Màster interuniversitari de seguretat de les tecnologies de la informació i de les comunicacions.

(MISTIC)

SEGURETAT EN APLICACIONS WEB

Treball fi de màster

Autor: Borja Guaita Pérez
Directors: Agustí Solanas i Jordi Duch
Juny 2015

Descripció del projecte

Generació de reports de vulnerabilitats i amenaces per a aplicacions web.
La quantitat d'informació personal i privada accessible a través d'un navegador web està augmentat exponencialment cada any però les mesures de seguretat per a protegir l'accés a aquestes dades no se sol considerar com una una prioritat per als desenvolupadors web, generalment degut a la complexitat per a la seva detecció. L'objectiu d'aquest projecte és proporcionar una eina amb una interfície fàcil d'utilitzar que puga ajudar als dissenyadors i desenvolupadors web a identificar els punts febles de les seves webs. Aquesta eina tindrà dues parts separades, una que identificarà les possibles amenaces de seguretat mitjançant la combinació de eines de detecció que ja existeixen al mercat i una segona part on es dissenyarà un sistema per a recollir els resultats i generar els informes que ajudaran a la resolució dels problemes.

Project description

Generation of vulnerabilities and threat reports for web applications
The amount of personal and private information accessible through a web browser is increasing exponentially every year, but the security measures to protect the access to this content are not usually considered as a top priority for the web developers, usually due to the complexity of finding them. The aim of this project is to provide a small product with a user friendly interface that will help web designers and developers identify the weak points of their websites. The product will have two separate parts, first it will require the identification of the security threats using a combination of analysis and exploration tools that are currently in the market, and second the design of a system to collect the results of the identification and generate reports that will help solve the problems found.

Índex de continguts

Capítol 1 – Introducció

1.1 - Introducció

Avui en dia l'ús d'Internet com a eina de treball i oci està en el dia a dia de les persones i les empreses. A nivell personal es comparteixen moltes dades en xarxes socials, blogs, fòrums, etc. i a nivell professional les empreses ofereixen els seus serveis a través d'aquesta xarxa.

Les dades que poden aglutinar tots aquests serveis són desitjades per gent que vol fer un mal ús d'aquestes ja siga per interessos personals, econòmics o delictius i és per això que en els últims anys hem observat com els atacs contra totes aquestes plataformes han augmentat de manera significativa.

Aquestos atacs poden tindre diferents motivacions com poden ser la recopilació de dades personals (correus electrònics, números de telèfons o números de targetes de crèdit), accés a bases de dades d'empreses, atacs dirigits de denegació de servei contra empreses en concret, realitzar canvis a la webs sense consentiment, accés als servidors, instal·lació de codi maliciós, etc.

Si ens centrem en la part de les webs exposades a la xarxa podem dir que els principals modes d'atacs serien per una part aprofitar les males configuracions de les eines que la formen i per un altra part l'explotació de les dades d'entrada a la web, com per exemple, els formularis. Quan introduïm dades a una web aquestes son interpretades per el sistema i interactuen amb altres serveis com poden ser les bases de dades i és per això que s'ha de tenir molta cura del control d'aquestes dades.

Existeixen molts tipus d'atacs que intenten aprofitar diferents tipus de vulnerabilitats dels que podem destacar els de tipus d'injecció *SQL (SQLi)*, *els Cross-Site Scripting (XSS)*, vulnerabilitats d'inclusió (*LFI/RFI*) o accés per força bruta cap a un control d'accés. D'aquestos atacs en parlarem més endavant amb més detall.

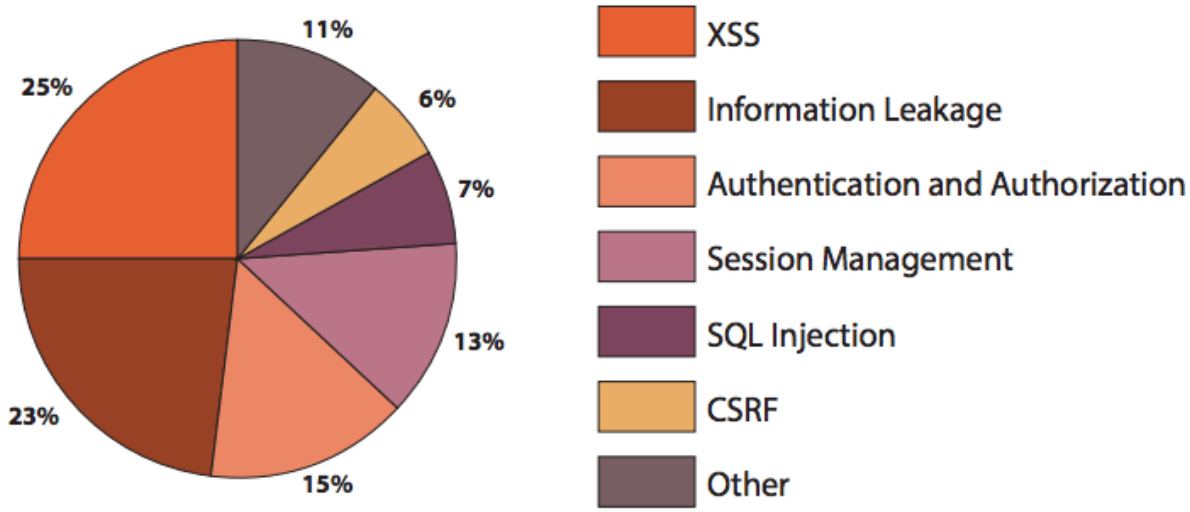
Per tal de conscienciar sobre la importància d'aquests atacs i tipus de vulnerabilitats existeixen diferents organismes que intenten mitigar aquestes accions malintencionades. Aquests organismes fan tasques d'estudi de les vulnerabilitats i dels diferents atacs per a proporcionar informació per tal de poder mitigar aquestes accions. Per un altra banda també ofereixen eines per poder comprovar la seguretat dels serveis exposats a la xarxa.

Aquestos organismes poden ser de caràcter públic o privat com poden ser els diferents Centres de Resposta a Incidents de Seguretat com el CERT i el FIRST a nivell mundial, el CCN-CERT i el CERTSI a nivell nacional o per exemple el CESICAT a nivell autonòmic. També existeixen organitzacions no governamentals com el OWASP (Open Web Application Security Project) que exerceixen una molt bona tasca per tal de millorar la seguretat en els serveis web.

També hi ha altres organismes que s'encarreguen de recopilar les vulnerabilitats que van apareixent en el dia a dia i publicar-les de forma que queden en coneixement de tots els interessats, i en el cas d'existir alguna solució també s'ofereix per a que es puguin corregir. Alguns dels organismes més importants són CVE (Common Vulnerabilities and Exposure), OSVDB (Open Source Vulnerability Database) o Packet Storm.

Aquest projecte s'ha centrat molt en la informació oferida per OWASP ja que és una organització amb molt bona reputació en el mon de la seguretat web i proporciona una guia (Testing Guide) que ens proporciona la informació necessària i algunes eines per tal de verificar i corregir les diferents vulnerabilitats que poden existir en una web.

El següent gràfic en mostra les vulnerabilitats web més conegudes i el percentatge d'atacs que es solen efectuar:



1.2 - Objectius

L'objectiu principal que s'intentarà assolir és el desenvolupament d'una web que ens permeti fer escanejos de *URLs* per tal de comprovar si aquestes tenen vulnerabilitats existents.

Aquesta web serà lo més simple possible per a poder fer-la servir sense la necessitat de tindre coneixements tècnics i de seguretat però que aquesta simplicitat no siga perjudicial a l'hora d'executar els escanejos i mostrar els resultats. Es a dir, ha de ser una web amb una interfície simple però que siga útil per a la comprovació de vulnerabilitats de seguretat.

Actualment existeixen molt bones eines especialitzades en escanejos de seguretat de tot tipus i que permeten moltes configuracions per tal de detectar vulnerabilitats determinades. Per a la realització del projecte he triat les següents: *Whatweb*, *W3af* i *Pompem*.

- **Whatweb**. És una eina especialitzada en la detecció del *fingerprint* de les webs. Ens proporciona informació referent a les aplicacions que componen la web i el servidor web que la executa com el tipus de servidor web, llenguatge de programació, capçaleres, errors de la web o aplicacions com *worpress*, *drupal*, *joomla*, *etc*.
- **W3af (Web Application Attack and Audit Framework)**. Aquesta eina ens un *framework* que ens permet auditar una web i detectar múltiples vulnerabilitats de forma automàtica i classificar-les per tal de conèixer la seva criticitat.
- **Pompem – Exploit Finder**. Eina que ens permet l'automatització per a la cerca de *exploits* existents per a les diferents vulnerabilitats en la majoria de les bases de dades de vulnerabilitats conegudes.

Els principals motius per a la elecció d'aquestes eines és que són eines de codi lliure (Open Source), que no requereixen d'una gran infraestructura ni coneixements per a la seva instal·lació i ús i per a la modularitat a l'hora de seleccionar quins tipus de vulnerabilitats es volen detectar mitjançant mòduls o *plugins*.

Finalment la web que integrarà totes aquestes es desenvoluparà amb els llenguatges de programació *html, php i python*.

1.3 - Planificació

El 16 de març de 2015 comença el projecte.

Durant les dos setmanes següents s'estudiaran i provaran les diferents eines existents per tal de valorar quines són les que millor s'adapten per a la estructura del projecte i quin serà el millor entorn per a muntar-lo.

