



WORDLISTS FOR PENTESTER

Tabla de contenido

| | |
|---|----------|
| Abstracto | 3 |
| ¿Qué son las listas de palabras? | 4 |
| Listas de palabras en Kali Linux | 4 |
| Listas de palabras Dirb | 5 |
| De la palabra Rockyou | 6 |
| Listas de palabras de Wfuzz | 6 |
| Listas de palabras en línea | 9 |
| Listas de palabras de Github | 9 |
| Seclistas | 10 |
| Listas de palabras de nodos de activos | 12 |
| Listas de palabras de Packetstorm | 12 |
| Limpieza de listas de palabras | 13 |
| Elaboración de listas de palabras | 14 |
| Cewl | 14 |
| Elaboración de lista de palabras: Crunch | 15 |
| Elaboración de listas de palabras: Cupp | decisión |
| Elaboración de listas de palabras: Pydictor | 17 |
| Elaboración de listas de palabras: Bopscrk | 17 |
| Elaboración de listas de palabras: BEWGOR | 19 |
| Fusionar listas de palabras: DyMerge | 21 |
| Elaboración de listas de palabras: mentalista | 22 |
| Reglas de Hashcat/John | 23 |
| Conclusión | 24 |
| Referencias | 24 |
| Sobre nosotros | 25 |

Abstracto

Un Pentester es tan bueno como sus herramientas y cuando se trata de descifrar la contraseña, enfatizar los paneles de autenticación o incluso un simple **directorio Bruteforce**, todo se desglosa hasta las listas de palabras que usa. Hoy entenderemos las listas de palabras, buscaremos algunas buenas listas de palabras, ejecutaremos algunas herramientas para administrar las listas de palabras y mucho más.

Desde que comenzó la evolución de los Penetration Testers, una de las cosas que vemos constantemente es que el atacante descifra la contraseña del objetivo y entra.

Bueno, en la mayoría de las representaciones de ataques en películas y series a menudo se muestra esta situación en detalle, ya que es el ataque más simple de representar. No importa cuán simple sea descifrar contraseñas o realizar **Credential Stuffing** alguna vez estuvo prohibido en las aplicaciones web. Hoy en día, de alguna manera tenemos un poco de control sobre ellos con el uso de **CAPTCHA** o limitación de velocidad, pero aún así son uno de los ataques efectivos. El alma de tales ataques es la lista de palabras.

¿Qué son las listas de palabras?

Una lista de palabras es un archivo (un archivo de texto en la mayoría de los casos, pero no limitado a él) que contiene un conjunto de valores que el atacante debe proporcionar para probar un mecanismo. Esto es un poco complejo, diluyámoslo un poco para entenderlo mejor. Siempre que un atacante se enfrenta a un mecanismo de autenticación, puede intentar solucionarlo, pero si eso no es posible, entonces el atacante debe probar algunas credenciales conocidas en el mecanismo de autenticación para intentar adivinar. Esta lista de credenciales conocidas es una lista de palabras. Y en lugar de ingresar manualmente los valores uno por uno, el atacante usa una herramienta o script para automatizar este proceso. De manera similar, en el caso de descifrar valores hash, la herramienta usa las listas de palabras y codifica las entradas de las listas de palabras en el mismo hash y luego usa una función de comparación de cadenas para hacer coincidir los hashes. Si se encuentra una coincidencia, el hash se considera descifrado. Se puede observar que la importancia de la lista de palabras es primordial en el mundo de la seguridad cibernética.

Listas de palabras en Kali Linux

Dado que Kali Linux fue diseñado especialmente para realizar pruebas de penetración, está lleno de varios tipos de listas de palabras. Esto se debe a las diversas herramientas que están presentes en Kali Linux para realizar ataques de fuerza bruta en inicios de sesión, directorios, etc. Repasemos algunas de las listas de palabras del enorme arsenal de listas de palabras que contiene Kali Linux.

Las listas de palabras se encuentran dentro del directorio `/usr/share`. Aquí, tenemos el directorio `dirb` para las listas de palabras que se usarán mientras se usa la herramienta `dirb` para realizar Directory Bruteforce. Luego tenemos el `dirbuster` que es una herramienta similar que también realiza Directory Bruteforce pero con algunas opciones adicionales. Luego tenemos un directorio `fern-wifi` que ayuda a romper las autenticaciones Wi-Fi. Luego tenemos Metasploit que usa listas de palabras para casi todo.

Luego hay una lista de palabras de `nmap` que contiene las que se pueden usar mientras se escanean algunos servicios específicos. Luego tenemos al Rockstar de las listas de palabras: `rockyou`. Está comprimido de forma predeterminada y tendrás que extraerlo antes de usarlo. Es muy grande con 1,44,42,062 valores que podrían ser contraseñas para muchas cuentas de usuarios en Internet. Por fin, tenemos el directorio `wfuzz` que tiene las listas de palabras que se pueden usar combinadas con `wfuzz`.



