 not found en el servidor IIS 7.5

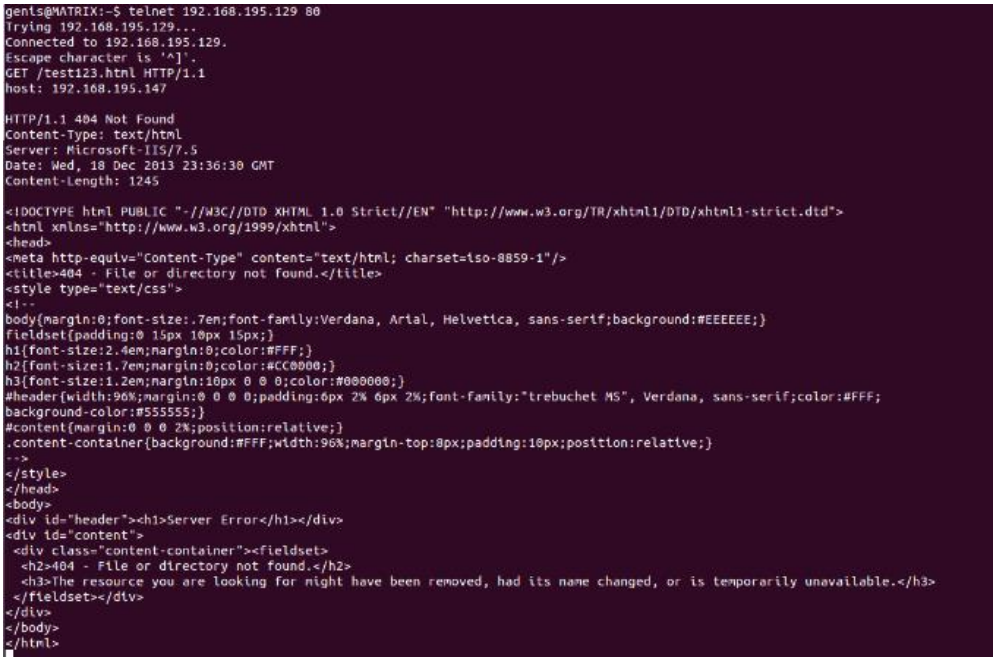

Figura 36. Resposta HTTP al error 404 en el servidor IIS 7.5

Conclusions de l'anàlisi

Podem concloure que IIS per defecte no mostra cap tipus d'informació sensible en els errors a nivell de servidor web, i permet modificar fàcilment les pàgines d'error.

IIS sí que mostra la versió del servidor en les capçaleres HTTP i no és senzill per l'administrador del servei modificar aquest comportament.

Una possibilitat per realitzar la identificació de la versió de servidor seria fer un *fingerprint* del codi retornat, és a dir, coneixent les diferències entre les respostes per cada versió podríem comparar aquestes amb la resposta obtinguda i així identificar si es tracta d'un servidor IIS 7.5 o IIS 8.5. Aquesta tècnica no es vàlida en aquest cas ja que les dues versions retornen exactament el mateix codi HTML per defecte com a resposta del error.

3.2.2. L'error 404 en Apache

Apache és el servidor web més popular. Està disponible tant per Windows com per Linux i és habitual el seu ús quan es volen mostrar pàgines estàtiques sense programació o pàgines en PHP. El nostre laboratori disposa de la versió 2.2.22 d'Apache sobre un Linux Ubuntu Server 12.04.

Apache per defecte mostra la versió del servidor en el codi HTML de la pàgina 404.

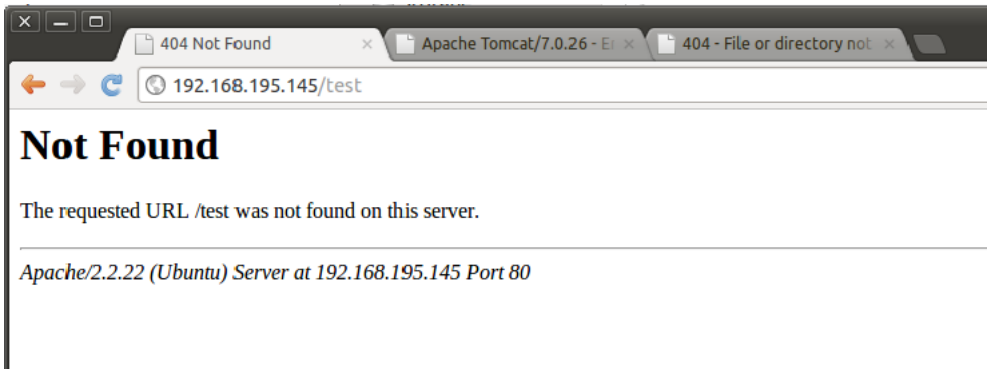

Figura 37. 404 not found per defecte en el servidor Apache

Si realitzem una petició HTTP mitjançant Telnet obtenim la següent informació:

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Mon, 23 Dec 2013 23:01:05 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 291
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /test123.html was not found on this server.</p>
<hr>
<address>Apache/2.2.22 (Ubuntu) Server at 192.168.195.145 Port 80</address>
</body></html>
```

Figura 38. Resposta HTTP al error 404 en el servidor Apache

En la petició HTTP a través de Telnet observem que el HTML conté la versió del servidor, però també la capçalera HTTP ens està donant exactament la mateixa informació. En les peticions satisfactòries també podem obtenir la versió del servidor en el camp Server de la capçalera HTTP.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET / HTTP/1.1
host: 192.168.195.145

HTTP/1.1 200 OK
Date: Mon, 23 Dec 2013 23:05:19 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Sat, 30 Nov 2013 13:23:00 GMT
ETag: "a1fec-b1-4ec64d884e138"
Accept-Ranges: bytes
Content-Length: 177
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Figura 39. Resposta HTTP satisfactòria 200 en el servidor Apache

Configuració del error 404 a Apache

Hi ha tres mètodes de configurar Apache per tal que no mostri informació sensible en els missatges d'error.

Mètode 1) Mitjançant les directives *ErrorDocument*

Les directives *ErrorDocument* permeten realitzar una personalització de les respostes d'error HTTP. Permeten escriure un missatge o redirigir a l'usuari a una pàgina d'error específica:

ErrorDocument 404 "La pàgina no ha estat trobada"
ErrorDocument 404 /404.html
ErrorDocument 404 http://paginaexterna.com/error.html

De la mateixa manera que configurem les respostes del error 404, podem configurar les respostes d'altres codis HTTP: 401, 403 o 500. Aquestes directives es configuren en l'arxiu apache2.conf i es poden aplicar a nivell de servidor, *virtual host* o directori.

He realitzat la configuració del error 404 en l'arxiu apache2.conf per el servidor global de tal manera que respondrà amb un missatge d'error personalitzat enlloc del missatge que mostra Apache per defecte.

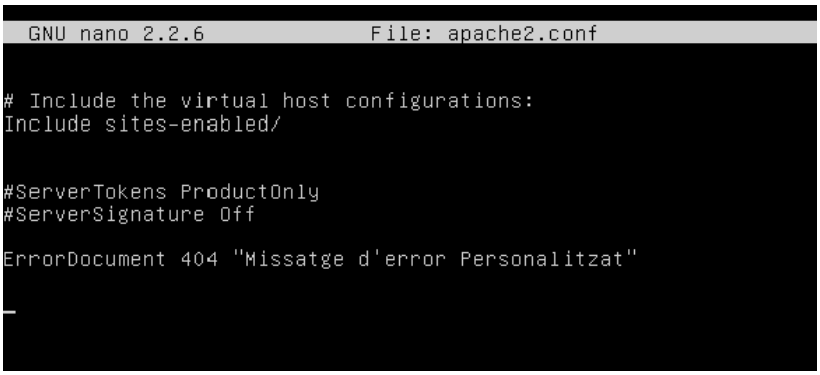

Figura 40. Configuració de la directiva ErrorDocument amb un missatge personalitzat al arxiu apache2.conf

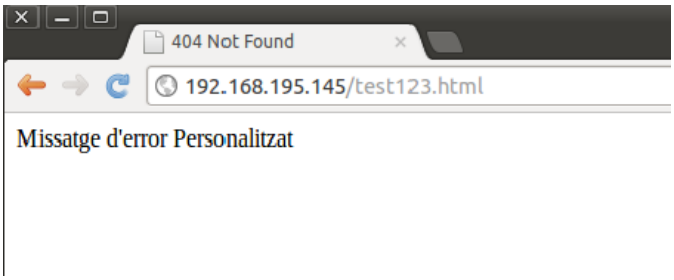

Figura 41. Missatge de resposta personalitzat segons la configuració anterior

En la petició HTTP a través de Telnet podem observar que el missatge d'error HTML que ens retorna no mostra la versió del servidor, però sí ho continua fent la capçalera HTTP.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Wed, 25 Dec 2013 12:12:18 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 30
Content-Type: text/html; charset=iso-8859-1

Missatge d'error Personalitzat
```

Figura 42. Missatge de resposta a través de telnet

Mètode 2) Utilitzant l'arxiu .htaccess

Quan no es té accés a la configuració d'Apache (per exemple en un entorn compartit) es disposa del arxiu *.htaccess* que permet configurar les directives d'Apache per a una carpeta específica.

Aquestes directives permeten realitzar la mateixa configuració que en l'apartat anterior.