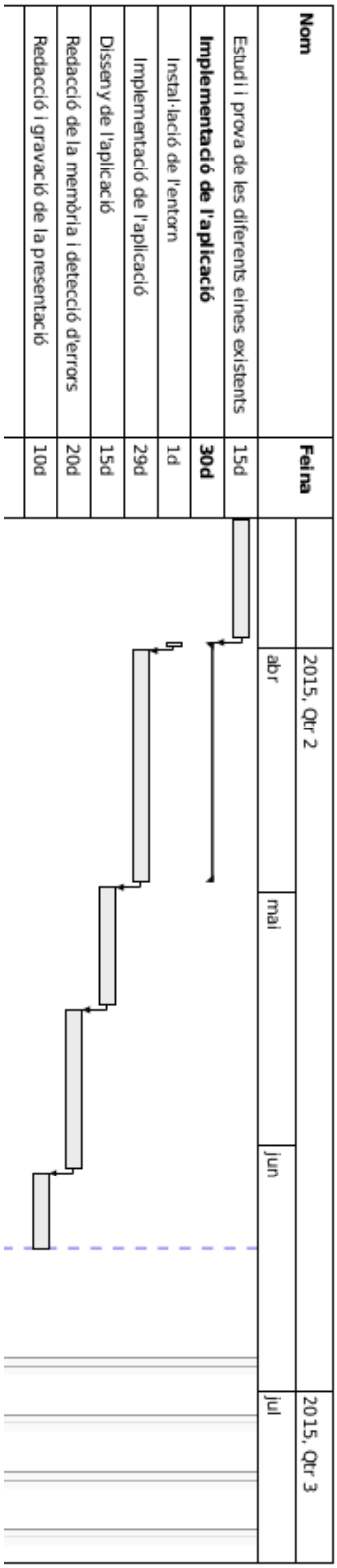
Una vegada ja estiguen les aplicacions triades durant el següent mes es seguiran fent proves específiques amb les eines triades i es començarà amb la implementació del projecte. Primer es muntarà una estructura bàsica de forma que es puguin fer les proves manuals des de línia de comandes i després s'anirà creant la web per a realitzar els escanejos i adaptar-la per tal que funcioni tot correctament.

En les dos setmanes següents s'ajustarà el disseny de la web per a que siga el més amigable possible i de forma que els resultats mostrats siguen comprensibles d'una forma clara. Durant aquestes setmanes, on el projecte ja tindrà una estructura més madura, també es podran afegir algunes opcions extra que considerem que siguen interessants per al projecte.

Per últim, fins a la entrega del projecte s'anirà escrivint i estructurant la memòria i la presentació. Per un altra banda també s'aniran realitzant proves amb la web per tal de detectar errors i poder corregir-los.

WBS	Nom	Comença	Acaba	Feina	Durada
1	Estudi i prova de les diferents eines existents	16 mar	30 mar	15d	15d
2	**Implementació de l'aplicació**	**31 mar**	**29 abr**	**30d**	**30d**
2.1	Instal·lació de l'entorn	31 mar	31 mar	1d	1d
2.2	Implementació de l'aplicació	1 abr	29 abr	29d	29d
3	Disseny de l'aplicació	30 abr	14 mai	15d	15d
4	Redacció de la memòria i detecció d'errors	15 mai	3 jun	20d	20d
5	Redacció i gravació de la presentació	4 jun	13 jun	10d	10d

1.3.1 Diagrama de Gantt



Capítol 2 – Tipus de vulnerabilitats

Existeixen molts tipus de vulnerabilitats webs diferents i cada dia van apareixent noves modificacions que afecten a tot tipus de tecnologia web. En aquest apartat s'explicarà alguns dels diferents tipus de vulnerabilitats més comuns per tal d'entendre millor en què consisteixen.

2.1 - Injecció de codi

Les vulnerabilitats de injecció de codi son degudes al no filtrar i fer un control sobre les dades d'entrada a una web, per exemple, mitjançant formularis. En aquesta categoria podem trobar varis tipus de injecció de codi com poden ser els *SQL Injection (SQLI)*, *Cross-Site Scripting (XSS)* o *LDAP Injection*.

La prevenció d'aquests tipus de vulnerabilitats passa per filtrar sempre les dades que es puguin introduir a la web ja siga mitjançant formularis, funcions d'accés d'usuaris o mitjançant la pròpia *url* de la web.

2.1.1 – SQL Injection

La injecció de codi *SQL* és un mètode que intenta inserir codi *SQL* aprofitant un error del programari de l'aplicació a nivell de validació de les dades d'entrada per poder realitzar consultes o modificacions a la base de dades de l'aplicació.

2.1.2 – Blind SQL Injection

Aquesta vulnerabilitat és un tipus de *SQL Injection* però que només executa comandes a la base de dades de forma que la resposta siga només verdader o fals. Aquest tipus d'atac es sol fer servir quan l'aplicació web només mostra errors genèrics al no produir-se resultats correctes al fer una consulta a la base de dades. Es a dir, només hi haurà resposta si el resultat és correcte.

Aquesta tècnica es pot fer servir en combinació amb diccionaris per tal d'intentar extreure informació de la base de dades mitjançant la tècnica coneguda com força bruta.

2.1.3 – XSS

L'atac XSS és un tipus d'injecció on s'intenta introduir codi maliciós a l'aplicació evadint els propis controls de la web. La injecció pot fer-se de diferents formes encara que els llocs més comuns són els formularis web i la pròpia "url" en el cas de que accepte paràmetres.

Aquesta vulnerabilitat pot estar present de dues formes:

Directa o persistent: consisteix en introduir codi *HTML* a llocs de la web que no validen correctament la informació introduïda, com les etiquetes de tipus *<script>* o *<iframe>*, de manera que el codi es quedi permanentment a la web ja siga en bases de dades o fitxers de la web..

Indirecta: en aquest cas, a diferència de l'anterior, el codi introduït a la web no es persistent en el temps, es a dir, només afecta a l'usuari que utilitza el navegador.

2.1.4 – LDAP Injection

Els atacs de *LDAP Injection* s'utilitzen per a atacar aplicacions web que construeixen sentències de tipus *LDAP* basades en la informació introduïda per l'usuari. Quan una aplicació falla a l'hora de verificar les dades introduïdes per l'usuari es possible modificar les sentències del *LDAP* i d'aquesta manera es pot intentar aconseguir executar comandes arbitraries per tal d'intentar aconseguir permisos no autoritzats o la modificació de l'arbre del *LDAP*.

2.2 – Injecció de comandes

Existeixen alguns llocs web que permeten a l'usuari executar comandes del sistema a través d'una entrada. El problema apareix quan no es fa un bon filtratge de les comandes introduïdes i de l'usuari que les executa al sistema ja que un usuari malintencionat podria intentar executar comandes diferents, per exemple, després de les comandes permeses mitjançant un "*pipe*".

2.3 - Session Hijacking

Session Hijacking (robatori de sessió) és una tècnica on un atacant aconsegueix l'identificador de sessió entre la pàgina web i l'usuari, d'aquesta manera es pot fer passar per el usuari i accedir a la web amb els permisos d'aquest. Existeixen diferents mètodes per a aconseguir la sessió d'un usuari, a continuació exposaré els més comuns.

2.3.1 – Atac per força bruta

Aquest atac consisteix en fer servir mètodes de força bruta per a crear identificadors de sessió de forma aleatòria fins que es trobi un que s'estiga fent servir a la web.

2.3.2 – Sniffing

L'atac per *sniffing* implica que l'atacant tinga una eina de *sniffing* a la xarxa de l'usuari afectat, d'aquesta manera pot interceptar el trànsit generat per el usuari, en cas de que aquest no estiga xifrat, i descobrir el identificador de la sessió, per exemple, mitjançant un atac de "*man in the middle"*.

També podem tindre un problema paregut quan està allotjada la web en un servidor compartit amb una configuració poc restrictiva ja que les sessions es guarden per defecte en un directori comú per a totes les webs per tant tots els usuaris que tinguen accés al servidor tindrien accés a tots els fitxers de sessió.

2.3.3 – Propagació a la URL

L'identificador de sessió s'envia a l'usuari mitjançant les galetes o directament en una part de la URL. És aquest segon cas el que pot ser aconseguit de diferent maneres per atacants malintencionats ja siga en un enllaç publicat per el propi usuari, mirant el historial de navegador o buscant a la capçalera "*referrer*" que envien alguns dels navegadors.

2.3.4 – Cross-Site Scripting (XSS)

El *Cross-Site Scripting* o *XSS*, com ja hem vist anteriorment, és una vulnerabilitat que ens permet la injecció de codi en una web. Si una pàgina web es vulnerable a *XSS* aquest pot ser aprofitat per a introduir codi que envie les galetes de sessió d'un usuari a un compte seu i poder extreure l'identificador de sessió.

2.4 – Session Fixation

La fixació de sessió (*Session Fixation*) és un mètode de robatori de sessió especial ja que normalment s'intenta aconseguir el identificador de sessió de l'usuari i en aquest cas el que se intenta aconseguir és assignar un identificador conegut per el atacant a un usuari abans de que aquest s'autentiqui a la web.

2.5 – Information Leakage

Aquest tipus de vulnerabilitat s'entén com una fuita d'informació no controlada de forma que un atacant puga arribar a conèixer informació que no està autoritzat. Normalment es dona quan un procés dissenyat per a restringir l'accés a certa informació només a usuaris autoritzats revela part d'aquesta informació degut a errors en el disseny de l'aplicació. Un exemple podria ser fer servir un protocol de comunicació sense xifratge on un usuari malintencionat pot capturar aquesta comunicació i veure informació com usuaris i contrasenyes.

2.6 – Remote File Inclusion (RFI)

Vulnerabilitat que només afecta a pàgines web programades en "PHP" a causa d'una mala programació que permet l'enllaç de fitxers remots, mitjançant la funció "*include()*", que estan a altres servidors.

Un atacant podria aprofitar aquest error, per exemple, per a modificar la crida que realitza la *url* de la web per a intentar executar un fitxer remot controlat per ell de forma que li permeta executar comandes al servidor legitim.

2.7 – Cross-Site Request Forgery (CSRF / XSRF)

Aquesta vulnerabilitat permet que un atacant puga executar comandes no autoritzades mitjançant un usuari legítim del lloc web. Es a dir, permet generar peticions sobre l'aplicació web vulnerable a partir de la sessió d'un usuari legítim de l'aplicació. En aquest cas l'usuari legítim seria víctima de

l'atac.

Aquests tipus d'atacs forcen que el navegador web de la víctima (usuari legítim) envii una petició a l'aplicació web vulnerable per tal de realitzar l'acció maliciosa elegida a través d'ell.