Ubicación: `/usr/share/wordlists`

```
(root@kali)-[/usr/share/wordlists/metasploit]
# cd /usr/share/wordlists

(root@kali)-[/usr/share/wordlists]
# ls -la
total 136660
drwxr-xr-x  2 root root    4096 Feb 26 14:57 .
drwxr-xr-x 351 root root  12288 Mar 11 12:42 ..
lrwxrwxrwx  1 root root     25 Nov 17 07:24 dirb -> /usr/share/dirb/wordlists
lrwxrwxrwx  1 root root     30 Nov 17 07:24 dirbuster -> /usr/share/dirbuster/wordlists
lrwxrwxrwx  1 root root     41 Nov 17 07:24 fasttrack.txt -> /usr/share/set/src/fasttrack/wordlist.txt
lrwxrwxrwx  1 root root     45 Nov 17 07:24 fern-wifi -> /usr/share/fern-wifi-cracker/extras/wordlists
lrwxrwxrwx  1 root root     46 Nov 17 07:24 metasploit -> /usr/share/metasploit-framework/data/wordlists
lrwxrwxrwx  1 root root     41 Nov 17 07:24 nmap.lst -> /usr/share/nmap/nselib/data/passwords.lst
-rw-r--r--  1 root root 139921507 Feb 26 14:50 rockyou.txt
lrwxrwxrwx  1 root root     25 Nov 17 07:24 wfuzz -> /usr/share/wfuzz/wordlist
```

Listas de palabras Dirb

Para observar más de cerca uno de los directorios, usamos el comando de árbol para enumerar todas las listas de palabras dentro del directorio dirb. Aquí tenemos diferentes listas de palabras que difieren en tamaño e idiomas. También hay una lista de palabras de extensiones para que el atacante pueda usar ese directorio para realizar un Directory Bruteforce. También hay algunas listas de palabras específicas de aplicaciones, como apache.txt o sharepoint.txt.



Ubicación: /usr/share/wordlists/dirb

```
(root@kali)-[/usr/share/wordlists]
# cd dirb

(root@kali)-[/usr/share/wordlists/dirb]
# tree
.
├── big.txt
├── catala.txt
├── common.txt
├── euskera.txt
├── extensions_common.txt
├── indexes.txt
├── mutations_common.txt
├── others
│   ├── best1050.txt
│   ├── best110.txt
│   ├── best15.txt
│   └── names.txt
├── small.txt
├── spanish.txt
├── stress
│   ├── alphanum_case_extra.txt
│   ├── alphanum_case.txt
│   ├── char.txt
│   ├── doble_uri_hex.txt
│   ├── test_ext.txt
│   ├── unicode.txt
│   └── uri_hex.txt
├── vulns
│   ├── apache.txt
│   ├── axis.txt
│   ├── cgis.txt
│   ├── coldfusion.txt
│   ├── domino.txt
│   ├── fatwire_pagenames.txt
│   ├── fatwire.txt
│   ├── frontpage.txt
│   └── hnsmb.txt
```


De la palabra Rockyou

Rockyou.txt es un conjunto de contraseñas comprometidas del desarrollador de aplicaciones de redes sociales también conocido como RockYou. Desarrolló widgets para la aplicación Myspace. En diciembre de 2009, la empresa experimentó una violación de datos que dejó expuestas a más de 32 millones de cuentas de usuarios. Se debió principalmente a la política de la empresa de almacenar las contraseñas en texto sin cifrar.



Ubicación: /usr/share/wordlists

Cuando inicie Kali Linux por primera vez, se comprimirá en un archivo gz. Para descomprimir ejecute el siguiente comando. Se descomprimirá y estará listo para usarse en cualquier tipo de ataque que desee.

```
gzip -d /usr/share/wordlists/rockyou.txt.gz
```

```
(root@kali)-[/usr/share/wordlists]
# ls -la
-rw-r--r-- 1 root root 139921507 Feb 26 14:50 rockyou.txt
```

Listas de palabras de Wfuzz

La herramienta Wfuzz fue desarrollada para realizar ataques de fuerza bruta en aplicaciones web. También se puede utilizar para enumerar aplicaciones web. Puede enumerar directorios, archivos y scripts, etc. También puede cambiar la solicitud de GET a POST. Esto es útil en varios escenarios, como comprobar si hay inyecciones SQL. Viene con un conjunto de listas de palabras predefinidas. Estas listas de palabras están diseñadas para usarse con wfuzz pero pueden usarse en cualquier lugar que desee. Las listas de palabras se dividen en categorías como general, inyecciones, estrés, vulnerabilidades, servicios web y otras.



Ubicación: /usr/share/wordlists/wfuzz

```
(root@kali)-[/usr/share/wordlists]
# tree

.
├── dirb → /usr/share/dirb/wordlists
├── dirbuster → /usr/share/dirbuster/wordlists
├── fasttrack.txt → /usr/share/set/src/fasttrack/wordlist.txt
├── fern-wifi → /usr/share/fern-wifi-cracker/extras/wordlists
├── metasploit → /usr/share/metasploit-framework/data/wordlists
├── nmap.lst → /usr/share/nmap/nselib/data/passwords.lst
├── rockyou.txt
└── wfuzz → /usr/share/wfuzz/wordlist
```