Per poder utilitzar *.htaccess* i poder configurar la directiva *ErrorDocument*, Apache ha de tenir configurada la directiva *AllowOverride* en el directori de treball per permetre al seu administrador sobreescriure les directives d'Apache per el seu lloc web. *AllowOverride* es pot configurar per permetre modificar totes les directives, un grup determinat de directives o no permetre modificar les directives.

En el nostre laboratori hem realitzat la configuració *d'AllowOverride* a apache2.conf (figura 43) per tal que es pugui configurar el directori "test404" amb el conjunt de directives "FileInfo", que inclou *ErrorDocument*. Hem configurat l'arxiu *.htaccess* (figura 44) per tal que redirigeixi les pàgines 404 al arxiu 404.html (figures 45 i 46).

```
  GNU nano 2.2.6            File: apache2.conf


# Include the virtual host configurations:
Include sites-enabled/


#ServerTokens ProductOnly
#ServerSignature Off

<Directory /var/www/test404>
 AllowOverride FileInfo
</Directory>
-
```

Figura 43. Configuració d'apache2.conf per permetre al administrador del *site* modificar la directiva *ErrorDocument*

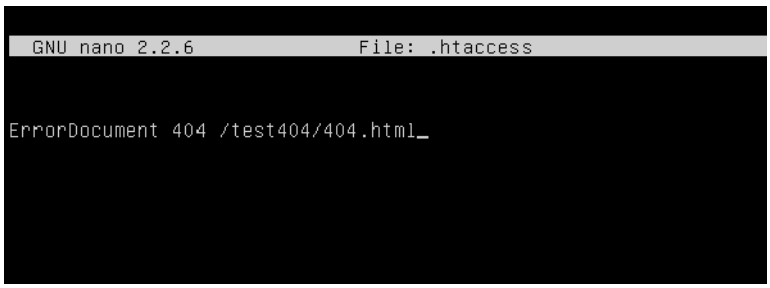

Figura 44. Configuració .htaccess configurant la directiva *ErrorDocument* per mostrar la resposta d'error personalitzada 404.html

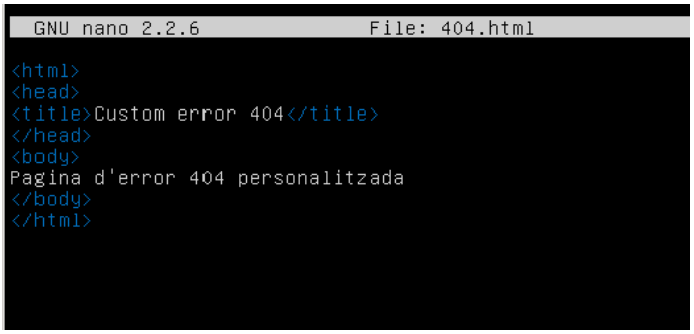

Figura 45. Pàgina d'error 404.html personalitzada

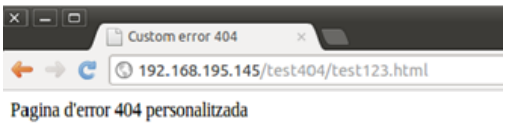

Figura 46. Resposta 404 Not Found mostrant la pàgina 404.html personalitzada

Mètode 3) Directives ServerSignature i ServerTokens

Finalment, Apache també proporciona dues directives addicionals configurables en l'apache2.conf i que permeten escollir quina és la informació que el servidor ha de mostrar respecte la versió d'Apache.

La primera directiva és *ServerSignature* que es pot configurar en *On* o *Off*. Aquesta directiva defineix el comportament del servidor en quant a mostrar la versió del *software* en diferents apartats del servidor, entre ells, el missatge d'error per defecte. Quan està *Off*, el servidor no mostrarà cap informació. *On* és la configuració per defecte i mostrarà informació sobre la versió del servidor.

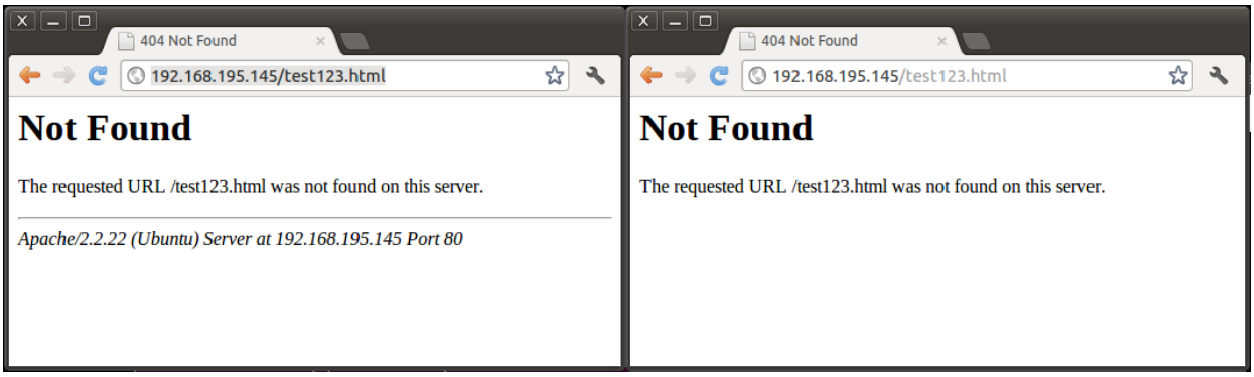

Figura 47. Diferència de la resposta entre *ServerSignature On* i *Off*

Com podem observar en la figura 48, *ServerSignature Off* no modifica el comportament de les capçaleres HTTP quan es realitza una petició a través de Telnet. Continuem veient la versió del servidor.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Wed, 25 Dec 2013 12:26:44 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 210
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /test123.html was not found on this server.</p>
</body></html>
```

Figura 48. Resposta HTTP al error 404 en el servidor Apache amb la directiva *ServerSignature Off*. La resposta HTML no proporciona la versió.

La segona directiva que podem configurar en l'arxiu apache2.conf és *ServerTokens*. Aquesta directiva modifica la forma en que el servidor mostra la informació de la seva versió. Per defecte, Apache mostrarà la configuració *Full*, però les opcions disponibles són:

- ProductOnly – Server: Apache
- Major – Server: Apache/2
- Minor – Server: Apache/2.2
- Minimal – Server: Apache/2.2.22
- OS – Server: Apache/2.2.22 (Ubuntu)
- Full – Server: Apache/2.2.22 (Ubuntu) PHP/4.2.1

Per poder demostrar el comportament de l'opció Full respecte l'opció OS, hem instal·lat PHP5 en el servidor Ubuntu.

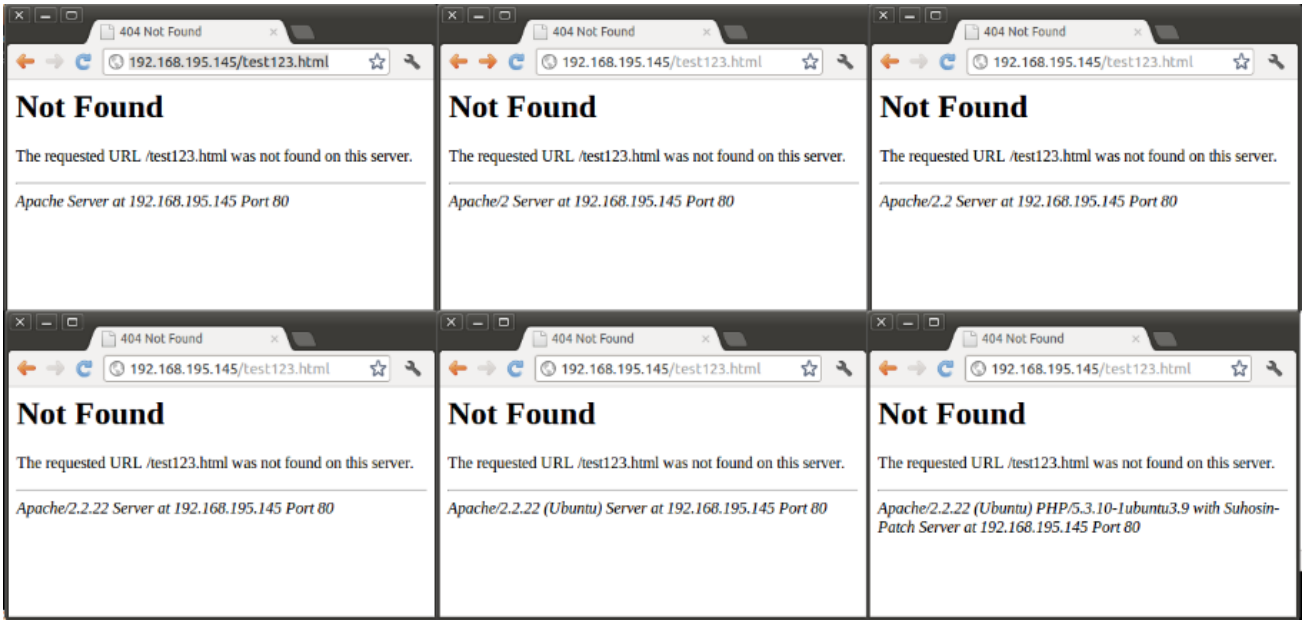

Figura 49. Diferència de la resposta entre ServerTokens a ProductOnly, Major, Minor, Minimal, OS i Full

Aquesta configuració modifica el comportament de la capçalera Server en HTTP. *ServerTokens* ens permet modificar la informació que rebrem en la capçalera HTTP tant en els missatges d'error com en les peticions satisfactòries.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Wed, 25 Dec 2013 12:49:52 GMT
Server: Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch
Vary: Accept-Encoding
Content-Length: 332
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /test123.html was not found on this server.</p>
<hr>
<address>Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 192.168.195.145 Port 80</
address>
</body></html>
```