2.8 – Insecure Direct Object Reference

És una de les vulnerabilitats més explotades i aprofitades en aplicacions basades en web. Es dona quan existeix un mal desenvolupament de l'aplicació on es poden trobar referències a objectes interns de la implementació, com per exemple fitxers, directoris o bases de dades sense cap tipus de mecanismes de validació.

Aquests tipus d'errors poden ser aprofitats per atacants per a accedir a informació no autoritzada o modificar alguns paràmetres per tal d'arribar a altres parts de l'aplicació.

Capítol 3 – Descripció de l'aplicació.

L'aplicació desenvolupada en aquest projecte intenta ser una eina per tal de poder detectar vulnerabilitats en pàgines web.

Aquesta aplicació intenta ser lo més simplista possible per tal de proporcionar un mitjà a persones que no tenen alts coneixements informàtics relacionats amb la seguretat d'aplicacions web i que mostri els resultats de forma clara.

L'eina és la unió de tres aplicacions ja existents i una web que fa de nexe entre elles a l'hora d'executar un test a un altra web.

Les eines que engloben l'aplicació, com ja hem comentat en un punt anterior són *Whatweb* que s'encarrega de fer el *fingerprint* de la web escanejada, w3af que fa la cerca de les vulnerabilitats que puguen existir en la web i per últim *Pompem* que fa la cerca de possibles *exploits* existents en les diferents bases de dades *d'exploits* conegudes.

La web principal mostra únicament un formulari on introduirem la *url* de la web que es vol testejar i una vegada a finalitzat el escaneig es mostra un altra plana amb els resultats obtinguts. La plana amb els resultats els classifica en quatre grups Informació, Baixa, Mitjana i Alta mostrant cada grup amb un color diferent, blau per a Informació, verd per a Baixa, taronja per a Mitjana i roig per a Alta.

Les eines que engloben l'aplicació, especialment *Whatweb* i *W3af* tenen moltes opcions de configuració a base de mòduls i *plugins*. Per a aquest projecte, després de valorar les opcions, s'ha decidit que no es donaria la opció de personalitzar les aplicacions i així fer una eina que siga simplista de cara a l'usuari final, entenent que aquest no tindrà molts coneixements sobre vulnerabilitats web. Per tant, el que s'ha fet es crear un perfil bàsic amb les opcions que he considerat més importants de forma que quan es realitze un test a una web siga més o menys ràpid i detecte les vulnerabilitats més importants.

Capítol 4 – Estructura de l'aplicació

L'aplicació està dissenyada per tal de seguir un model que ens permet primer cercar informació sobre la estructura de la web que es vol testejar, segon intentar trobar les diferents vulnerabilitats que puguin existir a la web i per últim cercar als diferents repositoris de vulnerabilitats conegudes si existeix algun "*exploit*" conegut que afecte a la web.

Seguidament explicaré les tres eines que s'han triat per a realitzar les diferent fases del test i la web resultant de la integració d'aquestes.

4.1 – Whatweb

A l'hora de fer un test sobre una web moltes vegades els desenvolupadors fan servir gestors de continguts (*CSM*) o aplicacions webs ja conegudes, per tant, pot ser molt útil saber si les webs a les que volem fer un test fan servir algun d'aquestos *CMS* per tal de buscar si la versió que estan fent servir pot estar afectada per alguna vulnerabilitat ja coneguda.

Whatweb és una eina que ens permet fer aquesta tasca de forma còmoda des de la línia de comandes. És una aplicació senzilla programada en ***Python*** i que no necessita d'una instal·lació complicada.

L'execució del programa és simple i com veurem a la següent imatge se li pot passar com a paràmetre una sola *url* o un fitxer amb varies *urls* a analitzar.

```
borja@sw_test:/var/www/uoc_project/whatweb$ ./whatweb

.$$$     $.                                   .$$$     $.
$$$$     $$. .$$$  $$$ .$$$$$$.  .$$$$$$$$$$. $$$$     $$. .$$$$$$$. .$$$$$$.
$ $$     $$$ $ $$  $$$ $ $$$$$$. $$$$$ $$$$$$ $ $$     $$$ $ $$   $$ $ $$$$$$.
$ `$     $$$ $ `$  $$$ $ `$  $$$ $$' $ `$ `$$ $ `$     $$$ $ `$      $ `$  $$$'
$. $     $$$ $. $$$$$$ $. $$$$$$ `$  $. $  :' $. $     $$$ $. $$$$   $. $$$$$.
$::$   .  $$$ $::$  $$$ $::$  $$$     $::$     $::$   .  $$$ $::$      $::$  $$$$
$;;$ $$$ $$$ $;;$  $$$ $;;$  $$$     $;;$     $;;$ $$$ $$$ $;;$      $;;$  $$$$
$$$$$$ $$$$$ $$$$  $$$ $$$$  $$$     $$$$     $$$$$$ $$$$$ $$$$$$$$$ $$$$$$$$$'

WhatWeb - Next generation web scanner version 0.4.8-dev.
Developed by Andrew Horton aka urbanadventurer and Brendan Coles
Homepage: http://www.morningstarsecurity.com/research/whatweb

Usage: whatweb [options] <URLs>

TARGET SELECTION:
  <URLs>                Enter URLs, filenames or nmap-format IP ranges.
                        Use /dev/stdin to pipe HTML directly
  --input-file=FILE, -i Identify URLs found in FILE. You can pipe
                        hostnames or URLs directly with -i /dev/stdin
```

Per a la nostra aplicació la execució serà només amb una única *url* que se li passarà des de un formulari web.

També ens permet definir l'agressivitat amb la que s'executarà l'escaneig. En aquest cas la configuració que s'ha aplicat a l'aplicació es la que s'executa per defecte "Stealthy" per tal que no siga molt agressiva.

```
AGGRESSION:
  The aggression level controls the trade-off between speed/stealth and
  reliability.
  --aggression, -a=LEVEL Set the aggression level. Default: 1
  Aggression levels are:
  1. Stealthy   Makes one HTTP request per target. Also follows redirects.
  2. Unused
  3. Aggressive Can make a handful of HTTP requests per target. This triggers
                aggressive plugins for targets only when those plugins are
                identified with a level 1 request first.
  4. Heavy      Makes a lot of HTTP requests per target. Aggressive tests from
                all plugins are used for all URLs.
```

Per un altra banda comentar que es compon de més de 900 mòduls que són els que permeten detectar el tipus de tecnologia que s'està utilitzant a les webs. Es pot veure un llistat de tots els "*plugins*" amb l'opció '*--list-plugins*'.

```
PLUGINS:
  --list-plugins, -l    List all plugins
  --plugins, -p=LIST    Select plugins. LIST is a comma delimited set of
                        selected plugins. Default is all.
                        Each element can be a directory, file or plugin name and
                        can optionally have a modifier, eg. + or -
                        Examples: +/tmp/moo.rb,+/tmp/foo.rb
                        title,md5,+./plugins-disabled/
                        ./plugins-disabled,-md5
                        -p + is a shortcut for -p +plugins-disabled
  --info-plugins, -I=PLUGINS    Display detailed information for plugins.
                        Optionally search with keywords in a comma delimited
                        list.
  --grep, -g=STRING     Search for STRING in HTTP responses. Reports with a
                        plugin named Grep
  --custom-plugin=DEFINITION    Define a custom plugin named Custom-Plugin,
                        Examples: ":text=>'powered by abc'"
                        ":version=>/powered[ ]?by ab[0-9]/"
                        ":ghdb=>'intitle:abc \"powered by abc\"'"
                        ":md5=>'8666257030b94d3bdb46e05945f60b42'"
                        "{:text=>'powered by abc'},{:regexp=>/abc [ ]?1/i}"
  --dorks=PLUGIN        List Google dorks for the selected plugin
```

Per defecte l'aplicació executa tots els "*plugins*" disponibles però per a intentar guanyar un poc de temps a l'hora de fer un test per al projecte s'ha decidit fer una selecció dels gestors de contingut més populars que existeixen al mercat.

"*Plugins*" seleccionats per al projecte:

- Drupal
- WordPress
- Joomla
- Blogger
- Magento
- PrestaShop
- Zope
- OSCommerce
- Plone
- TYPO3
- Zen-Cart
- Mambo
- XOOPS
- dotclear
- Dspace
- OpenCms
- Moodle
- ModxCMS
- Symfony
- Dokeos
- Claroline
- Chamilo

Per últim també és important les opcions que ens dona l'eina per tal de generar el report dels test que s'ha realitzat. Com podem veure a la següent figura tenim varies opcions i la que més s'adapta al projecte és crear el report en format *XML* ja que és fàcilment interpretable amb els llenguatge de programació *PHP*.

```
OUTPUT:
  --verbose, -v         Verbose output includes plugin descriptions. Use twice
                        for debugging.
  --colour,--color=WHEN control whether colour is used. WHEN may be `never',
                        `always', or `auto'
  --quiet, -q           Do not display brief logging to STDOUT
  --no-errors           Suppress error messages

LOGGING:
  --log-brief=FILE      Log brief, one-line output
  --log-verbose=FILE    Log verbose output
  --log-xml=FILE        Log XML format
  --log-json=FILE       Log JSON format
  --log-json-verbose=FILE Log JSON Verbose format
  --log-magictree=FILE  Log MagicTree XML format
  --log-object=FILE     Log Ruby object inspection format
  --log-mongo-database  Name of the MongoDB database
  --log-mongo-collection Name of the MongoDB collection. Default: whatweb
  --log-mongo-host      MongoDB hostname or IP address. Default: 0.0.0.0
  --log-mongo-username  MongoDB username. Default: nil
  --log-mongo-password  MongoDB password. Default: nil
  --log-errors=FILE     Log errors
```

4.2 – W3af

W3af (Web Application Attack and Audit Framework) és una eina de codi obert per a realitzar tests d'intrusió a webs, està desenvolupada en *Python* i la seva funcionalitat principal és realitzar escanejos de vulnerabilitats de tipus web. L'aplicació és senzilla d'utilitzar i ens es d'utilitat per a automatitzar diferents anàlisis en un mateix procés.