5 directories, 3 files

```
(root@kali)-[/usr/share/wordlists]
# cd wfuzz

(root@kali)-[/usr/share/wordlists/wfuzz]
# tree

.
├── general
│   ├── admin-panels.txt
│   ├── big.txt
│   ├── catala.txt
│   ├── common.txt
│   ├── euskera.txt
│   ├── extensions_common.txt
│   ├── http_methods.txt
│   ├── medium.txt
│   ├── megabeast.txt
│   ├── mutations_common.txt
│   ├── spanish.txt
│   └── test.txt
├── Injections
│   ├── All_attack.txt
│   ├── bad_chars.txt
│   ├── SQL.txt
│   ├── Traversal.txt
│   ├── XML.txt
│   └── XSS.txt
├── others
│   ├── common_pass.txt
│   └── names.txt
├── stress
│   ├── alphanum_case_extra.txt
│   ├── alphanum_case.txt
│   ├── char.txt
│   ├── doble_uri_hex.txt
│   ├── test_ext.txt
│   └── uri_hex.txt
└── vulns
    ├── apache.txt
    ├── cgis.txt
    ├── coldfusion.txt
    ├── dirTraversal-nix.txt
    ├── dirTraversal.txt
    └── dirTraversal-win.txt
```

Al mirar el directorio de Inyecciones, vemos que tenemos un All_attack.txt que es una lista de palabras bastante genérica para probar inyecciones. Luego tenemos uno específico para SQL, Directory Traversal, XML, inyecciones XSS. Pasando al directorio general, vemos que tenemos el big.txt que comentamos en la sección Dirb. Tenemos common.txt que también es la lista de palabras predeterminada en muchas herramientas debido a su pequeño tamaño. Luego tenemos extensions_common.txt que contiene unas 25 extensiones que podrían enumerarse como algunos archivos que pueden considerarse frutos al alcance de la mano. Luego tenemos la lista de palabras http_methods.txt. Contiene los métodos HTTP como POST, GET, PUT, etc. Se pueden usar mientras se prueba si la aplicación de destino tiene habilitado algún método mal configurado o si se olvidó de deshabilitarlo a nivel de aplicación y servidor. mutaciones_common.txt también contiene un montón de extensiones poco comunes que podrían conducir a enumeraciones de artefactos raros.

```
(root@kali)-[/usr/share/wordlists/wfuzz/Injections]
# cat XSS.txt
"><script>"
<script>alert("WXSS")</script>
<<script>alert("WXSS");//<</script>
<script>alert(document.cookie)</script>
'><script>alert(document.cookie)</script>
'><script>alert(document.cookie);</script>
\";alert('XSS');//
%3cscript%3ealert("WXSS");%3c/script%3e
%3cscript%3ealert(document.cookie);%3c%2fscript%3e
%3Cscript%3Ealert(%22X%20SS%22);%3C/script%3E
&lt;script&gt;alert(document.cookie);</script>
&lt;script&gt;alert(document.cookie);&lt;script&gt;alert
<xss><script>alert('WXSS')</script></vulnerable>
<IMG%20SRC='javascript:alert(document.cookie)'\>
<IMG%20SRC="javascript:alert('WXSS');">
<IMG%20SRC="javascript:alert('WXSS')">
<IMG%20SRC=javascript:alert('WXSS')>
<IMG%20SRC=JaVaScRiPt:alert('WXSS')>
<IMG%20SRC=javascript:alert(&quot;WXSS&quot;)>
<IMG%20SRC=`javascript:alert("WXSS")`>
<IMG%20""><SCRIPT>alert("WXSS")</SCRIPT>">
<IMG%20SRC=javascript:alert(String.fromCharCode(88,83,83))>
<IMG%20SRC='javasc      ript:alert(document.cookie)'\>
<IMG%20SRC="jav ascript:alert('WXSS');">
<IMG%20SRC="jav&#x09;ascript:alert('WXSS');">
<IMG%20SRC="jav&#x0A;ascript:alert('WXSS');">
<IMG%20SRC="jav&#x0D;ascript:alert('WXSS');">
<IMG%20SRC="%20&#14;%20javascript:alert('WXSS');">
<IMG%20DYNsrc="javascript:alert('WXSS')">
<IMG%20LOWsrc="javascript:alert('WXSS')">
```

Luego tenemos la lista de palabras en español.txt para las palabras/nombres/contraseñas en español, como lo habrá adivinado. El otro directorio contiene las contraseñas y nombres comunes que se pueden usar para extraer nombres de usuario o contraseñas en algún formulario de olvido de contraseña donde responde con mensajes de que el usuario existe o no existe. Pasemos al directorio de estrés.

Contiene una lista de palabras diseñada para poner a prueba el mecanismo. Contiene listas de palabras que contienen alfabetos, números o caracteres especiales y códigos hexadecimales para los mismos. Luego tenemos el directorio vulns, que contiene listas de palabras creadas especialmente para probar una vulnerabilidad particular.

Tenemos la lista de palabras de apache, la lista de palabras CGI, la lista de palabras de directorio, la lista de palabras de iis,

Lista de palabras de Oracle9, lista de palabras de SharePoint, lista de palabras de Tomcat y muchas más. Utilice estas listas de palabras en un escenario específico en el que esté confirmado sobre el marco y la información de versiones y simplemente úselas para apuntar a un punto de entrada en particular.

Listas de palabras en línea

Listas de palabras de Github

Conocimos sobre la enorme colección que contiene Kali Linux. Pero a veces tienden a no ser tan recientes como requerimos. Esto puede suceder en un escenario en el que se haya descubierto un nuevo día 0. No habrá ninguna entrada en esos diccionarios. Aquí es donde podemos volvernos locos buscando en Internet, pero es mucho y lleva más tiempo. Aquí es donde podemos husmear en GitHub, ya que muchas personas podrían crear dicho diccionario. Por lo tanto, buscar en GitHub puede brindarle esos diccionarios nuevos y actualizados o puede ayudarlo a encontrar ese diccionario específico que necesita para definir un marco específico.