Figura 50. HTTP mostra la informació completa tant en la resposta d'error 404 com en les capçaleres HTTP quan *ServerTokens* és *Full*

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Wed, 25 Dec 2013 12:51:01 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 275
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /test123.html was not found on this server.</p>
<hr>
<address>Apache Server at 192.168.195.145 Port 80</address>
</body></html>
```

Figura 51. HTTP mostra només "Apache" en les capçaleres HTTP quan *ServerTokens* és *ProductOnly*

Configuració de la capçalera HTTP a Apache

Podem modificar la capçalera HTTP en Apache mitjançant el mòdul addicional *mod_security*. Un cop instal·lat aquest mòdul, podem afegir una nova directiva dins de l'arxiu de configuració apache2.conf anomenada *SecServerSignature* seguida d'un text. Aquest text és el que es mostrarà en la capçalera server de la resposta HTTP.

En l'exemple de la figura 52 s'ha configurat la directiva SecServerSignature amb el valor ServidorMISTIC.

```
genis@MATRIX:~$ telnet 192.168.195.145 80
Trying 192.168.195.145...
Connected to 192.168.195.145.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.145

HTTP/1.1 404 Not Found
Date: Wed, 25 Dec 2013 13:08:17 GMT
Server: ServidorMISTIC
Vary: Accept-Encoding
Content-Length: 283
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /test123.html was not found on this server.</p>
<hr>
<address>ServidorMISTIC Server at 192.168.195.145 Port 80</address>
</body></html>
```

Figura 52. Resposta HTTP a l'error 404 amb la capçalera Server personalitzada.

Conclusions de l'anàlisi

Apache proporciona eines per poder ocultar la informació sensible dels errors HTTP, tant de manera global com de manera especifica. Tot i això, per defecte mostra tota la informació de versions i mòduls instal·lats de que disposa.

La millora respecte IIS és que Apache permet modificar la informació que proporcionen les capçaleres HTTP i també permet configurar de manera global la informació de versió que proporciona el servidor.

3.2.3. L'error 404 en Tomcat

Tomcat és un servidor web molt utilitzat quan es vol realitzar una pàgina web en Java i JSP. Està disponible en Windows i Linux però és més habitual torbar-lo en entorns Linux.

En el nostre laboratori hem instal·lat la versió 7.0.26 de Tomcat sobre un Linux Ubuntu 12.04

Figura 53. Pàgina de resposta per defecte per l'error 404 en Tomcat

Observem que Tomcat, per defecte, mostra informació sobre la versió instal·lada en el servidor en la resposta d'error HTTP 404. Tomcat s'instal·la per defecte en el port 8080, per tant haurem d'utilitzar aquest port en el Telnet.

```
genis@MATRIX:~$ telnet 192.168.195.148 8080
Trying 192.168.195.148...
Connected to 192.168.195.148.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.148

HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 991
Date: Wed, 25 Dec 2013 18:51:19 GMT

<html><head><title>Apache Tomcat/7.0.26 - Error report</title><style><!--H1 {font-family:Tah
oma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-family:T
ahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-family
:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} BODY {font-fa
mily:Tahoma,Arial,sans-serif;color:black;background-color:white;} B {font-family:Tahoma,Aria
l,sans-serif;color:white;background-color:#525D76;} P {font-family:Tahoma,Arial,sans-serif;b
ackground:white;color:black;font-size:12px;}A {color : black;}A.name {color : black;}HR {col
or : #525D76;}--></style> </head><body><h1>HTTP Status 404 - /test123.html</h1><HR size="1"
noshade="noshade"><p><b>type</b> Status report</p><p><b>message</b> <u>/test123.html</u></p>
<p><b>description</b> <u>The requested resource (/test123.html) is not available.</u></p><HR
 size="1" noshade="noshade"><h3>Apache Tomcat/7.0.26</h3></body></html>
```

Figura 54. Resposta HTTP al error 404 en el servidor Tomcat

Podem observar que en el codi HTML i en el títol de la pàgina d'error es mostra la versió del servidor "Apache Tomcat/7.0.26". Per altra banda, la capçalera HTTP proporciona el nom del servidor Apache-Coyote/1.1.

En una resposta correcta "200" obtenim també la capçalera Server que ens indica que el servidor és Apache-Coyote 1.1.

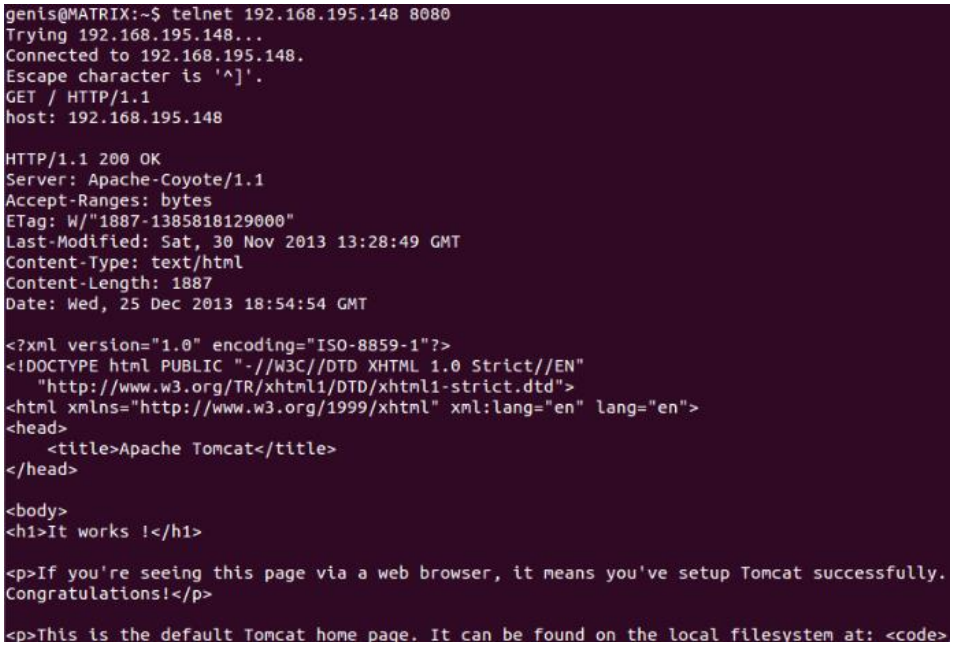

Figura 55. Resposta HTTP 200 satisfactòria en el servidor Tomcat

Apache-Coyote és el connector de Tomcat que s'utilitza com a interfície de servei web i gestiona les peticions HTTP. Des de la versió 4 de Tomcat s'està utilitzant Apache-Coyote 1.1 però, més que indicar la versió del connector, el 1.1 indica que es tracta d'un connector per la versió 1.1 del protocol HTTP.

Tomcat no proporciona informació sobre la versió instal·lada en la capçalera HTTP, però sí indica que es tracta d'un servidor Tomcat.

Configuració del error 404 a Tomcat

Tomcat permet personalitzar els missatges de resposta HTTP mitjançant l'arxiu web.xml (figura 56), on s'ha de configurar l'etiqueta <error-page> amb el codi d'error i la pàgina web de resposta que es vol proporcionar.

Mitjançant aquesta configuració, estem redirigint els missatges d'error 404 a l'arxiu 404.html (figura 57) que hem creat a l'arrel del servidor, evitant que el servidor proporcioni la seva versió.

```
 GNU nano 2.2.6                File: web.xml

<!-- If no welcome files are present, the default servlet either serves a -->
<!-- directory listing (see default servlet configuration on how to       -->
<!-- customize) or returns a 404 status, depending on the value of the    -->
<!-- listings setting.                                                    -->
<!--                                                                      -->
<!-- If you define welcome files in your own application's web.xml        -->
<!-- deployment descriptor, that list *replaces* the list configured      -->
<!-- here, so be sure to include any of the default values that you wish  -->
<!-- to use within your application.                                       -->

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/404.html</location>
    </error-page>
```

Figura 56. Configuració del arxiu web.xml per personalitzar l'error 404 apuntant-lo a /404.html

```
 GNU nano 2.2.6                File: 404.html

<html>
<head>
<title>Custom error 404</title>
</head>
<body>
Pagina personalitzada 404
</body>
</html>
```

Figura 57. Pàgina d'error /404.html personalitzada

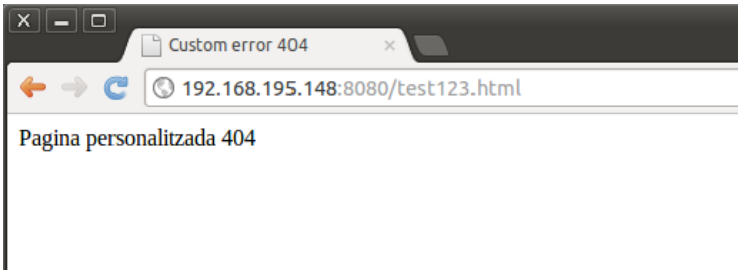

Figura 58. Resposta al error 404 on veiem que s'està utilitzant la pàgina 404.html personalitzada

```
genis@MATRIX:~$ telnet 192.168.195.148 8080
Trying 192.168.195.148...
Connected to 192.168.195.148.
Escape character is '^]'.
GET /index123.html HTTP/1.1
host: 192.168.195.148

HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 103
Date: Wed, 25 Dec 2013 19:39:04 GMT

<html>
<head>
<title>Custom error 404</title>
</head>
<body>
Pagina personalitzada 404
</body>
</html>
```

Figura 59. Resposta HTTP al error 404 personalitzat

Aquest mateix grup d'etiquetes <error-page> de Tomcat es pot utilitzar per altres codis d'error HTTP, i també es pot utilitzar per mostrar pàgines d'error quan es mostren excepcions en la capa d'aplicació mitjançant la subetiqueta <exception-type>.

Configuració de la capçalera HTTP a Tomcat

Tomcat permet modificar la capçalera de resposta HTTP amb un valor personalitzat del camp Server. Per fer-ho, simplement s'ha de modificar a l'arxiu server.xml l'etiqueta *Connector* i afegir-hi el camp server="Valor", de la manera que es mostra a la següent imatge. Els canvis s'aplicaran després de reiniciar Tomcat.