Aquesta eina és totalment modular on cada mòdul serveix per a buscar diferents tipus de vulnerabilitats. Tots els mòduls es poden configurar individualment i estan agrupats en diferents categories.

Ens permet treballar en dos entorns un en línia de comandes (que és el que farem servir al projecte) i un altre amb una interfície gràfica. L'execució per línia de comandes executa una consola per on ens podem moure per les diferents categories per tal de configurar l'aplicació i seleccionar els mòduls que volem fer servir.

```
borja@sw_test:/var/www/uoc_project/w3af$ ./w3af_console
w3af>>> help
|-------------------------------------------------------------------------------------------------|
| start          | Start the scan.                                                                |
| plugins        | Enable and configure plugins.                                                  |
| exploit        | Exploit the vulnerability.                                                     |
| profiles       | List and use scan profiles.                                                    |
| cleanup        | Cleanup before starting a new scan.                                            |
|-------------------------------------------------------------------------------------------------|
| help           | Display help. Issuing: help [command] , prints more specific help about       |
|                | "command"                                                                      |
| version        | Show w3af version information.                                                 |
| keys           | Display key shortcuts.                                                         |
|-------------------------------------------------------------------------------------------------|
| http-settings  | Configure the HTTP settings of the framework.                                  |
| misc-settings  | Configure w3af misc settings.                                                  |
| target         | Configure the target URL.                                                      |
|-------------------------------------------------------------------------------------------------|
| back           | Go to the previous menu.                                                       |
| exit           | Exit w3af.                                                                     |
|-------------------------------------------------------------------------------------------------|
| kb             | Browse the vulnerabilities stored in the Knowledge Base                        |
|-------------------------------------------------------------------------------------------------|
w3af>>>
```

Si ens centrem en la part del "*plugins*" existents per a la cerca de vulnerabilitats veiam que estan estructurants en nou categories diferents, aquesta característica ens facilita la feina a l'hora de triar quins tipus de "*plugins*" volem triar.

1. **Infrastructure.** Utilitzen diferents tècniques per tal d'identificar el funcionament del sistema del que s'està fent el test i ens donarà informació com el tipus de servidor web, sistema operatiu, si existeix algun tallafocs, usuaris remots, etc.
2. **Bruteforce.** La tasca principal d'aquestos "*plugins*" és fer un rastreig a la web en busca de inicis de sessió i intenten aconseguir l'accés mitjançant accions de força bruta.

3. **Grep.** Analitzen les respostes a les peticions que es van realitzant de forma que es puga extreure informació com errors, comentaris al codi font, correus electrònics, direccions IP privades, galetes de sessió, etc.
4. **Output.** Aquestos "*plugins*" són els que s'encarreguen de generar un tipus i format de sortida dels reports generats, per exemple, fitxer XML o HTML, un correu electrònic entre altres.
5. **Mangle.** Serveixen per a modificar peticions i respostes durant l'escaneig.
6. **Audit.** Obtenen informació dels "*plugins*" de "*Crawl*" per tal de localitzar vulnerabilitats del tipus *SQLi*, *buffer overflow*, *csrf*, *dav*, *xss*, *ldapi*, *file upload*, *etc*.
7. **Crawl.** Aquests tipus de "*plugins*" fan servir diferents tipus de tècniques per a identificar noves *urls*, formularis o qualsevol altre recurs que podria ser d'utilitat durant el test. També alimenten els "*plugins*" configurats de la categoria de "*Bruteforce*".
8. **Evasion.** Ens permeten modificar les peticions o part de les peticions realitzades per tal d'intentar aconseguir evadir la detecció dels IDP o IPS.
9. **Auth.** Aquestos "*plugins*" fan possible l'escaneig de webs que requereixen autenticació. Abans de realitzar el test executarà un inici de sessió i al finalitzar tancaran la sessió oberta.

```
w3af/plugins>>> help
|-------------------------------------------------------------------------------------------------------|
| list                | List available plugins.                                                         |
|-------------------------------------------------------------------------------------------------------|
| back                | Go to the previous menu.                                                        |
| exit                | Exit w3af.                                                                      |
|-------------------------------------------------------------------------------------------------------|
| infrastructure      | View, configure and enable infrastructure plugins                               |
| bruteforce          | View, configure and enable bruteforce plugins                                   |
| grep                | View, configure and enable grep plugins                                         |
| output              | View, configure and enable output plugins                                       |
| mangle              | View, configure and enable mangle plugins                                       |
| audit               | View, configure and enable audit plugins                                        |
| crawl               | View, configure and enable crawl plugins                                        |
| evasion             | View, configure and enable evasion plugins                                      |
| auth                | View, configure and enable auth plugins                                         |
|-------------------------------------------------------------------------------------------------------|
```

Un altra característica important d'aquest programa és la de poder agrupar les diferents configuracions dels mòduls en perfils. Existeixen alguns perfils ja preconfigurats com el "f*ast_scan*" que realitza un escaneig ràpid amb "*plugins*" de les categories "Discovery" i "Audit" o el perfil "*OWASP_TOP10*" que es centra només en les deu principals vulnerabilitats catalogades per el *OWASP* però també ens dona la possibilitat de crear els nostres propis perfils.

Després d'avaluar les diferents opcions i perfils existents per al projecte s'ha decidit realitzar un perfil propi de forma que es puguen identificar les vulnerabilitats més habituals i més perilloses, per un altra part, també s'ha valorat la velocitat d'execució de forma que l'escaneig total de la web no trigui un temps excessivament elevat. A **l'Annex 1** es pot veure el perfil creat per al projecte amb les diferents configuracions seleccionades.

Igual que en l'aplicació descrita en el punt anterior, i ja que l'eina *W3af* ho permet, s'ha seleccionat com a format de sortida per als reports generats el tipus de fitxer en format *XML* per tal d'aprofitar al màxim el codi creat en *PHP* per a la interpretació d'aquest format.

4.3 – Pompem

Com ja s'ha comentat anteriorment la idea del projecte és aconseguir un anàlisis mitjançant el mètode de obtenció de informació de la web, anàlisis de vulnerabilitats a la web i cerca de "*exploits*" coneguts a les diferents bases de dades de vulnerabilitats existents.

Les dos eines explicades anteriorment en els punts 4.1 i 4.2 s'encarreguen de l'obtenció de la informació relativa a la web que s'està escanejant i l'anàlisi de les vulnerabilitats existents. Per tal de completar el model ***Pompem*** serà l'eina encarregada de cercar els "*exploits*" coneguts per a la informació obtinguda.

L'aplicació està desenvolupada en *Python* i no requereix d'una instal·lació i configuració complexa, simplement s'ha de baixar dels seus repositoris i executar-la des de la línia de comandes. És una eina de codi obert dissenyada per a automatitzar la cerca de "*exploits*" a diferents bases de dades com *Exploit-db*, *1337day* o *Packetstorm Security* entre altres. Com podem observar en la següent imatge no te gaires opcions, només cal passar-li com a paràmetre les paraules que volem cercar i el format de sortida per al report generat.

```
borja@sw_test:/var/www/uoc_project/Pompem-dev$ python pompem.py

           Pompem - Exploit Finder  |  Developed by Relax Lab

   Rafael Francischini (Programmer and Ethical Hacker) - @rfunix

   Bruno Fraga (Security Researcher) - @brunofraga_net

             Usage: pompem.py [-s/--search <keyword,keyword,keyword,...>]
                              [--txt Write txt file                      ]
                              [--html Write html file                    ]
                              [-g/--get Download exploit files           ]

             Get basic options and Help, use: -h\--help
```

Com a funcionalitat extra l'aplicació ens ofereix l'opció de descarregar-nos al nostre equip els "*exploits*" que es puguin trobar.

Com que aquest projecte està basat només en l'obtenció de la informació de la infraestructura de la web que s'està escanejant aquesta última opció no la farem servir.

Com podem veure l'aplicació no ofereix l'opció de desar els reports en un fitxer amb el format *XML* com les dos eines anteriors ja que només podem triar com a sortida de l'anàlisi els formats *TXT* i *HTML*. En aquest cas, per a la realització del projecte, s'ha elegit el tipus de format *HTML*.

4.4 – Aplicació web

Fins ara s'ha explicat individualment les diferents eines que es fan servir en el projecte. En aquest apartat s'explicarà com s'ha organitzat la web per tal de unir les aplicacions anteriors de forma que s'executin en un bloc a partir d'un formulari d'entrada. Com veurem tot seguit els llenguatges de programació que s'han fet servir són el *html*, *php* i *Python*.

La web està formada principalment per quatre fitxers:

- **index.html**

  Aquest fitxer conté el formulari web amb el disseny de la portada i una vegada s'introdueix una *url* fa una crida al fitxer *scan.php*.

- **scan.php**

  Des de aquest fitxer es d'on s'executen les crides a les eines *Whatweb* i *W3af* per a realitzar l'anàlisi de la *url* que s'ha introduït en el formulari. Una vegada tenim els reports generats en format *XML* des del mateix fitxer els llegim i interpretem.

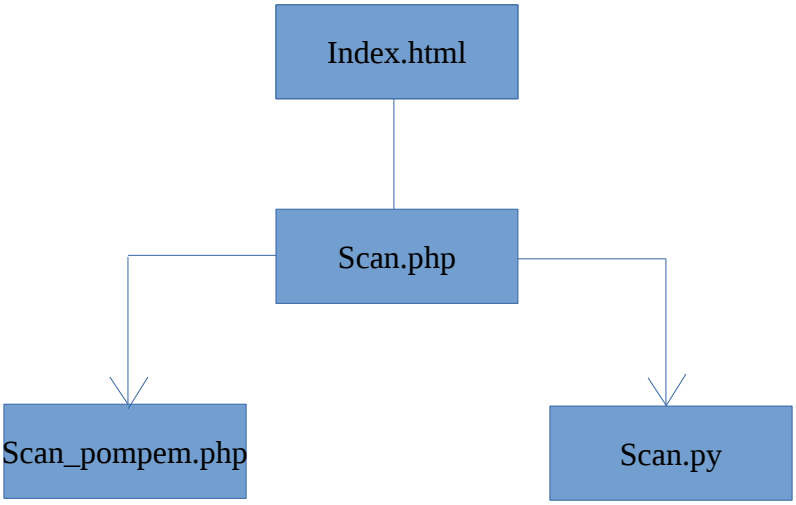
- **scan.py**

  A aquest fitxer se'l crida des del fitxer anterior i el que fa es carregar el perfil que s'ha creat per a l'execució de l'eina W3af i configurar el format de sortida dels reports.