Ubicación: /usr/share/wordlists/wfuzz




Enlace: [Listas de palabras de GitHub](#)

The screenshot shows the GitHub search interface with the query 'wordlists'. The search results are sorted by 'Best match' and show 2,286 repository results. The left sidebar displays navigation options like Repositories (2K), Code, Commits (73K), Issues (4K), Discussions (Beta), Packages (2), Marketplace (0), Topics (7), Wikis (566), and Users (3). Below this is a 'Languages' section with counts for Python (775), Shell (131), Java (113), JavaScript (77), C (57), C++ (54), Perl (43), Go (41), Ruby (35), and PHP (33). The main content area lists several repositories:

- berzerk0/Probable-Wordlists**: Version 2 is live! **Wordlists** sorted by probability originally created for password generation and testing - make sure...
Tags: dictionary, password, wordlist, password-strength, password-safety, dictionary-attack
6.9k stars, CC-BY-SA-4.0 license, Updated on Dec 4, 2019
- xajkep/wordlists**: Infosec **Wordlists**
Tags: security, dictionary, discovery, fuzzing, infosec, recon, payloads, wordlists
271 stars, Python, Updated on Jan 21, 2020
- kennyn510/wpa2-wordlists**: A collection of **wordlists** dictionaries for password cracking
Tags: dictionaries, wordlist, passwords, wireless-network, kali-linux
595 stars, Shell, Updated on May 14, 2020
- geovedi/indonesian-wordlist**: Indonesian **wordlist**
140 stars, Updated on Feb 18, 2016
- jeanphorn/wordlist**: Collection of some common **wordlists** such as RDP password, user name list, ssh password **wordlist** for brute force. IP C...
875 stars, Updated on Apr 2, 2016
- ZephFish/Wordlists**: Various Payload **wordlists**
186 stars, Updated on Aug 1, 2020

Seclistas

Las seclistas son una colección de múltiples tipos de listas de palabras que se pueden usar durante las pruebas de penetración o la evaluación de vulnerabilidad, todas recopiladas en un solo lugar. Estas listas de palabras pueden contener nombres de usuario, contraseñas, URL, patrones de datos confidenciales, cargas útiles de fuzzing, shells web, etc. Para instalar en Kali Linux, usaremos el comando apt seguido de Seclists como se muestra en la imagen a continuación.

 GitHub: [Seclists](#)

```
gzip -d /usr/share/wordlists/rockyou.txt.gz
```

```
(root@kali)~# apt install seclists
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are not
needed:
  galera-3 libcapstone3 libconfig-inifiles-perl libcrypto++6
  libpython3.8-dev libpython3.8-minimal libpython3.8-stdlib
  python3-atomicwrites python3.8 python3.8-dev python3.8-minimal
  xfce4-statusnotifier-plugin xfce4-weather-plugin
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
```