```
 GNU nano 2.2.6                File: server.xml

    -->


    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               URIEncoding="UTF-8"
               redirectPort="8443"
                server="ServidorMISTIC" />
```

Figura 60. Configuració del arxiu server.xml per modificar el camp Server de la capçalera HTTP

```
genis@MATRIX:~$ telnet 192.168.195.148 8080
Trying 192.168.195.148...
Connected to 192.168.195.148.
Escape character is '^]'.
GET /index123.html HTTP/1.1
host: 192.168.195.148

HTTP/1.1 404 Not Found
Content-Type: text/html
Content-Length: 103
Date: Thu, 26 Dec 2013 09:30:22 GMT
Server: ServidorMISTIC

<html>
<head>
<title>Custom error 404</title>
</head>
<body>
Pagina personalitzada 404
</body>
</html>
```

Figura 61. Resposta HTTP al error 404 amb la capçalera Server modificada amb el valor ServidorMISTIC

Comportament d'altres versions de Tomcat

S'ha instal·lat la versió 6.0.32 de Tomcat en un Ubuntu Desktop 11.10 per comprovar que el comportament d'aquesta versió de Tomcat és idèntic al de la versió 7.

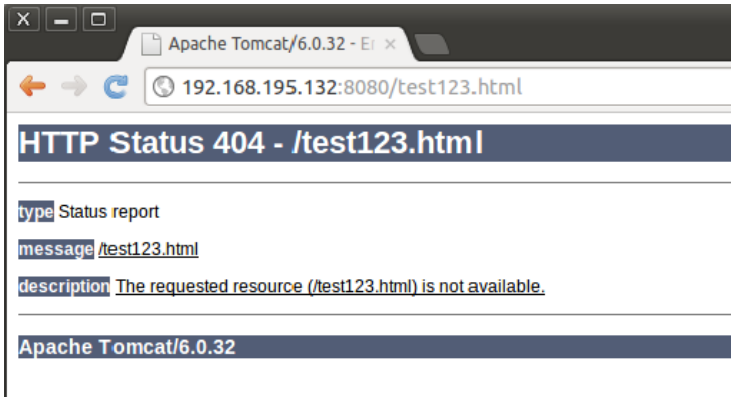

Figura 62. Resposta al error 404 en el servidor Tomcat 6.0.32

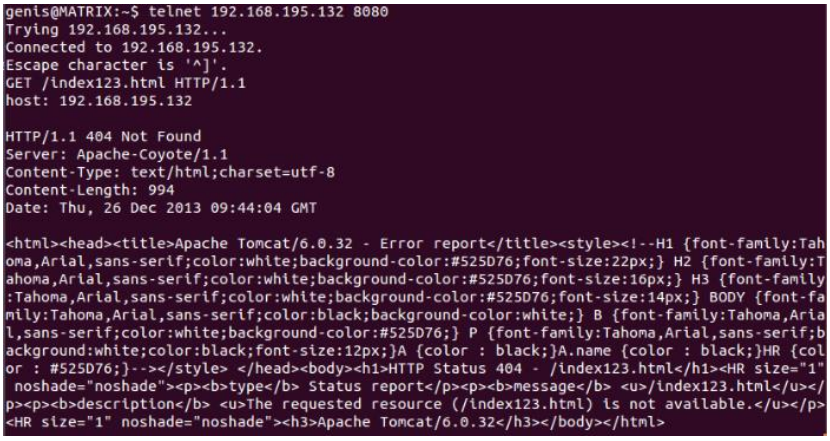

Figura 63. Resposta HTTP al error 404 en el servidor Tomcat 6.0.32

També s'ha instal·lat la versió 4.1.40 de Tomcat sobre un Windows 2008 i s'ha pogut comprovar que, malgrat la pàgina de resposta té un estil diferent, el comportament és el mateix.

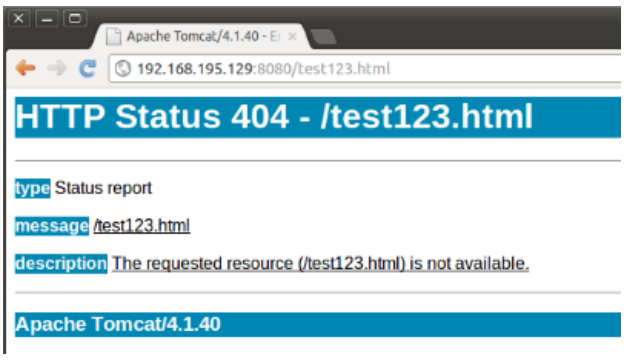

Figura 64. Resposta al error 404 en el servidor Tomcat 4.1.40

```
genis@MATRIX:~$ telnet 192.168.195.129 8080
Trying 192.168.195.129...
Connected to 192.168.195.129.
Escape character is '^]'.
GET /test123.html HTTP/1.1
host: 192.168.195.129

HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Transfer-Encoding: chunked
Date: Thu, 26 Dec 2013 11:01:44 GMT