- **scan_pompem.php**

  Igual que el fitxer anterior aquest també se'l crida des del fitxer *scan.php* i és l'encarregat d'executar l'eina *Pompem*

Per tal d'entendre millor el flux de l'aplicació la següent figura intenta mostrat quines són les dependències dels fitxers.



Una vegada explicada de forma general com es composa l'aplicació ara explicaré més en detall en què consisteix cada fitxer.

4.4.1 – Index.html

Aquesta part de la implementació no te cap dificultat. Com ja s'ha comentat simplement és un formulari web que mitjançant el mètode "*POST*" envia la *url* introduïda cap al fitxer "*scan.php*".

El codi del formulari seria el següent:

```
<form method="POST" action="scan.php">
    <input type="text" name="urlSearch" id="urlSearch" value="http://www.domain.com"
      default="http://www.domain.com">
      <button type="submit"><img class="icon-search" src="images/search.png" /></button>
      <div></div>
    </form>
```

Com podem observar és un formulari normal que en el camp "*action*" executa el fitxer "*scan.php*"

4.4.2 – Scan.php

Aquest fitxer està programat amb els llenguatges de programació "*html*" i "*php*" i és el fitxer principal de tota l'aplicació ja que s'encarrega de rebre la "url" que es vol escanejar, executa les tres eines que engloben el projecte i processa els diferents reports generats per tal de mostrar-los en un report general.

La seva funció principal és executar les eines "*Whatweb*" i "*W3af*" i anar processant els reports generats a partir dels fitxers "*XML*" per a finalment mostrar-los en format web.

L'execució d'aquestes dos eines i la lectura dels reports generats es fa mitjançant "PHP" en aquesta part del codi:

```
if (file_exists('reports/ww_' . $host)) {
   unlink('reports/ww_' . $host);
}
exec('./whatweb/whatweb ' . $url . ' -p whatweb/plugins-enabled' . ' --log-xml reports/' . 'ww_' . $host);
$xml_ww = simplexml_load_file('reports/ww_' . $host) or die("Error: Cannot create object");


if (file_exists('reports/w3_' . $host)) {
   unlink('reports/w3_' . $host);
}
exec('python scan_test.py' . ' ' . $url);
$xml = simplexml_load_file('reports/w3_' . $host) or die("Error: Cannot create object");
```

Com podem observar abans de fer l'execució comprovem que no existeix cap fitxer de report per a la *url* escanejada. En cas d'existir el report l'esborrem.

Una vegada executades les dos aplicacions es guarden els resultats a les variables "*$xml_ww*" per a l'eina "*Whatweb*" i "*$xml*" per a l'eina "*W3af*".

També podem veure al codi que es realitza una crida a la funció "s*implexml_load_file*". Aquesta funció és l'encarregada de processar els fitxers en format "*XML*".

La resta de codi del fitxer està dedicat a anar mostrant la web resultant amb el report general de l'escaneig a partir dels diferents registres emmagatzemats a les variables "*$xml_ww*" i "*$xml*" i classificar les vulnerabilitats per categories (Informació, Baixa, Mitja i Alta).

Per tal d'entendre millor el contingut de les dos variables que acabem de comentar podem veure aquest exemple de l'estructura del report d'una vulnerabilitat:

<vulnerability id="[92]" method="GET" name="File upload form" plugin="file_upload" severity="Information" url="http://192.168.1.154/dvfw/vulnerabilities/upload/" var="None">

<description>The URL: "http://192.168.1.154/dvfw/vulnerabilities/upload/" has form with file upload capabilities.</description>

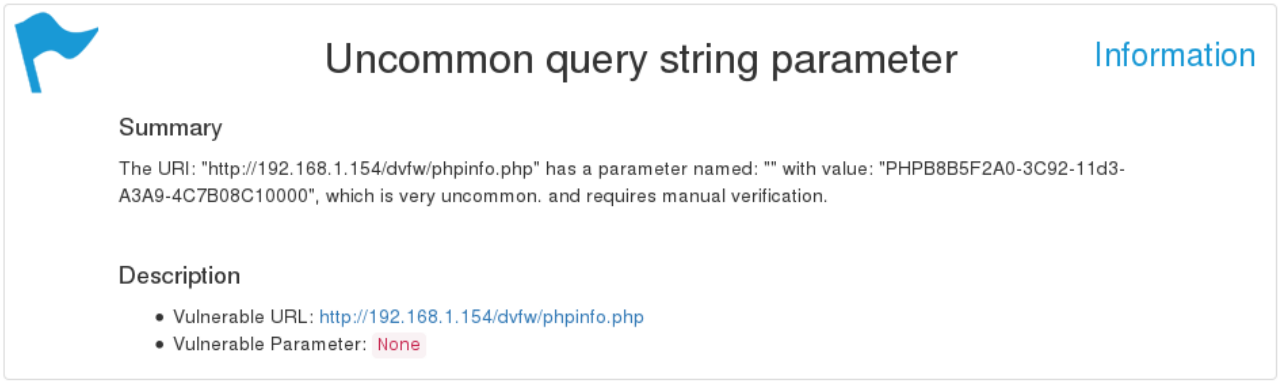
<long-description>The design of many web applications require that users be able to upload files that will either be stored or processed by the receiving web server.

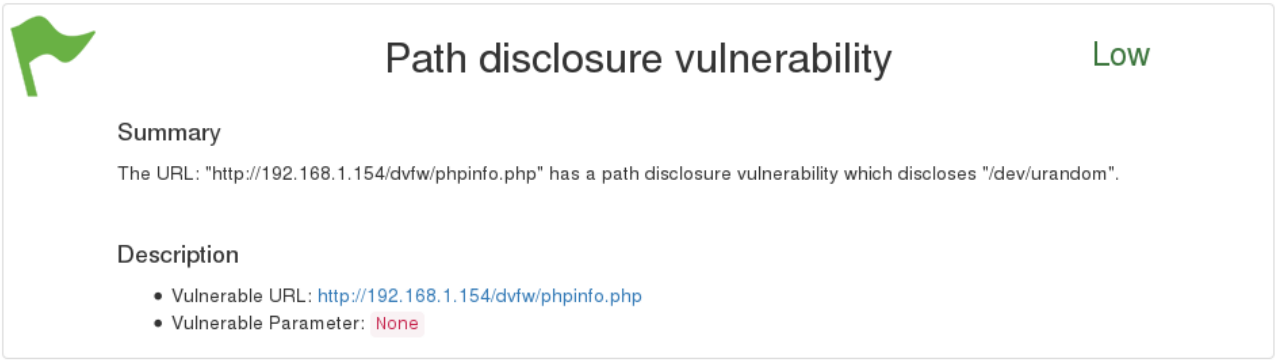
The tool has flagged this not as a vulnerability, but as a prompt for the penetration tester to conduct further manual testing on the file upload function.

An insecure form-based file upload could allow a cyber-criminal a means to abuse and successfully exploit the server directly, and/or any third party that may later access the file. This can occur through uploading a file containing server side-code (such as PHP) that is then executed when requested by the client.</long-description>

Com podem observar el text està estructurat amb etiquetes paregudes al format del llenguatge de programació "html". Veiem que en aquest exemple existeixen tres etiquetes <vulnerability>, <description> i <long-description>. D'aquesta manera podem anar buscant les diferents etiquetes dintre dels registres continguts a les variables per tal d'anar mostrant-les a la web.

El resultat final del report amb la classificació de les vulnerabilitats seria paregut al següent:





Cross site scripting vulnerability

Medium

Summary

A Cross Site Scripting vulnerability was found at: "http://192.168.1.154/dvfw/vulnerabilities/xss_r/", using HTTP method GET. The sent data was: "name=" The modified parameter was "name".

Description

Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction. Cross Site Scripting (XSS) allows clients to inject arbitrary scripting code into a request and have the server return the script to the client in the response. This occurs because the application is taking untrusted data (in this example, from the client) and reusing it without performing any validation or encoding.

- Vulnerable URL: http://192.168.1.154/dvfw/vulnerabilities/xss_r/
- Vulnerable Parameter: `name`

Fix

To remedy XSS vulnerabilities, it is important to never use untrusted or unfiltered data within the code of a HTML page. Untrusted data can originate not only form the client but potentially a third party or previously uploaded file etc. Filtering of untrusted data typically involves converting special characters to their HTML entity encoded counterparts (however, other methods do exist, see references). These special characters include: * `&` * `<` * `>` * `"` * `'` * `/` An example of HTML entity encoding is converting `<` to `<`. Although it is possible to filter untrusted input, there are five locations within an HTML page where untrusted input (even if it has been filtered) should never be placed: 1. Directly in a script. 2. Inside an HTML comment. 3. In an attribute name. 4. In a tag name. 5. Directly in CSS. Each of these locations have their own form of escaping and filtering. _Because many browsers attempt to implement XSS protection, any manual verification of this finding should be conducted using multiple different browsers and browser versions._

References

- WASC

SQL injection

High

Summary

SQL injection in a MySQL database was found at: "http://192.168.1.154/dvfw/vulnerabilities/brute/", using HTTP method GET. The sent data was: "username=a%27b%22c%27d%22&Login=Login&password=FrAmE30." The modified parameter was "username".