La instalación creará un directorio con el nombre de Seclists dentro de la ubicación /usr/share. Al revisar podemos ver las diferentes categorías de listas de palabras, como Discovery, Fuzzing, IOC, Misc, Contraseñas, Coincidencia de patrones, Cargas útiles, Nombres de usuario y Web-Shells.

```
(root@kali)-[~]
# cd /usr/share/seclists

(root@kali)-[/usr/share/seclists]
# ls -la
total 56
drwxr-xr-x 11 root root 4096 Mar 17 14:07 .
drwxr-xr-x 352 root root 12288 Mar 17 14:06 ..
drwxr-xr-x 9 root root 4096 Mar 17 14:06 Discovery
drwxr-xr-x 8 root root 4096 Mar 17 14:07 Fuzzing
drwxr-xr-x 2 root root 4096 Mar 17 14:07 IOCs
drwxr-xr-x 5 root root 4096 Mar 17 14:07 Miscellaneous
drwxr-xr-x 12 root root 4096 Mar 17 14:07 Passwords
drwxr-xr-x 3 root root 4096 Mar 17 14:07 Pattern-Matching
drwxr-xr-x 9 root root 4096 Mar 17 14:07 Payloads
-rw-r--r-- 1 root root 1953 Feb 11 16:59 README.md
drwxr-xr-x 4 root root 4096 Mar 17 14:07 Usernames
drwxr-xr-x 9 root root 4096 Mar 17 14:07 Web-Shells

(root@kali)-[/usr/share/seclists]
# cd Passwords

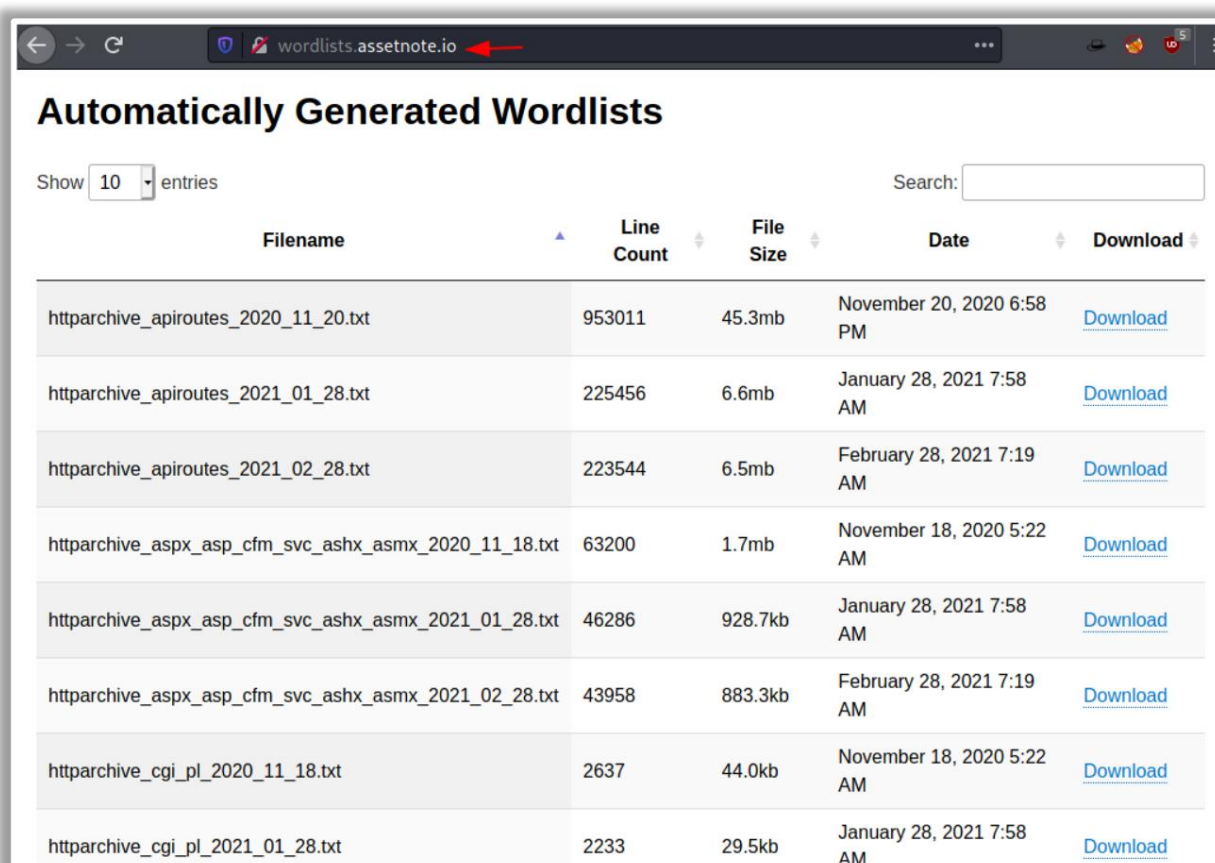
(root@kali)-[/usr/share/seclists/Passwords]
# tree
.
├── 2020-200_most_used_passwords.txt
├── BiblePass
│   ├── BiblePass_part01.txt
│   ├── BiblePass_part02.txt
│   ├── BiblePass_part03.txt
│   ├── BiblePass_part04.txt
│   ├── BiblePass_part05.txt
│   ├── BiblePass_part06.txt
│   ├── BiblePass_part07.txt
│   ├── BiblePass_part08.txt
│   ├── BiblePass_part09.txt
│   ├── BiblePass_part10.txt
│   ├── BiblePass_part11.txt
│   ├── BiblePass_part12.txt
│   ├── BiblePass_part13.txt
│   ├── BiblePass_part14.txt
│   ├── BiblePass_part15.txt
│   ├── BiblePass_part16.txt
│   └── BiblePass_part17.txt
├── bt4-password.txt
├── cirt-default-passwords.txt
├── clarkson-university-82.txt
├── Common-Credentials
│   ├── 100k-most-used-passwords-NCSC.txt
│   ├── 10k-most-common.txt
│   ├── 10-million-password-list-top-1000000.txt
│   ├── 10-million-password-list-top-100000.txt
│   ├── 10-million-password-list-top-10000.txt
│   ├── 10-million-password-list-top-1000.txt
│   ├── 10-million-password-list-top-100.txt
│   ├── 10-million-password-list-top-500.txt
│   ├── 500-worst-passwords.txt
│   └── best1050.txt
```

Listas de palabras de nodos de activos

Assetnode Wordlist publica una lista de palabras especialmente seleccionada para una amplia gama de áreas, como el descubrimiento de subdominios o el descubrimiento de artefactos especiales. La mejor parte es que se actualiza el día 28 de cada mes según su sitio web. Esta es la segunda mejor cosa que se ha lanzado desde los SecListas. Para descargar todas las listas de palabras a la vez, cualquiera puede usar el siguiente comando wget.