2d4
<html><head><title>Apache Tomcat/4.1.40 - Error report</title><STYLE><!--H1{font-family : sa
ns-serif,Arial,Tahoma;color : white;background-color : #0086b2;} H3{font-family : sans-serif
,Arial,Tahoma;color : white;background-color : #0086b2;} BODY{font-family : sans-serif,Arial
,Tahoma;color : black;background-color : white;} B{color : white;background-color : #0086b2;
} HR{color : #0086b2;} --></STYLE> </head><body><h1>HTTP Status 404 - /test123.html</h1><HR
size="1" noshade="noshade"><p><b>type</b> Status report</p><p><b>message</b> <u>/test123.htm
l</u></p><p><b>description</b> <u>The requested resource (/test123.html) is not available.</
u></p><HR size="1" noshade="noshade"><h3>Apache Tomcat/4.1.40</h3></body></html>
0
```

Figura 65. Resposta HTTP al error 404 en el servidor Tomcat 4.1.40

Conclusions de l'anàlisi

Tomcat compleix les mateixes característiques que el servidor Apache: per defecte mostra la informació de la versió del servidor, però també permet una personalització completa tant dels codis d'error HTTP com de la capçalera server de HTTP.

La dierència respecte IIS i Apache, és que Tomcat no indica per defecte la versió del producte en la capçalera HTTP, sinó que mostra la versió del *Connector* que està utilitzant.

TEMA 4 – Aplicacions d'anàlisi

4.1. Introducció

L'ús i l'anàlisi de la informació que proporcionen els errors per obtenir informació sensible és completament diferent per cadascuna de les capes i, fins i tot, per cadascuna de les aplicacions, sistemes o servidors que es volen analitzar. També hi ha diferents maneres de realitzar aquesta anàlisi. Per aquest motiu, he volgut crear una aplicació base anomenada ErrorMint que sigui fàcilment extensible a diferents errors afegint nous mòduls a mesura que es fa un estudi més exhaustiu de cadascun dels errors.

Addicionalment també he creat el mòdul "httperror" per realitzar l'anàlisi de la informació que s'obté a partir dels errors HTTP. És molt comú configurar un error personalitzat per la pàgina 404, però no s'acostuma a tenir en compte la resta d'errors. Aquest mòdul realitzarà un conjunt de peticions HTTP per obtenir diferents codis d'error i generar un informe de quina informació s'obté dels errors.

4.2. Aplicació Base ErrorMint

ErrorMint s'encarrega de generar un projecte, definir els servidors als que es faran les proves i escollir els mòduls que es volen executar. S'ha realitzat el codi de tal manera que es poden afegir mòduls addicionals sense haver de modificar el codi de l'aplicació base.

4.2.1. Funcionalitats

Les funcionalitats detallades de l'aplicació són:

1- Per als usuaris de l'aplicació:ç
   - Definició de projecte
     - Si ja existeix, permet eliminar l'existent
   - Definició de servidors objectius
     - A partir d'un arxiu
     - Manualment
   - Definició de mòduls a utilitzar
     - Mostra llistat de mòduls instal·lats i permet escollir quins es volen utilitzar
   - Creació dels arxius del projecte
     - Crea una carpeta per cada un dels objectius
     - Crea l'arxiu "project summary" amb el resum de la configuració del projecte

2- Per als desenvolupadors de l'aplicació:
   - Gestió de mòduls
     - Permet afegir mòduls sense modificar el codi de l'aplicació principal
   - Gestió de connexions
     - Classe amb mètodes per connectar amb el servidor de diferents maneres
   - Utils
     - Classe amb mètodes amb funcionalitats genèriques

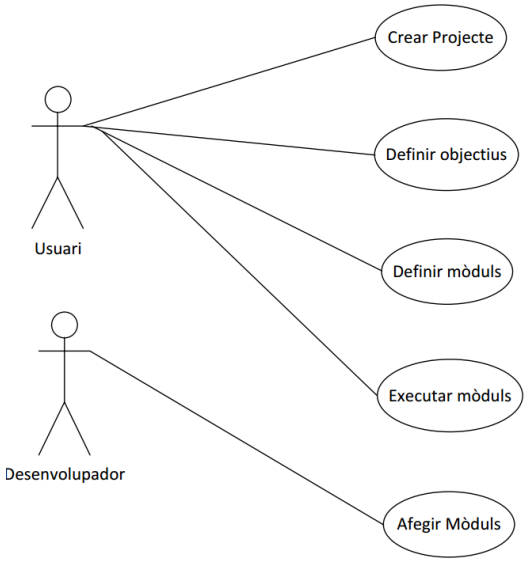

Figura 66. Diagrama de casos d'ús

4.2.2. Codi, arxius i mètodes

L'aplicació base està formada per tres classes: *controller*, *http* i *utils* i dos arxius de text: servers.txt i modules.txt

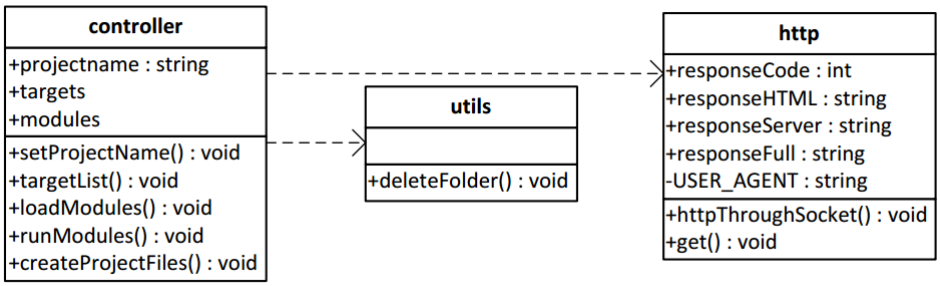

Figura 67. Diagrama de classes de l'aplicació principal

controller

La classe *controller* conté el fluxe de l'aplicació i els mètodes directament relacionats amb ella.

1) void setProjectName

Demana a l'usuari el nom del projecte. Es comprova si el nom ja existeix i, si és així, es dóna la possibilitat d'eliminar el projecte existent.

2) void targetList

Demana a l'usuari com vol introduir els servidors que vol utilitzar com a objectiu. Es pot escollir entre 1) obtenir-los des del arxiu servers.txt, o 2) es poden introduir manualment.

3) void loadModules

El mètode obté el llistat de mòduls actius de l'arxiu modules.txt i mostra aquest llistat a l'usuari. L'usuari pot escollir els mòduls que vol executar.

4) void runModules

S'utilitza la funcionalitat *reflection* de Java per poder cridar dinàmicament a les classes que contenen els mòduls segons la informació obtinguda del arxiu modules.txt. Aquesta és la funcionalitat que permet afegir mòduls sense necessitar modificar el codi font. Aquest mètode executarà el mètode run() de la classe amb el nom obtingut del arxiu modules.txt.

5) void createProjectFiles

Crea la carpeta del projecte segons la informació proporcionada per l'usuari. Dins de la carpeta del projecte crearà una carpeta per cadascun dels servidors objectius per tal que els mòduls hi guardin la informació específica de cada servidor. També crea un arxiu project_summary.txt amb el resum de la configuració del projecte.

6) main()

No realitza cap operació, simplement crida als mètodes anteriors en l'ordre apropiat.

http

La classe *http* conté mètodes que poden utilitzar els diferents mòduls per fer peticions HTTP als servidors i obtenir les respostes.

1) void get(String url)

Realitza una petició GET estàndard a la URL que es passa per paràmetre. Utilitza l'objecte HttpURLConnection de Java per realitzar la connexió.

2) void httpThroguhSocket(String server, int port, String request,int delay)

Realitza una petició HTTP a través de l'obertura d'un *socket* contra el servidor i el port que s'indiquen per paràmetre. Això permet realitzar peticions HTTP avançades i en altres formats no estàndards, mitjançant el paràmetre *request*. També permet provocar *time-outs* mitjançant el paràmetre *delay*. Utilitza un *socket* per realitzar la connexió el que permet molta més flexibilitat.

utils

L'objectiu de la classe *utils* és proporcionar mètodes per poder realitzar operacions genèriques des dels mòduls. Per ara només conté un mètode:

1) void deleteFolder(File folder)

Esborra la carpeta, arxius i subcarpetes de la ruta que es passi per paràmetre.

servers.txt

Conté el llistat de servidors (IPs o dominis) sobre els que es volen executar els mòduls.

modules.txt

Conté el llistat de mòduls i la descripció disponibles en l'aplicació.

4.2.3. Us

Funcionament bàsic

Un cop s'executa el programa ErrorMint es demana el nom del projecte a l'usuari. Si ja existeix la carpeta amb el nom del projecte, demana a l'usuari si vol eliminar la carpeta existent. Si es selecciona "No", es demanarà afegir un nou nom de projecte.

```
C:\Users\genis\Desktop\ErrorMint>java controller
Project name:

Labo
Project already exists, do you want to remove the existant projecct? [Yes] or [No]
No
Project name:

Labo2
```

Figura 68. Execució inicial del programa, selecció del nom "Labo", el programa avisa que ja existeix, selecció del nom del projecte Labo2

Després es demana d'on es volen obtenir els objectius de les proves.

```
Choose method to enter the targets
1) get targets from servers.txt
2) set targets manually
2
Insert the domains one by one. Insert 0 to finish adding domains
192.168.195.145
192.168.195.147
192.168.195.148
0
```

Figura 69. Selecció dels servidors objectius de les proves. S'han afegit tres servidors manualment.

El següent pas serà definir quins mòduls es volen utilitzar.