Description

Due to the requirement for dynamic content of today's web applications, many rely on a database backend to store data that will be called upon and processed by the web application (or other programs). Web applications retrieve data from the database by using Structured Query Language (SQL) queries. To meet demands of many developers, database servers (such as MSSQL, MySQL, Oracle etc.) have additional built-in functionality that can allow extensive control of the database and interaction with the host operating system itself. An SQL injection occurs when a value originating from the client's request is used within a SQL query without prior sanitisation. This could allow cyber-criminals to execute arbitrary SQL code and steal data or use the additional functionality of the database server to take control of more server components. The successful exploitation of a SQL injection can be devastating to an organisation and is one of the most commonly exploited web application vulnerabilities. This injection was detected as the tool was able to cause the server to respond to the request with a database related error.

- Vulnerable URL: http://192.168.1.154/dvfw/vulnerabilities/brute/
- Vulnerable Parameter: `username`

Fix

The only proven method to prevent against SQL injection attacks while still maintaining full application functionality is to use parameterized queries (also known as prepared statements). When utilising this method of querying the database, any value supplied by the client will be handled as a string value rather than part of the SQL query. Additionally, when utilising parameterized queries, the database engine will automatically check to make sure the string being used matches that of the column. For example, the database engine will check that the user supplied input is an integer if the database column is configured to contain integers.

References

- SecuriTeam

Finalment comentar que el report general de l'execució de l'aplicació està estructurat en tres parts. Una primera part ens mostra les dades referents a la *url* escanejada amb informació com la data de l'escaneig, la *ip* de la *url* i el país. Després la segona part ens apareix la informació relacionada amb la estructura del servidor escanejat com el sistema operatiu, el servidor web i les diferents tecnologies detectades. Es en aquesta part on apareix també l'opció de buscar "*exploits*" coneguts per a les tecnologies detectades mitjançant l'eina *Pompem* .I finalment la tercera part presenta les vulnerabilitats trobades durant l'escaneig classificades per colors depenent del risc de la vulnerabilitat.

El codi complet d'aquest fitxer es pot consultar a **l'annex II** de la memòria.

4.4.3 – Scan.py

A diferència de les altres eines que engloben el projecte l'aplicació "W3af" necessita de certes configuracions per a la seva execució. Aquest fitxer és l'encarregat de generar les configuracions i executar l'aplicació.

En aquest cas s'ha programat amb el llenguatge de programació *Python* i la seva execució la realitza el fitxer "*scan.php*" automàticament.

Si anem pas per pas primerament veurem que guardem a la variable "*target*" la *url* que se li passa des del fitxer "*scan.php*" i seguit es crea un fitxer temporal amb extensió "*.w3af*" on s'aniran introduint les diferents opcions. Aquestes opcions s'han de introduir com si estiguéssim navegant directament per la consola del *W3af*.

Per tal d'anar configurant les opcions primer entrem al menú "*profiles*" i carreguem el perfil creat per al projecte (amb el mòduls configurats) de nom "project.pw3af" i tornem al menú principal. Seguidament entrem al menú "*plugins*" i al *submenu* "*output*" on es configura el format de sortida del fitxer de reports que crearà l'eina. En aquest cas li definim que el fitxer de sortida siga un fitxer de format *XML* de nom "*w3_domini.xml*" i que el desi al directori "reports" situat en el directori general del projecte i tornem al menú principal de l'eina.

Ara passem al menú "*target*" on es defineix la *url* que es vol escanejar. Aquesta *url* la tenim guardada a la variable de nom "*target*" i seguidament tornem al menú principal.

A continuació configurem la comanda "*start*" per tal que comenci l'escaneig, sortim la de sortida de l'aplicació "*exit*".

Finalment executem el fitxer temporal que acabem de crear seguidament l'esborrem.

El codi complet del fitxer és el següent:

```
#!/usr/bin/python

import os, sys

from time import sleep

if len(sys.argv) != 2:
```

```
  print "Usage: python scan.py <URL>"
  sys.exit(0)
target=sys.argv[1]

# Creates a temporary w3af audit file
fname='w3af_conf/w3af_'+target.split('/')[2]+".w3af"
report=target.split('/')[2]

f=open(fname, 'w')
f.write("profiles\n")
f.write("use w3af/profiles/project.pw3af\n")
f.write("back\n")
f.write("plugins\n")
f.write("output console,xml_file\n")
f.write("\n")
f.write("output config console\n")
f.write("set verbose False\n")
f.write("back\n")
f.write("\n")
f.write("output config xml_file\n")
f.write("set output_file /var/www/uoc_project/reports/w3_"+report+"\n")
#f.write("set verbose False\n")
f.write("back\n")
f.write("back\n")
f.write("target\n")
f.write("set target "+target+"\n")
f.write("back\n")
f.write("\n")
f.write("start\n")
f.write("exit\n")
f.close()

# Invoke w3af_console with audit script
```

```
os.system("w3af/./w3af_console -s "+fname)
sleep(2)


# Remove audit script
os.system("rm "+fname)
```

4.4.4 – Scan_pompem.py

Aquest fitxer és el encarregar d'executar l'eina *Pompem* amb la paraula que rep com a paràmetre des del fitxer "*scan.php*" i és que en permet buscar si existeixen "*exploits*" relacionats amb aquesta paraula rebuda.

Està programat en *php* i, com podem veure al codi, el que fa es posicionar-se al directori de l'eina *Pompem*, comprova si ja existeix algun fitxer de report generat i en cas afirmatiu l'esborra i executa l'aplicació passant-li com a valor la paraula rebuda en la crida. Una vegada s'ha acabat l'execució es carrega el fitxer generat en *html* i es mostra per pantalla.

Per tal de carregar el nou fitxer *html* generat per l'aplicació he fet servir una llibreria de *php* anomenada "*simple_html_dom.php*" que conté la funció que necessitava "*file_get_html()*" que permet carregat un fitxer en format *html* i poder presentar-lo per pantalla.

Com a curiositat comentar que aquesta eina només crea el fitxer del report si s'executa des de dintre del directori on està l'executable, si s'intenta l'execució des de fora del directori falla la creació del fitxer.

El codi complet del fitxer és el següent:

```
<?php
  include('simple_html_dom.php');
  $var = htmlspecialchars($_POST['pompem']);
  if (file_exists('./pompem/out.html')) {
     unlink('./pompem/out.html');
  }
  exec('cd pompem && python pompem.py' . ' ' . '--html' . ' ' . '-s' . ' ' . $var);
  exec('cd ..');
  $html = file_get_html('./pompem/out.html');
  print $html;
?>
```

Finalment comentar que aquest fitxer s'executa quan es prem un botó que apareix en la pàgina de report de vulnerabilitats general del projecte.

Capítol 5 – Implantació de l'aplicació

Per al desenvolupament de la pàgina web s'han fet servir els llenguatges de programació *HTML* i *PHP*. Per tal que funcionin bé aquestos llenguatges es necessita d'una infraestructura que permeti l'execució d'aquests.

En el meu cas he optat per crear una maquina virtual amb la tecnologia *LXC* (*Linux Containers*). *LXC* és una tecnologia de virtualització a nivell de sistema operatiu que permet que un servidor físic execute múltiples instàncies de sistemes operatius aïllats. A diferència de altres plataformes de virtualització *LXC* no crea una maquina virtual amb el seu propi *kernel* i simulant el hardware del sistema és més bé com un entorn virtual que té el seu propi espai de processos i de xarxa però que comparteix les crides al *kernel* del host amfitrió.

La versió del *LXC* és la 0.9 i el servidor virtual creat és una *Debian Wheezy* (7.8) de 32 bits amb *kernel* 3.13.

Per a l'execució de l'entorn web s'ha instal·lat el servidor web *Apache2* versió 2.2.22 des dels mateixos repositoris de la *Debian* amb les seves dependències. Per un altra banda per a l'execució del llenguatge de programació *PHP* s'ha instal·lat la llibreria php5 versió 5.4.36 amb les seves dependències també des dels repositoris de la distribució *Debian*.

La instal·lació tant del servidor web *Apache2* com de les llibreries *PHP* ho podem fer amb la següent comanda:

apt-get install apache2 php5

Ens demanarà confirmar que volem instal·lar els dos paquets amb les dependències necessàries i una vegada confirmem que volem fer la instal·lació baixarà els paquets i els instal·larà al sistema.

També necessitarem instal·lar els paquets necessaris per a poder executar aplicacions escrites en el llenguatge de programació *Python* ja que les tres eines que composen la base de l'aplicació estan escrites en aquest llenguatge. En aquest cas s'ha instal·lat la versió 2.7.3 també des dels repositoris del sistema operatiu i també instal·larem dos paquets relacionats amb *Python* que necessitarem per a la instal·lació d'una de les aplicacions del projecte. Podem fer-ho amb la següent comanda:

apt-get install python2.7 python-pip python-dev

pip install –upgrade pip

pip install –upgrade virtualenv

Per últim necessitarem instal·lar l'eina ***git***. *Git* és un programari de sistema de control de versions dissenyat originalment per a sistemes operatius Linux però que ja disposa de diferents versions per a altres sistemes operatius. La instal·lació la farem també mitjançant els repositoris del sistema operatiu i la versió instal·lada és la 1.7.10. La instal·lació la executarem amb la següent comanda:

apt-get install git

5.1 – Configuració del servidor Apache2 i del PHP

Encara que el servidor on s'ha fet la programació de l'aplicació i els test està en una xarxa local i no està exposat directament a Internet s'han fet uns canvis per tal que sigui un poc més segur. La instal·lació per defecte, encara que ja és funcionat, requereix d'alguns ajustaments per tal que els servidor web doni la mínima informació possible sobre la infraestructura muntada.

Per a evitar que tant el servidor *Apache* com el *PHP* mostrin aquesta informació hem de inhabilitar algunes variables i configuracions.