🔗 Sitio web: [Listas de palabras de Assetnote](https://wordlists-cdn.assetnote.io/)

```
wget -r --no-parent -R "index.html*" https://wordlists-cdn.assetnote.io/ -nH
```



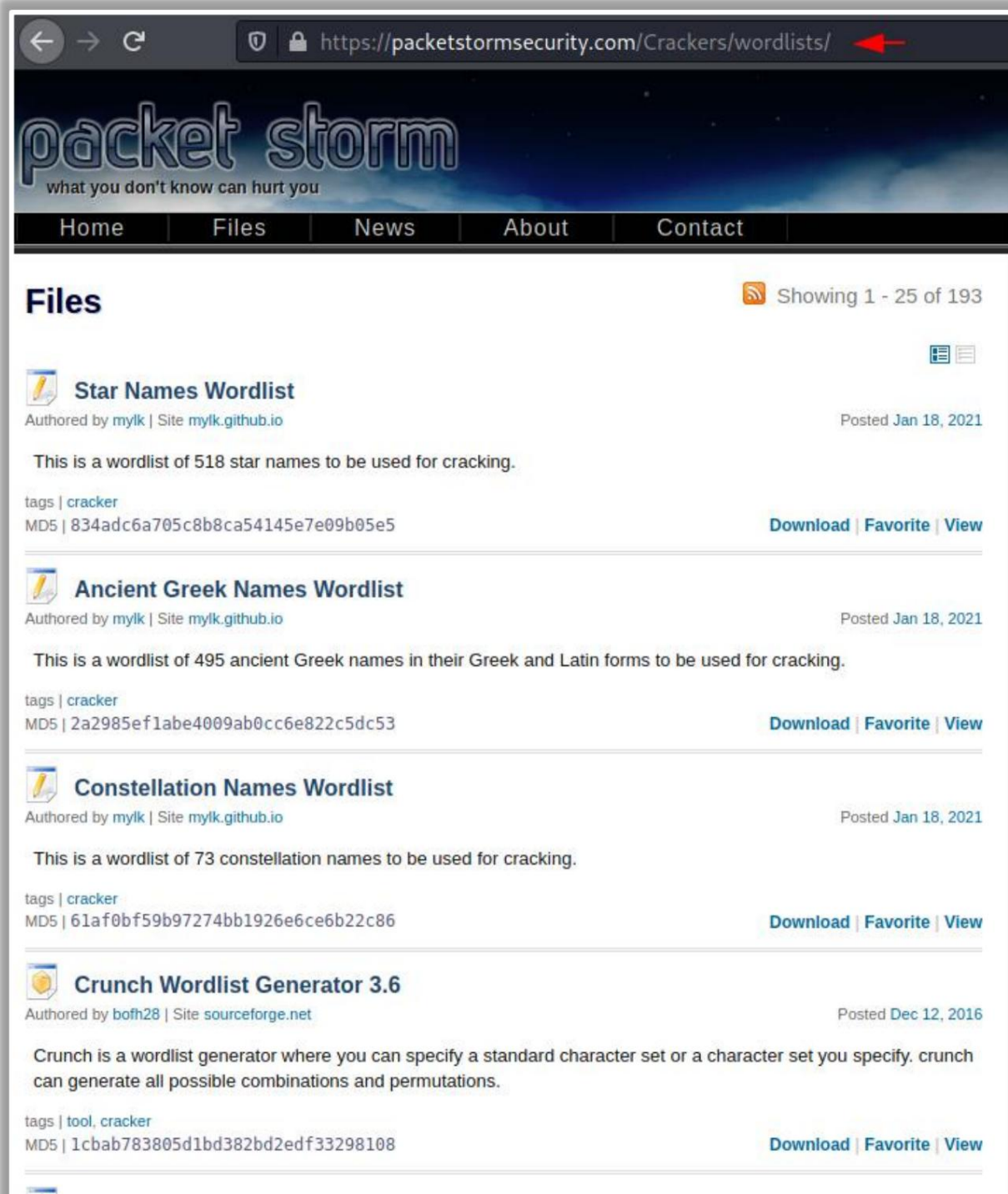
The screenshot shows a web browser displaying the 'Automatically Generated Wordlists' page on wordlists.assetnote.io. The page features a search bar and a table listing various wordlists. The table columns are Filename, Line Count, File Size, Date, and Download. Each row represents a different wordlist, such as 'httparchive_apiroutes_2020_11_20.txt', with its corresponding line count, file size, and download date. A 'Download' link is provided for each entry.

| Filename | Line Count | File Size | Date | Download |
|---|------------|-----------|---------------------------|--------------------------|
| httparchive_apiroutes_2020_11_20.txt | 953011 | 45.3mb | November 20, 2020 6:58 PM | Download |
| httparchive_apiroutes_2021_01_28.txt | 225456 | 6.6mb | January 28, 2021 7:58 AM | Download |
| httparchive_apiroutes_2021_02_28.txt | 223544 | 6.5mb | February 28, 2021 7:19 AM | Download |
| httparchive_aspx_asp_cfm_svc_ashx_asmx_2020_11_18.txt | 63200 | 1.7mb | November 18, 2020 5:22 AM | Download |
| httparchive_aspx_asp_cfm_svc_ashx_asmx_2021_01_28.txt | 46286 | 928.7kb | January 28, 2021 7:58 AM | Download |
| httparchive_aspx_asp_cfm_svc_ashx_asmx_2021_02_28.txt | 43958 | 883.3kb | February 28, 2021 7:19 AM | Download |
| httparchive_cgi_pl_2020_11_18.txt | 2637 | 44.0kb | November 18, 2020 5:22 AM | Download |
| httparchive_cgi_pl_2021_01_28.txt | 2233 | 29.5kb | January 28, 2021 7:58 AM | Download |

Listas de palabras de Packetstorm

Packet Storm Security es un sitio web de seguridad de la información que ofrece herramientas de seguridad informática, exploits y avisos de seguridad actuales e históricos. Está operado por un grupo de entusiastas de la seguridad que publican nueva información de seguridad y ofrecen herramientas con fines educativos y de prueba. Pero, para nuestra sorpresa, también publica listas de palabras. Cualquier usuario que haya creado una lista de palabras específica puede enviar su lista de palabras en su sitio web. Entonces, si está buscando una lista de palabras única, asegúrese de consultarla.