```
This is the list of current modules. Choose the modules you want to use. Insert 0 to finish adding module
s
1) httperror - HTTP errors analysys
2) module2 - Future Module 2
3) module3 - Future Module 3
4) module4 - Future Module 4
1
Module httperror selected, insert 0 to finish or insert another module number
0
```

Figura 70. Selecció del mòdul httperror. Els altres tres mòduls es mostren només com a exemple.

A partir d'aquest punt s'executaran els mòduls un a un i el fluxe dependrà de cadascun dels mòduls.

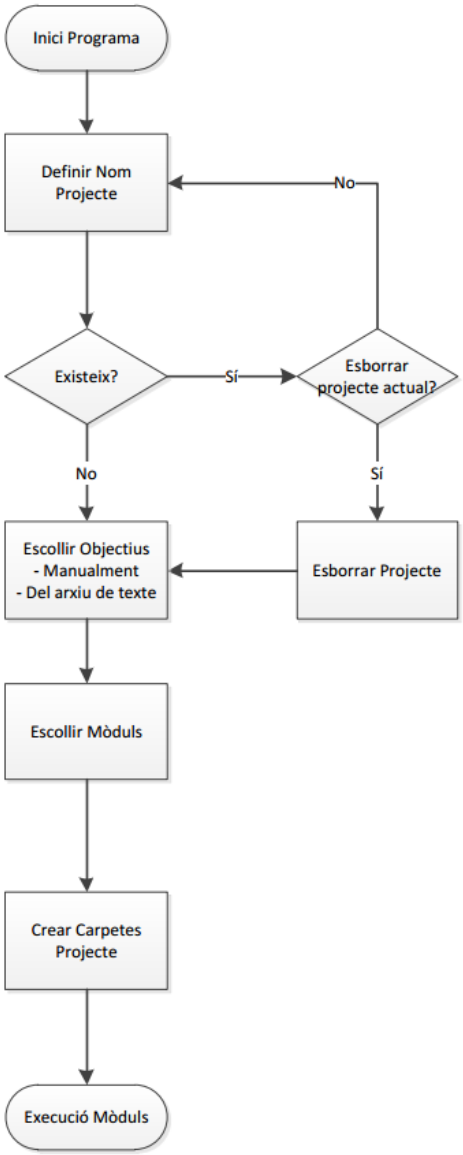

Figura 71. Diagrama de flux de l'aplicació ErrorMint

Instruccions per afegir un nou mòdul

Per afegir un nou mòdul al programa no és necessari tocar el codi font. S'ha de realitzar la següent configuració:

- Afegir una entrada nova en l'arxiu modules.txt, respectant les dues columnes: nom del mòdul i descripció, separat per una ",". El nom del mòdul ha de correspondre amb el nom de la classe que conté el codi. Aquesta informació es mostrarà a l'usuari quan hagi d'escollir el mòdul.

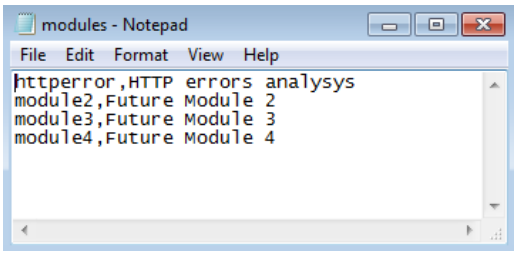

Figura 72. Contingut del arxiu modules

- La classe del mòdul ha de tenir un mètode Run() sense paràmetres.

- El mòdul pot obtenir informació sobre el projecte i objectius a partir de les variables públiques de les classes http i controller:

    - String projectname = new String();
    - Vector<String> targets = new Vector<String>();
    - Vector<String> modules = new Vector<String>();
    - responseCode
    - String responseHTML = new String();
    - String responseServer = new String();
    - String responsefull = new String();

4.2.4. Millores a realitzar

Hi ha certes millores en el programa que es poden realitzar, tot i que queden fora dels objectius d'aquest treball.

S'ha d'estudiar les implicacions de la utilització de refractor de Java per afegir mòduls i analitzar com la modificació de l'execució del programa en temps real podria causar problemes de seguretat a l'usuari que l'executi.

S'ha de millorar també l'accés directe a les variables de les classes del programa. Es pot fer que siguin privades i que s'hagi d'accedir a través de Get's i Set's.

Les classes *utils* i *http* es poden ampliar amb nous mètodes que es puguin utilitzar altres mòduls. Per exemple, es pot desenvolupar un mètode HTTPS per tal de realitzar peticions HTTP a través d'aquest protocol.

4.3. Mòdul aplicació "httperror"

El primer mòdul que s'ha desenvolupat per l'aplicació ErrorMint és el "httperror". Aquest mòdul executa un seguit de peticions HTTP utilitzant diferents formats i mètodes per generar errors i obtenir les respostes del servidor objectiu a cadascun d'ells. En aquest punt fa una cerca en la resposta del servidor per comprovar si està mostrant o no informació sobre la versió del servidor.

Aprofitant que es realitzen les peticions HTTP, "httperror" també realitza un llistat del camp *Server* que els servidors proporcionen en les capçaleres HTTP de les respostes. Aquesta informació no es proporciona directament quan es produeix un error però sí està relacionada, ja que sovint els servidors proporcionen la mateixa informació en la capçalera HTTP que en el missatge d'error.

Durant la realització del programa s'ha pogut comprovar que alguns servidors retornen capçaleres diferents quan la petició és correcta i quan la petició és errònia.

Cada petició té com objectiu obtenir un codi d'error, tot i així l'aplicació és flexible i registra qualsevol codi d'error que s'obtingui, ja que els servidors responen de manera diferent segons el format de les peticions.

4.2.1. Funcionalitats

Les funcionalitats detallades de l'aplicació son:

- Bateria de peticions HTTP contra el servidor:
    - Petició get estàndard de la pàgina d'inici, s'espera un error 200
    - Petició get d'una pàgina aleatòria, s'espera un error 404
    - Petició utilitzant mètodes normalment no permesos o no vàlids: DELETE, TRACE i RENAMEA, s'esperen els errors 405 i 501
    - Petició get mal formada, s'espera l'error 400
    - Petició amb un retard elevat, s'espera l'error 408
    - Petició amb un host incorrecte, s'esperen els codis de redirecció 301 i 302
    - Petició amb una versió HTTP incorrecta, s'espera l'error 505
- Cerca del nom de servidor i versió en les respostes HTML del servidor
- Mitjançant les capçaleres Server HTTP, aprenentatge de nous noms de servidor a cercar en les respostes.
- Generació d'un informe CSV amb el llistat de servidors, codis de resposta obtinguts, versions de servidor obtingudes de les capçaleres HTTP i informació obtinguda de la resposta.
- Emmagatzematge de les respostes als errors en arxius HTML.

4.2.2. Codi, arxius i mètodes

"httperror" consta d'una classe *httperror* i d'un arxiu de text httperror_serverstrings.txt

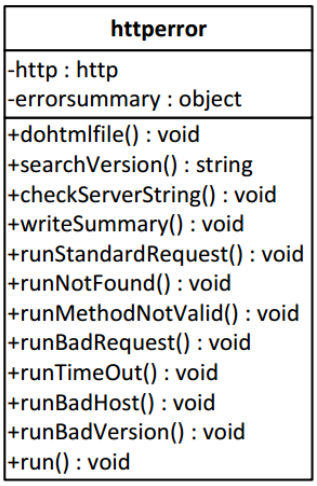

Figura 73. Classe httperror

httperror

La classe "httperror" conté tots els mètodes que executa el mòdul. Hi ha un conjunt de mètodes els quals la seva funcionalitat és realitzar les peticions HTTP esperant un tipus de resposta determinada. Tots ells utilitzant el mètode httpThroguhSocket de la classe *http* d'ErrorMint.

1) void dohtmlfile (String target,String filename)

Crea un arxiu HTML en la carpeta i amb el nom que s'especifiquen com a paràmetre. Obté el codi de la classe *http* d'ErrorMint.

2) String searchVersion()

Fa una cerca en la resposta de l'error de les cadenes que conté l'arxiu httperror_serverstrings.txt. En cas que trobi la cadena en l'arxiu HTML, retornarà un String que contindrà la informació de versió que estigui proporcionant la resposta.

Aquest mètode s'executarà per cada resposta d'error que proporcioni un servidor.

3) void checkServerString(String serverstring)

Compara els valors de l'arxiu httperror_serverstings.txt amb la cadena que se li passa per paràmetre. En cas que la cadena no estigui en l'arxiu, se li pregunta a l'usuari si la vol afegir a l'arxiu. Aquest és el mètode que proporciona la funcionalitat d'autoaprenentatge al mòdul i permet afegir nous noms de servidor a l'aplicació.

Cada cop que es realitzi una petició HTTP es verificarà que el text de la capçalera HTTP server es trobi en l'arxiu.

4) void writeSummary(String target,String comments,String method)

Aquest mètode s'encarrega d'escriure una línia en el informe del mòdul amb la informació que es passa per paràmetre i la informació que pot obtenir de la classe *http*. També crida el mètode searchVersion per fer la cerca de la versió del servidor en la resposta d'error.

5) void runStandardRequest (String target)

Realitza una petició GET normal de l'arrel del servidor (target). S'espera una resposta 200 i és una petició de control. En algunes situacions obtindrem una resposta 301 o 302 si el servidor fa una redirecció a una altra pàgina.

6) void runNotFound(String target)

Realitza una petició GET de la pàgina index95381.html i s'espera una resposta 404.

7) void runMethodNotValid(String target)

Realitza peticions DELETE, TRACE i RENAMEA per obtenir les respostes 405 i 501.

8) void runBadRequest(String target)

Realitza peticions mal formades per obtenir el codi d'error 400.

9) void runTimeOut(String target)

Realitza una petició GET a l'arrel del servidor, però amb un retard de 21 segons entre que s'envia la primera part de la petició i es tanca el *socket*. S'espera rebre el codi d'error 408.

10) void runBadHost(String target)

Realitza una petició GET amb un host que no correspon al destí de la petició. Els servidors responen amb diferents errors.

11) void runBadVersion(String target)

Realitza una petició GET indicant que la versió de HTTP utilitzada és la 1.2 per obtenir un error 505.

12) void run()

És el mètode que cridarà ErrorMint i actua com a "main" del mòdul. Primer pregunta a l'usuari si vol realitzar la prova de *time-out*, després defineix el nom del arxiu on es crearà l'informe i estableix les capçaleres del arxiu. Finalment executa la bateria de peticions HTTP per cadascun dels servidors que s'han configurat en ErrorMint.

4.2.3. Us

Un cop s'ha realitzat la configuració d'ErrorMint, aquest executarà el mòdul "httperror" si ha estat escollit. El primer que demana és si es vol executar o no la petició HTTP de *time-out*. Aquesta petició causa un retard de 21 segons per cada servidor que es vulgui analitzar, per lo que l'usuari pot preferir realitzar les proves sense esperar aquests temps.

```
Do you want to run TimeOut group requests? It takes 21 seconds per server [Yes] [No]
Yes
```

Figura 74. El mòdul demana a l'usuari si vol realitzar les proves de *Time-Out*

El mòdul realitzarà les peticions HTTP per cada servidor i mostrarà quin servidor està analitzant.

```
Running httperror for server 192.168.195.145
Running httperror for server 192.168.195.147
Running httperror for server 192.168.195.148
```

Figura 75. El mòdul informa del progrés de les proves.

Si el programa troba algun nom de servidor que no estigui registrat en l'arxiu httperror_serverstrings.txt, preguntarà al usuari si vol afegir aquest nom al llistat.