5.1.1 No mostrar la versió del Apache

La configuració predeterminada conté una capçalera que mostrarà la versió del *Apache* i la versió del PHP, seria paregut a açò:

```
Server: Apache/2.2.15 (Debian) PHP/5.3.5
```

Per a evitar-ho editarem el fitxer '*security*' i canviarem les següents variables:

```
/etc/apache2/conf.d/security
ServerSignature Off
ServerTokens Prod
TraceEnable off
```

5.1.2 No mostrat informació del PHP

Igual que en el cas anterior hem de canviar certes variables del fitxer "*php.ini"* per tat que mostri la mínima informació necessària. Primer deshabilitarem les següents funcions:

```
/etc/php5/apache2/php.ini
disable_functions = ,system,show_source,phpinfo/pre>
```

I al mateix fitxer canviarem aquestes variables:

```
allow_url_fopen=Off
display_errors=Off
expose_php = Off
```

5.2 – Instal·lació Whatweb

La versió estable es la 0.4.7 i la instal·lació d'aquesta eina és molt senzilla, simplement ens hem de baixar el fitxer comprimit des de la seva web. Una vegada baixat i descomprimit només cal deixar el directori que ens crea a la ubicació desitjada. Els passos realitzats han segut aquests:

1. Descàrrega del paquet comprimit:

   wget -c http://www.morningstarsecurity.com/downloads/whatweb-0.4.7.tar.gz

2. Descomprimim el fitxer descarregat:

   tar zxvf whatweb-0.4.7.tar.gz

3. Per últim movem el directori extret a la ubicació principal del projecte:

   mv whatweb /var/www/uoc_project/

Amb aquestos tres passos ja tenim la eina instal·lada, podem comprovar la versió amb la següent comanda:

```
borja@sw_test:/var/www/uoc_project/whatweb$ ./whatweb --version
WhatWeb version 0.4.8-dev ( http://www.morningstarsecurity.com/research/whatweb/ )
```

5.3 – Instal·lació W3af

La versió instal·lada de l'aplicació és la 1.6.51 ja que és l'última versió estable. El procés també és molt senzill i es fa des del seu repositori de *github* (https://github.com/):

1. Descarreguem l'aplicació del repositori amb l'eina git:

   git clone --depth 1 https://github.com/andresriancho/w3af.git

2. Movem el directori que s'ha creat a la ubicació principal del projecte:

   mv w3af /var/www/uoc_project/

3. Executem l'aplicació i instal·lem les dependències que ens demana:

   ./w3af_console

Si observem el directori creat veurem que hi ha dos executables *w3af_console* i *w3af_gui*, nosaltres per al projecte només necessitarem el primer ja que es el que s'executa en línia de comandes i interactua amb la web.

Podem comprovar que tot està bé comprovant la versió de l'aplicació amb la següent comanda:

```
borja@sw_test:/var/www/uoc_project/w3af$ ./w3af_console --version
w3af - Web Application Attack and Audit Framework
Version: 1.6.51
Revision: 9052e89be9 - 22 Apr 2015 13:07
Branch: master
Local changes: Yes
Author: Andres Riancho and the w3af team.
```

5.4 – Instal·lació Pompem

Igual que en el punt anterior aquesta eina s'instal·la des del seu repositori a *github* (https://github.com/rfunix/Pompem). La versió instal·lada es la de desenvolupament ja que versions anteriors donaven problemes amb la versió instal·lada de *Python*. Els passos que seguits han segut aquestos:

1. Descarreguem l'aplicació des del repositori:

   git clone https://github.com/rfunix/Pompem.git Pompem-dev

2. Movem el directori a la ubicació desitjada del projecte

   mv Pompem-dev /var/www/uoc_project/

3. Creem l'entorn virtual:

   virtualenv .env

4. Activem l'entorn virtual

   source .env/bin/activate

5. Instal·lem les dependències

   cd Pompem-dev

   pip install -r requirements.txt

Podem comprovar que estem a l'última versió executant la següent comanda:

```
borja@sw_test:/var/www/uoc_project/Pompem-dev$ python pompem.py --update
updating Pompem to the latest development version from the GitHub repository
[21:55:35] [INFO] update in progress
.
```

5.5 – Instal·lació Damn Vulnerable Web App

Aquesta aplicació està basada en *PHP* i *MySQL* i consisteix en una plataforma web amb vulnerabilitats que ens permet fer test de seguretat. La instal·lació no és molt complicada:

1. Descarreguem l'aplicació de la web:

   wget -c https://github.com/RandomStorm/DVWA/archive/v1.0.8.zip

2. Descomprimim el fitxer:

   unzip v1.0.9.zip

3. Movem el directori que ens ha creat al directori base del projecte i canviem els permisos:

   mv dvwa /var/www/uoc_project/

   chmod -R 755 dvwa

4. Per últim s'ha de crear la base de dades i editar el fitxer '*config.inc.php*' per tal de definir el usuari i contrasenya de la base de dades.

Una vegada instal·lat podem accedir amb el navegador web i si tot ha anat bé veurem una pantalla com aquesta que ens demanarà l'usuari i la contrasenya per a accedir:

Capítol 6 – Conclusions

Tal i com ha quedat l'eina desenvolupada i respecte a l'estat inicial del projecte podem dir que s'han complert els principals objectius marcats a l'inici. Els resultats aconseguits són prou satisfactoris ja que s'ha aconseguit desenvolupar una eina funcional que a partir d'una interfície molt simplista permet realitzar un escaneig de vulnerabilitats cap a una aplicació web on els resultats poden ser interpretats per persones sense alts coneixements de seguretat.

També cal remarcar que encara que l'eina es funcional es podria considerar que encara es una prova de concepte ja que per a poder convertir-la en una eina robusta requeriria una serie de millores i ampliacions.

Una vegada finalitzat el projecte, i després d'haver estudiat els diferents tipus de vulnerabilitats comuns en aplicacions web i les diferents eines existents al mercat de detecció d'aquestes, crec que les eines triades donen molt de joc a l'hora de fer els diferents tipus de test de vulnerabilitats. Per una part les aplicacions *Whatweb* i *W3af* es poden considerar eines madures que estan en constant desenvolupament i on es van incorporant les noves tècniques que van apareixent en el dia a dia de la seguretat, però per un altra part l'aplicació *Pompem*, encara que ajuda en la cerca automatitzada de "*exploits*" existent, encara es podria millorar ja que els resultats obtinguts no sempre s'ajusten bé a les cerques realitzades.

Per últim m'agradaria remarcar que l'eina desenvolupada s'ha de considerar com un punt de partida cap a una eina més robusta i eficient on es podrien implementar algunes millores i noves implementacions ja que les eines triades per al desenvolupament són molt potents i ens donen aquesta oportunitat.

Alguns punts interessants que es podrien treballar per a un desenvolupament futur poden ser els següents:

- Primer de tot, degut a que no soc un programador expert en aquestes tecnologies, s'hauria de repassar el codi per tal de crear un codi més eficient i un control d'errors exhaustiu per tal de controlar les possibles excepcions.
- Es podrien afegir també millores en la implementació de la interfície gràfica per tal de guanyar eficiència i fer que la resposta als escanejos siga més ràpida.
- Tal i com està l'eina actualment es podria introduir un control d'usuaris mitjançant autenticació i que els usuaris pugin tindre un espai on poder gestionar els escanejos realitzats.
- També es podria definir una funció per tal que els reports generat s'envien per correu electrònic, amb opció d'enviament xifrat, a l'usuari.
- Un altra opció interessant seria la incorporació de definir varies adreces a escanejar i que es puga executar de forma automàtica sense la intervenció de l'usuari.

- Un altra opció podria ser la creació de diferents perfils d'escanejos, per exemple per tipus de vulnerabilitats, on l'usuari puga triar el perfil que més li interesse en cada moment ja que no sempre podem estar interessats en cercar totes les vulnerabilitats definides.
- Implementar un sistema de planificació de les execucions on es puga definir el dia i la hora en el que es vol realitzar el escaneig. Hi ha escanejos que són molt agressius i poden afectar al rendiment de l'aplicació web que estem estudiant, es per això que aquesta funcionalitat podria ser d'utilitat quan estem provant aplicacions que ja estan en producció.
- Introducció d'un *proxy* per tal de poder fer els escanejos de forma més anònima.
- Poder realitzar exportacions dels resultats a diferents formats, *pdf*, *csv*, *xml*, etc.

Annex – I. Perfil crear per a l'eina W3af

[grep.strange_headers]

[grep.svn_users]

[crawl.web_spider]

[grep.blank_body]

[misc-settings]

maxThreads = 15

form_fuzzing_mode = tmb

fuzzed_files_extension = gif

autoDependencies = True

demo = False

max_discovery_time = 120

fuzzable_headers =

showProgressBar = True

fuzzCookie = False

fuzz_form_files = True

fuzz_url_filenames = False

fuzz_url_parts = False

stop_on_first_exception = False

[grep.strange_http_codes]

[grep.oracle]

[output.console]

verbose = False

[grep.motw]

[grep.objects]

[grep.strange_parameters]

[grep.error_pages]

[grep.file_upload]

[grep.credit_cards]

[grep.error_500]

[grep.feeds]

[audit.xss]

persistent_xss = True

[grep.hash_analysis]

[profile]

name = project

description = Profile created for the security project.