🔗 Enlace: [Listas de palabras de seguridad de Pack Storm](https://packetstormsecurity.com/wordlists/)



The screenshot shows the 'Files' section of the packetstormsecurity.com website. The browser address bar displays 'https://packetstormsecurity.com/Crackers/wordlists/'. The website header features the 'packet storm' logo with the tagline 'what you don't know can hurt you' and a navigation menu with links to Home, Files, News, About, and Contact. The 'Files' section lists four wordlists, each with a download icon, title, author information, description, tags, MD5 hash, and action links (Download, Favorite, View).

| File Name | Author | Site | Posted | Description | Tags | MD5 | Actions |
|-------------------------------|--------|-----------------|--------------|---|---------------|----------------------------------|----------------------------|
| Star Names Wordlist | mylk | mylk.github.io | Jan 18, 2021 | This is a wordlist of 518 star names to be used for cracking. | cracker | 834adc6a705c8b8ca54145e7e09b05e5 | Download Favorite View |
| Ancient Greek Names Wordlist | mylk | mylk.github.io | Jan 18, 2021 | This is a wordlist of 495 ancient Greek names in their Greek and Latin forms to be used for cracking. | cracker | 2a2985ef1abe4009ab0cc6e822c5dc53 | Download Favorite View |
| Constellation Names Wordlist | mylk | mylk.github.io | Jan 18, 2021 | This is a wordlist of 73 constellation names to be used for cracking. | cracker | 61af0bf59b97274bb1926e6ce6b22c86 | Download Favorite View |
| Crunch Wordlist Generator 3.6 | bofh28 | sourceforge.net | Dec 12, 2016 | Crunch is a wordlist generator where you can specify a standard character set or a character set you specify. crunch can generate all possible combinations and permutations. | tool, cracker | 1cbab783805d1bd382bd2edf33298108 | Download Favorite View |

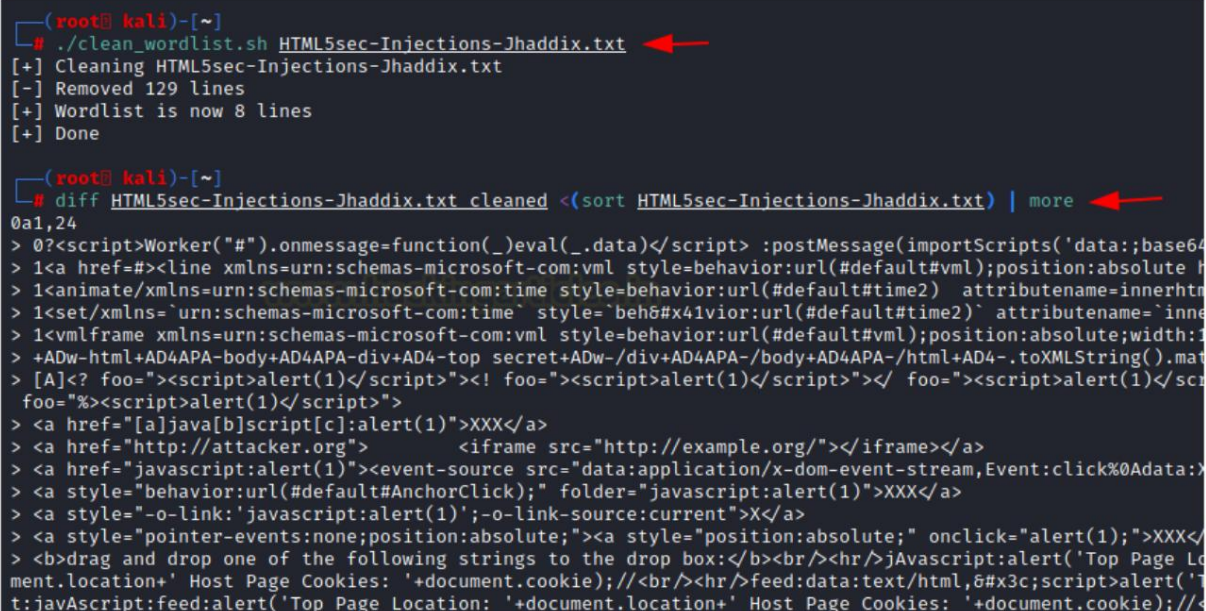
Limpieza de listas de palabras

Hasta ahora hemos visto múltiples listas de palabras que contienen miles y miles de entradas en su interior. Ahora bien, durante las pruebas de penetración en su servidor vulnerable o cualquier CTF, posiblemente esté bien ya que están diseñados para manejar este tipo de fuerza bruta, pero cuando llegamos al escenario de la vida real las cosas se complican un poco. Como en la vida real, ningún equipo de desarrollo o propietario le permitirá realizar mil tras mil listas de palabras con fuerza bruta. Esto puede obstaculizar la calidad del servicio a otros clientes. Entonces, deberíamos disminuir las entradas de la lista de palabras. se que suena

contraproducente pero no lo es. Las listas de palabras pueden contener algunas cargas útiles que pueden exceder los 100 caracteres o incluso ser demasiado específicas para extraer algo directamente. Luego tenemos algunas cargas útiles que son muy similares entre sí y si reemplazamos cualquiera de ellas, el resultado sigue siendo el mismo. Jon Barber creó un script que puede eliminar cartas ruidosas como ! (, %. Además, ordene la lista de palabras para que pueda ser más efectiva.