```
Located a new server string: Apache-Coyote
Do you want to add this string to the httperror_serverstrings.txt file? [Yes] [No]
Yes
```

Figura 76. S'ha trobat un nou tipus de servidor "Apache-Coyote" i demana a l'usuari si vol afegir-lo o no.

Finalment es generarà l'arxiu *errors-summary* que inclourà tots els missatges d'error obtinguts, les capçaleres Server per cada petició i si s'ha trobat la versió del servidor en la resposta d'error.

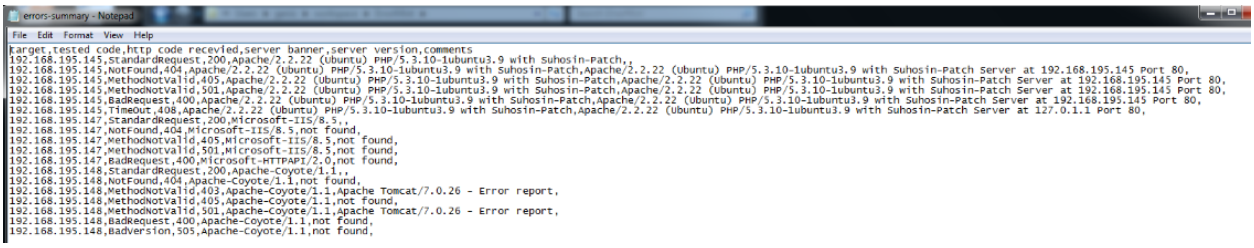

Figura 77. Arxiu *errors-summary* amb el resultat de l'execució de "httperror"

L'arxiu és un *CSV* que es pot obrir amb certs programes per fer-lo més llegible, per exemple, amb l'Excel:

	A	B	C	D	E
1	target	tested code	http code	server banner	server version
2	192.168.195.145	StandardRequest	200	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	
3	192.168.195.145	NotFound	404	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 192.168.195.145 Por
4	192.168.195.145	MethodNotValid	405	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 192.168.195.145 Por
5	192.168.195.145	MethodNotValid	501	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 192.168.195.145 Por
6	192.168.195.145	BadRequest	400	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 192.168.195.145 Por
7	192.168.195.145	TimeOut	408	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch	Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suhosin-Patch Server at 127.0.1.1 Port 80
8	192.168.195.147	StandardRequest	200	Microsoft-IIS/8.5	
9	192.168.195.147	NotFound	404	Microsoft-IIS/8.5	not found
10	192.168.195.147	MethodNotValid	405	Microsoft-IIS/8.5	not found
11	192.168.195.147	MethodNotValid	501	Microsoft-IIS/8.5	not found
12	192.168.195.147	BadRequest	400	Microsoft-HTTPAPI/2.0	not found
13	192.168.195.148	StandardRequest	200	Apache-Coyote/1.1	
14	192.168.195.148	NotFound	404	Apache-Coyote/1.1	not found
15	192.168.195.148	MethodNotValid	403	Apache-Coyote/1.1	Apache Tomcat/7.0.26 - Error report
16	192.168.195.148	MethodNotValid	405	Apache-Coyote/1.1	not found
17	192.168.195.148	MethodNotValid	501	Apache-Coyote/1.1	Apache Tomcat/7.0.26 - Error report
18	192.168.195.148	BadRequest	400	Apache-Coyote/1.1	not found
19	192.168.195.148	BadVersion	505	Apache-Coyote/1.1	not found

Figura 78. *errors-summary* processat per l'Excel per mostrar els resultats més clarament

Cada columna del arxiu conté la següent informació:

- Target – Servidor al que s'han enviat les peticions
- Tested code – Grup de peticions al que correspon el resultat
- HTTP Code – Codi d'error obtingut
- Server Banner – Informació de servidor obtinguda de les capçaleres HTTP
- Server Version – Informació de servidor obtinguda del missatge d'error

També es guarda l'arxiu HTML amb cada resposta a la carpeta del servidor objectiu dins del projecte.

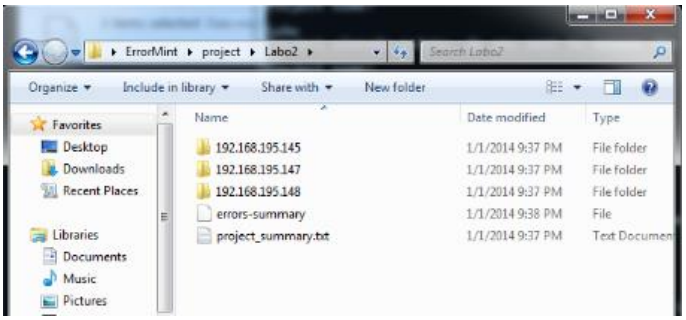

Figura 79. Contingut de la carpeta generada per el projecte, amb una carpeta per servidor i els arxius *errors-summary* i project_summary.txt

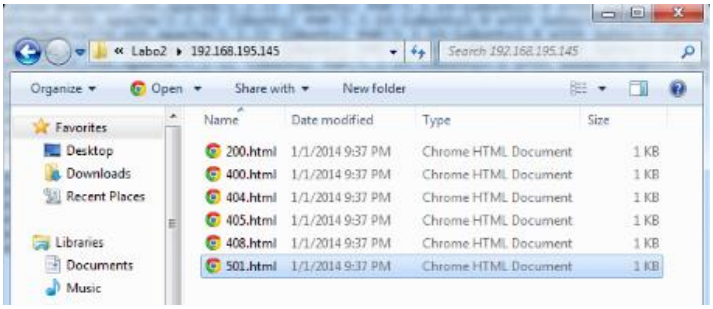

Figura 80. Contingut de la carpeta d'un servidor amb les respostes HTML que ha obtingut per cada error.

Figura81. Arxiu 404.html descarregat per el programa en local i que mostra la resposta d'Apache.

En els resultats obtinguts podem veure que el servidor Apache 192.168.195.145 mostra informació sensible tant en les capçaleres HTTP com en tots els missatges de resposta. El servidor Tomcat 192.168.195.148 només mostra la versió del servidor pels errors 403 i 405, la resta d'errors no mostren detalls de la versió i la capçalera indica que es tracta d'un Tomcat "Apache-Coyote 1.1". Finalment el

servidor Windows 192.168.195.147 només mostra la versió en les capçaleres i no proporciona cap informació en els missatges d'error.

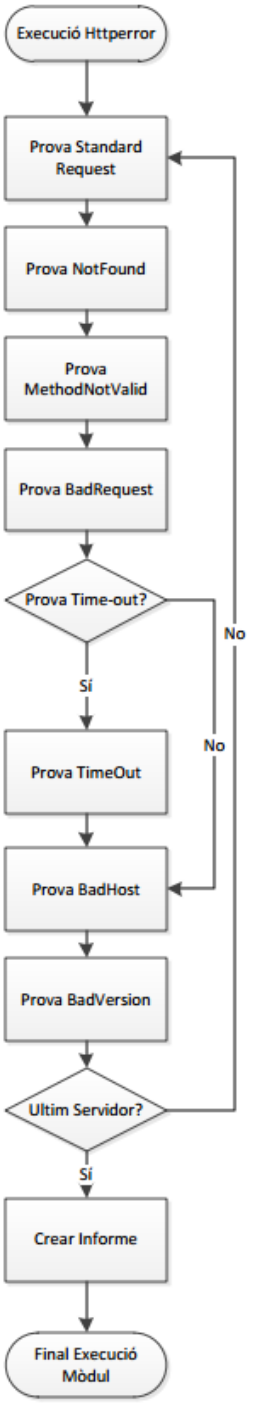

Figura 82. Diagrama de flux del mòdul httperror.

4.2.4. Millores a realitzar

Una millora que es pot realitzar en el mòdul és l'execució concurrent de les proves de manera que no és necessari esperar 21 segons per cada servidor, sinó que es poden llançar totes les peticions de *time-out* alhora.

Per altra banda, es pot ampliar el número de peticions que es realitzen per millorar l'eficàcia de les deteccions i ampliar la quantitat de codis d'error obtinguts.

4.4 Altres mòduls de l'aplicació

Un plantejament que podria fer-se d'un altre mòdul seria l'anàlisi de les pàgines d'error basat en signatures o *fingerprints.* És a dir, es pot comparar la pàgina d'error obtinguda amb les pàgines d'error conegudes de versions específiques de software i d'aquesta manera conèixer quina versió de servidor està instal·lada. És obvi que els estils dels errors que proporcionen Apache, IIS i Tomcat són totalment diferents entre ells, però, encara que no donin informació sobre la versió, podem encertar quin servidor hi ha darrere només veient el disseny. Realitzant un anàlisi de les pàgines en detall, utilitzant resums de les respostes i autoaprenentatge, es podria crear una base de dades que permetés trobar la versió del servidor malgrat no s'indiqui en la pàgina d'error.

Addicionalment es podria comparar la pàgina d'error obtinguda del servidor amb les pàgines d'error conegudes i així poder identificar si són pàgines personalitzades o no.

Una ampliació del plantejament anterior en conjunt dels errors a la capa d'aplicació seria l'anàlisi de les pàgines d'error personalitzades per aplicacions com Wordpress o Drupal.

Conclusions

L'obtenció d'informació sensible a partir dels errors que proporcionen els programes és, tal com indica OWASP dedicant un capítol propi en la seva Testing Guide, un factor important a considerar dins de la seguretat en les aplicacions web, i també és un tema molt ampli, amb moltes branques per cada capa i, fins i tot, per cada aplicació. Els errors en bases de dades i els diferents llenguatges de programació mereixen un estudi dedicat, que segurament es pot dividir per llenguatge i sistema.

Aquest treball es centra en un petit segment d'aquesta matèria, concretament el dels errors HTTP 404 – Not found. Durant la realització d'aquest treball he pogut comprovar que la informació proporcionada per els errors HTTP es limita a la versió del servidor, sistema operatiu i els mòduls que hi ha instal·lat en alguns casos.

Hi ha tres factors que podrien restar importància a la informació proporcionada per els servidors web en els errors.

El primer d'ells és que en l'entorn de la seguretat informàtica es considera que la seguretat per ofuscació no fa un sistema segur i això menysprea la problemàtica de mostrar públicament la versió del servidor. Però això seria només cert si tots els servidors es mantinguessin actualitzats sempre a la última versió. És evident que la realitat no és aquesta i que el primer pas de qualsevol atac consisteix en obtenir informació sobre l'objectiu atacat, per tant, és millor no facilitar la feina a l'atacant proporcionant aquesta informació.

Respecte el segon factor, s'ha pogut comprovar que la informació proporcionada pels errors està molt lligada amb la informació proporcionada per la capçalera HTTP Server en qualsevol petició (no necessàriament errors). Normalment si un error mostra informació sobre la versió del servidor, aquesta informació també apareix en la capçalera HTTP. D'aquesta manera es podria considerar que la problemàtica no és només en els errors HTTP sinó en la configuració general del servidor. Quan es corregeixi la configuració del servidor, solucionarem la problemàtica amb els errors.

Finalment, la informació proporcionada pels errors i les capçaleres HTTP també es pot obtenir amb altres mètodes, com ara el *fingerprinting* del comportament del servidor.