[grep.lang]

[grep.form_autocomplete]

[grep.ssn]

[grep.dot_net_event_validation]

[grep.meta_tags]

[audit.sqli]

[grep.strange_reason]

[grep.http_in_body]

[grep.dom_xss]

[grep.directory_indexing]

[grep.code_disclosure]

[grep.analyze_cookies]

[grep.get_emails]

only_target_domain = True

[grep.private_ip]

[grep.path_disclosure]

[http-settings]

maxFileSize = 400000

max_http_retries = 2

ignore_session_cookies = False

timeout = 0

user_agent = w3af.org

[grep.wsdl_greper]

[grep.http_auth_detect]

[grep.symfony]

override = False

[grep.user_defined_regex]

[grep.click_jacking]

Annex – II. Codi del fitxer 'scan.php'

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <title>Application Security Scan Report</title>

    <link rel="stylesheet" type="text/css" href="bootstrap-3.3.2.min.css">

    <style>
        .table {
            table-layout:fixed;
        }

        .table td {
          white-space: nowrap;
          overflow: hidden;
          text-overflow: ellipsis;
        }
    </style>
</head>

<?php
   $url = htmlspecialchars($_POST['urlSearch']);
   $purl = parse_url($url);
   $host = $purl['host'];
   if (file_exists('reports/ww_' . $host)) {
      unlink('reports/ww_' . $host);
   }
   exec('./whatweb/whatweb ' . $url . ' -p whatweb/plugins-enabled' . ' --log-xml reports/' . 'ww_' . $host);
```

```
$xml_ww = simplexml_load_file('reports/ww_' . $host) or die("Error: Cannot create object");

if (file_exists('reports/w3_' . $host)) {
  unlink('reports/w3_' . $host);
}
exec('python scan_test.py' . ' ' . $url);
$xml = simplexml_load_file('reports/w3_' . $host) or die("Error: Cannot create object");

# PLUGINS Whatweb
for ( $i = 0; $i < 5 ; $i++) {
  if ( $xml_ww->target->plugin[$i]->name == "Country" )  {
    $plugins[0] = array($xml_ww->target->plugin[$i]->string, $xml_ww->target->plugin[$i]->module);
  }
  if ( $xml_ww->target->plugin[$i]->name == "IP" )  {
    $plugins[1] = $xml_ww->target->plugin[$i]->string;
  }
  if ( $xml_ww->target->plugin[$i]->name == "HTTPServer" )  {
    $plugins[2] = array($xml_ww->target->plugin[$i]->os, $xml_ww->target->plugin[$i]->string);
  }
  if ( $xml_ww->target->plugin[$i]->name == "X-Powered-By" )  {
    $plugins[3] = $xml_ww->target->plugin[$i]->string;
  }
  if ( $xml_ww->target->plugin[$i]->name == "Drupal" )  {
    $plugins[4] = $xml_ww->target->plugin[$i]->name;
  }
  if ( $xml_ww->target->plugin[$i]->name == "WordPress" )  {
    $plugins[4] = $xml_ww->target->plugin[$i]->name;
  }
  if ( $xml_ww->target->plugin[$i]->name == "Joomla" )  {
    $plugins[4] = $xml_ww->target->plugin[$i]->name;
  }
  if ( $xml_ww->target->plugin[$i]->name == "Blogger" )  {
    $plugins[4] = $xml_ww->target->plugin[$i]->name;
```

```
}
if ( $xml_ww->target->plugin[$i]->name == "Magento" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "PrestaShop" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "Zope" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "OSCommerce" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "Plone" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "TYPO3" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "Zen-Cart" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "Mambo" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "XOOPS" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "dotclear" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
}
if ( $xml_ww->target->plugin[$i]->name == "Dspace" )  {
  $plugins[4] = $xml_ww->target->plugin[$i]->name;
```

```
    }
    if ( $xml_ww->target->plugin[$i]->name == "OpenCms" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "Moodle" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "ModxCMS" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "Symfony" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "Dokeos" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "Claroline" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }
    if ( $xml_ww->target->plugin[$i]->name == "Chamilo" )  {
      $plugins[4] = $xml_ww->target->plugin[$i]->name;
    }


  }
?>
<body>
        <div class="container">
                <div class="thumbnail">
                        <div class="row">
                                <div class="col-md-12">
                                        <h1>
                                                <p class="text-center">Security Scan Report</p>
                                        </h1>
```

```
                </div>
            </div>

                <div class="row">
                    <div class="col-md-1"></div>
                    <div class="col-md-4">
                    <h4>Scan date</h4>
                    <p><?php echo $xml->attributes()->{'start-long'};?> </p>
                    </div>

                    <div class="col-md-3">
                    <h4>Configured target URL</h4>
                        <ul>
                                <?php echo $xml_ww->target->uri;?>
                        </ul>
                    </div>

                    <div class="col-md-4">
                <h4>URL Information</h4>
                    <ul>
                                <li> <?php echo 'Country: ' . $plugins[0][0] . '(' .
$plugins[0][1] . ')';?>
                                <li> <?php echo 'IP: ' . $plugins[1];?> </li>
                    </ul>
              </div>

                    <div class="col-md-1"></div>
                </div>
            </div>

            <?php

            echo "<div class='thumbnail'>";
```

```
                    echo "<div class='row'>";
                        echo '<div class="col-md-2"><img src="images/information.png" /></div>';
                        echo '<div class="col-md-8"><h2 class="text-center">Server Fingerprint</h2></div>';
                        echo '<div class="col-md-2"><b><h3 class="text-success" style="color:#1999D6">Information</h3></b></div>';
                    echo "</div>";
                    echo "<div class='row'>";
                        if ( !empty($plugins[2][0]) ) {
                            $pompem = explode(" ", $plugins[2][0]);
                            echo '<div class="row"><div class="col-md-1"></div><div class="col-md-4"><p><span style="font-size:18px; font-family:inherit; display:inline;">Server OS: </span>' . $plugins[2][0] . '</p></div><div class="col-md-3"><form action="scan_pompem.php" method="POST"><button name="pompem" type="submit" value="' . $pompem[0] . '"> + Search for eXploits</button></form></div><div class="col-md-4"></div>';
                    }
                    if ( !empty($plugins[2][1]) ) {
                            echo '<div class="row"><div class="col-md-1"></div><div class="col-md-4"><p><span style="font-size:18px; font-family:inherit">Web Server: </span>' . $plugins[2][1] . '</p></div><div class="col-md-3"><form action="scan_pompem.php" method="POST"><button name="pompem" type="submit" value="' . $pompem[0] . '"> + Search for eXploits</button></form></div><div class="col-md-4"></div>';
                    }
                    if ( !empty($plugins[3]) ) {
                            echo '<div class="row"><div class="col-md-1"></div><div class="col-md-4"><p><span style="font-size:18px; font-family:inherit">PHP: </span>' . $plugins[3] . '</p></div><div class="col-md-3"><form action="scan_pompem.php" method="POST"><button name="pompem" type="submit" value="' . $pompem[0] . '"> + Search for eXploits</button></form></div><div class="col-md-4"></div>';
                    if ( !empty($plugins[4]) ) {
                             echo '<div class="row"><div class="col-md-1"></div><div class="col-md-4"><p><span style="font-size:18px; font-family:inherit">CMS: </span>' . $plugins[4] . '</p></div><div class="col-md-3"><form action="scan_pompem.php" method="POST"><button name="pompem" type="submit" value="' . $pompem[0] . '"> + Search for eXploits</button></form></div><div class="col-md-4"></div>';
                    }
                        echo "</div>";
                echo "</div>";

                $resultado = count($xml->vulnerability);
                //echo "<p>" . $resultado . "</p>";
```

```
for ( $i = 0; $i < $resultado ; $i++) {


echo "<div class='thumbnail'>";
echo "<div class='row'>";
                $severity = $xml->vulnerability[$i]['severity'];
                if ( $severity == "Low" ) {
                        $img = "images/low.png";
                        $color = "#3C763D";
                } elseif ( $severity == "Information" ) {
                        $img = "images/information.png";
                        $color = "#1999D6";
                } elseif ( $severity == "Medium" ) {
                        $img = "images/medium.png";
                        $color = "#FBAC23";
                } else {
                        $img = "images/high.png";
                        $color = "#22438";
                }
        echo "<div class='col-md-2'><img src=" . $img . " /></div>";
        echo "<div class='col-md-8'><h2 class='text-center'>" . $xml->vulnerability[$i]['name'] .
"</h2></div>";
        echo "<div class='col-md-2'><b><h3 class='text-success' style='color:" . $color . "'>" . $xml-
>vulnerability[$i]['severity'] . "</h3></b></div>";
echo "</div>";
echo "<div class='row'>";
        echo "<div class='col-md-1'></div>";
        echo "<div class='col-md-10'>";
                echo "<h4>Summary</h4>";
                echo "<p>" . $xml->vulnerability[$i]->description . "</p>";
        echo "<br />";
        echo "<h4>Description</h4>";
                        echo "<p>" . $xml->vulnerability[$i]->{'long-description'} . "</p>";
        echo "</div>";
        echo "<div class='col-md-1'></div>";
```

```
                echo "</div>";
                echo "<div class='row'>";
                        echo "<div class='col-md-1'></div>";
                        echo "<div class='col-md-10'>";
                                echo "<ul>";
                                echo "<li>Vulnerable URL: <a href=" . $xml->vulnerability[$i]['url'] . ">" . $xml->vulnerability[$i]['url'] . "</a></li>";
                                echo "<li>Vulnerable Parameter: <code>" . $xml->vulnerability[$i]['var'] . "</code></li>";
                                echo "</ul>";
                        echo "</div>";
                        echo "<div class='col-md-1'></div>";
                echo "</div>";
                 echo "<div class='row'>";
                 echo "<div class='col-md-1'></div>";
                 echo "<div class='col-md-10'>";
                                if ( $xml->vulnerability[$i]->{'fix-guidance'} ) {
                           echo "<h4>Fix</h4>";
                              echo $xml->vulnerability[$i]->{'fix-guidance'} . "<br>";
                  echo "<h4>References</h4>";
                  echo "<ul>";
                        echo "<li> <a href=" . $xml->vulnerability[$i]->references->reference["url"] . ">" . $xml->vulnerability[$i]->references->reference["title"] . "</a><br></li>";
                  echo "</ul>";
                                }
                 echo "</div>";
                 echo "<div class='col-md-1'></div>";
        echo "</div>";
                echo "</div>";
                }
                ?>
        </div>
</body>
</html>
```

Referències

- [1] MorningStar Security. Web i documentació de l'eina Whatweb. http://www.morningstarsecurity.com/research/whatweb
- [2] W3af – Web Application Attack and Audit Framework. Web i documentació de l'eina. http://w3af.org/
- [3]Pompem – Exploit Finder. Repositori de github on podem descarregar l'eina i alguna documentació. https://github.com/rfunix/Pompem
- [4]The Open Web Application Security Project. Lloc on podem trobar molta informació referent a vulnerabilitats de tipus web. https://www.owasp.org
- [5] Open Source Vulnerability Database. http://osvdb.org/
- [6] CVE – Common Vulnerabilities and Exposure Database. http://cve.mitre.org/
- [7] Linux Containers. Plataforma de virtualització. https://linuxcontainers.org/
- [8] Damn Vulnerable Web Application. http://www.dvwa.co.uk/