Tot i aquests tres factors, la configuració de les capçaleres HTTP i els missatges d'error és un pas bàsic i és tant senzill de realitzar que és indiscutible el fet que val la pena fer-ho i no assumir els riscos associats a mostrar aquesta informació. Com més complicada es faci la feina a l'atacant, menor serà la probabilitat de patir un atac.

Aquest treball és una petita mostra del estudi que es pot arribar a realitzar dels errors en entorn web i es pot utilitzar com a base per continuar altres estudis sobre errors en PHP, ASP, JSP, CGIs, MySQL, MSSQL, Oracle, Drupal, Wordpress… Es poden realitzar estudis per cadascun dels llenguatges o, fins i tot, estudis d'un error en concret d'un sistema, desenvolupant mòduls que puguin aprofitar aquest error en concret. El fet que en una aplicació web intervinguin móns tan diferents com l'administració del

servidor, la programació de l'aplicació i l'administració de la base de dades dificulta que una única persona pugui gestionar i assegurar tot l'entorn.

Aquest treball m'ha servit per aprendre i aprofundir en diferents conceptes i serveis els quals no estava familiaritzat i que tenia molt interès. He pogut conèixer i llegir sobre la organització OWASP, he fet proves i estudiat com fer "*Hacking* amb cercadors", he instal·lat i configurat diferents tipus de servidors sobre sistemes operatius en entorns virtuals dels quals no tenia experiència: Windows Server 2013, Ubuntu server, IIS, Apache i Tomcat. I finalment també m'ha permès recuperar i posar en pràctica coneixements de programació en Java que no aplicava des de feia molt de temps.

Referències

Informació General

OWASP - https://www.owasp.org
OWASP Testing Guide - https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents
Vincent – Exploiting Information Disclosure - http://www3.nccu.edu.tw/~yuf/SoftwareSecurity/infoexplore.pptx

JBOSS

JBOSS Web - http://www.jboss.org/jbossweb
JBOSS Web change log - http://docs.jboss.org/jbossweb/JBnativechangelog.html
CVE-2013-1976 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1976
CVE-2007-6433 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6433
CVE-2005-2158 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2158

Hacking en Cercadors

Exploit DB – Google Hacking Database - http://www.exploit-db.com/google-dorks/
Tècniques utilitzades d'exemple:
http://www.exploit-db.com/ghdb/91/
http://www.exploit-db.com/ghdb/435/
http://www.exploit-db.com/ghdb/563/
Johnny Long, The Google Hacker's Guide - http://pdf.textfiles.com/security/googlehackers.pdf
Johnny Long, Google Hacking for Penetration Testers, Vol 2. - http://goo.gl/fjrKs0

Altres eines

NMAP - Documentació - http://nmap.org/book/osdetect.html
Shodan - Ajdua - http://www.shodanhq.com/help
FOCA – Presentació Defcon 18 - http://www.securitytube.net/video/1353

HTTP

Wikipedia HTTP - http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
Wikipedia - Llistat de codis HTTP - http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
RFC 1945 – HTTP 1.0 - http://tools.ietf.org/html/rfc1945
RFC 2616 – HTTP 1.1 - http://tools.ietf.org/html/rfc2616
RFC 2775 – HTTP 2.0 - http://tools.ietf.org/html/rfc2774
Jtanium, Peticions HTTPS a través de Telnet - http://www.jtanium.com/2006/08/07/telnet-to-an-https-server
ESQSoft, Peticions HTTP a través de Telnet - http://www.esqsoft.com/examples/troubleshooting-http-using-telnet.htm
W3, Descripció dels mètodes HTTP en el RFC 2616 - http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

APACHE

Configuració de les directives ServerTokens i ServerSignature - http://httpd.apache.org/docs/2.2/mod/core.html#serversignature
http://stackoverflow.com/questions/15130443/remove-server-info-and-php-info-from-response-header
Errors personalitzats - http://httpd.apache.org/docs/current/custom-error.html
Directiva ErrorDocument - http://httpd.apache.org/docs/2.2/en/mod/core.html#errordocument
ModSecurity - http://www.thefanclub.co.za/how-to/how-install-apache2-modsecurity-and-modevasive-ubuntu-1204-lts-server
SecServerSignature - http://superuser.com/questions/286805/how-to-remove-server-header-from-the-http-response-with-apache

TOMCAT

Configuració capçalera HTTP – http://securitythoughts.wordpress.com/2011/03/30/how-to-modify-apache-coyote1-1-banner/
OWASP - Securització de Tomcat - https://www.owasp.org/index.php/Securing_tomcat
404 personalitzats - http://davidghedini.blogspot.com.es/2010/10/tomcat-custom-404-page.html

PHP

PHP – Llistat errors http://www.php.net/manual/en/errorfunc.constants.php

MySQL

MySQL Reference manual – Errors de servidor: http://dev.mysql.com/doc/refman/5.6/en/error-messages-server.html
MySQL Reference manual – Errors de client: http://dev.mysql.com/doc/refman/5.6/en/error-messages-client.html

Annexes

Inventari d'errors

L'arxiu adjunt Error "inventory.xls" al treball conté un inventari dels errors que poden donar les diferents capes que intervenen: servidor, aplicació i bases de dades. En el cas dels errors de servidors, hi ha 59 codis d'error diferents que pot proporcionar HTTP en diferents implementacions. Hi ha 6 d'ells assenyalats en verd i que són els que s'han estudiat en aquest treball.

Les pestanyes de *Application Errors* i *DataBase Errors* inclouen 16 famílies d'errors per PHP i 948 errors de MySQL respectivament. Aquest inventari s'ha d'ampliar amb els errors que podem obtenir d'altres sistemes i llenguatges i fer un anàlisi en cas que es puguin provocar, si proporcionen informació sensible o si es poden fer cerques d'aquests errors utilitzant Google o altres cercadors.

Aplicació ErrorMint

S'inclou la carpeta "ErrorMint – Source" que conté el codi font de les classes *controller*, *util*, *http* i *httperror*. D'altra banda, també s'inclou la carpeta "ErrorMint – Bins" que inclou el codi compilat amb els arxius necessaris per la seva execució. Per executar el codi en un entorn Windows s'ha d'utilitzar la comanda "java controller" tenint en compte la ruta tant de l'aplicació Java com de la classe *controller*.

Missatges de resposta per defecte

La carpeta "Missatges de resposta" inclou les respostes per defecte de cadascun dels servidors que s'han provat en els laboratoris: Apache 2.22, IIS 6,7,8.5 i Tomcat 4,6 i 7.

Proposta publicació OWASP

A partir del estudi realitzat en aquest treball, s'ha proposat una actualització del capítol "*Testing for Error Code (OWASP-IG-006)"* de la "*OWASP Testing Guide".*

La carpeta "OWASP - Testing for Error Code" conté la pàgina original, els canvis proposats i la versió completa amb els canvis proposats d'aquest capítol.

Es pot consultar la pàgina a la següent URL:

https://www.owasp.org/index.php/Testing_for_Error_Code_(OWASP-IG-006)

Publicació a SourceForge

L'aplicació ErrorMint s'ha publicat a la següent URL de SourceForge per fer-la pública. Està disponible tant les classes compilades de Java com el codi font.

http://sourceforge.net/projects/errormint/

Informe ErrorMint projecte UOC

S'ha executat el mòdul "httperror" d'ErrorMint per un conjunt de servidors de la UOC, tal com s'explica en la presentació del treball. Adjunt al treball es troba l'arxiu "Informe Analisi UOC.xls" en el que es pot veure la informació obtinguda de les capçaleres HTTP i les pàgines d'error de la UOC.

Presentació

Adjunt al treball també hi ha el l'arxiu "Presentacio.pdf" que inclou les diapositives utilitzades per presentar el treball. Per altra banda, el vídeo amb les demostracions es pot descarregar de les següents URLs:

Format WMV: https://drive.google.com/file/d/0B-JwECT0TxoqR0p0Y0Fic3E1WDA/edit?usp=sharing

Format AVI: https://drive.google.com/file/d/0B-JwECT0TxoqY29wTURqSUJoTGc/edit?usp=sharing