GitHub: [Lista de palabras limpias.sh](#)

```
./clean_wordlists.sh HTML5sec-Inyecciones-Jhaddix.txt
```



```
(root@kali)~# ./clean_wordlist.sh HTML5sec-Inyecciones-Jhaddix.txt
[+] Cleaning HTML5sec-Inyecciones-Jhaddix.txt
[-] Removed 129 lines
[+] Wordlist is now 8 lines
[+] Done

(root@kali)~# diff HTML5sec-Inyecciones-Jhaddix.txt cleaned <(sort HTML5sec-Inyecciones-Jhaddix.txt) | more
0a1,24
> 0?<script>Worker("#").onmessage=function(_){eval(_data)}</script> :postMessage(importScripts('data:;base64
> 1<a href=#><line xmlns=urn:schemas-microsoft-com:time style=behavior:url(#default#time2) attributename=innerhtml
> 1<animate/xmlns=urn:schemas-microsoft-com:time style=behavior:url(#default#time2) attributename=innerhtml
> 1<set/xmlns=urn:schemas-microsoft-com:time style=behavior:url(#default#time2) attributename=innerhtml
> 1<vmiframe xmlns=urn:schemas-microsoft-com:time style=behavior:url(#default#time2) attributename=innerhtml
> +ADw-html+AD4APA-body+AD4APA-div+AD4-top secret+ADw-/div+AD4APA-/body+AD4APA-/html+AD4-.toXMLString().mat
> [A]<? foo="><script>alert(1)</script>><! foo="><script>alert(1)</script>></ foo="><script>alert(1)</script>>
foo="%><script>alert(1)</script>>
> <a href="[a]java[b]script[c]:alert(1)">XXX</a>
> <a href="http://attacker.org"> <iframe src="http://example.org/"></iframe></a>
> <a href="javascript:alert(1)"><event-source src="data:application/x-dom-event-stream,Event:click%0Adata:
> <a style="behavior:url(#default#AnchorClick);" folder="javascript:alert(1)">XXX</a>
> <a style="-o-link:'javascript:alert(1)';-o-link-source:current">X</a>
> <a style="pointer-events:none;position:absolute;"><a style="position:absolute;" onclick="alert(1);">XXX</a>
> <bdrag and drop one of the following strings to the drop box:</b><br/><hr/>javascript:alert('Top Page Location:
ment.location+' Host Page Cookies: '+document.cookie);</b><br/><hr/>feed:data:text/html,8#x3c;script>alert('Top
t:javAscript:feed:alert('Top Page Location: '+document.location+' Host Page Cookies: '+document.cookie);</b>
```

Podemos verificar las líneas que se eliminaron de la lista de palabras de inyección HTML5 usando el comando diff como se muestra en la imagen de arriba.

```
diff HTML5sec-Inyecciones-Jhaddix.txt_cleaned < (ordenar HTML5sec-Inyecciones-Jhaddix.txt) |
más
```

Elaboración de listas de palabras

CeWL

CeWL es una aplicación Ruby que rastrea una URL determinada hasta una profundidad específica, siguiendo opcionalmente enlaces externos, y devuelve una lista de palabras que luego pueden usarse para descifradores de contraseñas como John the Ripper. CeWL también tiene una aplicación de línea de comandos asociada, FAB (Files Ya Bagged) que utiliza las mismas técnicas de extracción de metadatos para crear listas de autores/creadores a partir de los ya descargados. Aquí ejecutamos CeWL contra la URL de inicio y guardamos el resultado en una lista de palabras con el nombre dict.txt.

GitHub: [CeWL – Generador de listas de palabras personalizadas](#)

Más información: [Guía completa sobre la herramienta CeWL](#)

```
(root@kali)-[~]
# cewl https://www.ignitetechnologies.in/ -w dict.txt
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

(root@kali)-[~]
# head dict.txt
featured
end
the
icon
title
Testing
Penetration
ttm
Training
Security
```

Elaboración de lista de palabras: Crunch

Crunch es un generador de listas de palabras donde puede especificar un juego de caracteres estándar o un juego de caracteres que usted especifique. crunch puede generar todas las combinaciones y permutaciones posibles. Aquí, usamos crunch para crear una lista de palabras con un mínimo de 2 y un máximo de 3 caracteres y escribimos el resultado dentro de una lista de palabras con el nombre dict.txt.

Más información: [Guía completa sobre la herramienta Crunch](#)

```
(root@kali)-[~]
# crunch 2 3 -o dict.txt
Crunch will now generate the following amount of data: 72332 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 18252

crunch: 100% completed generating output

(root@kali)-[~]
# head dict.txt
aa
ab
ac
ad
ae
af
ag
ah
ai
aj
```

Elaboración de listas de palabras: Cupp

Una contraseña débil puede ser muy corta o utilizar solo caracteres alfanuméricos, lo que simplifica el descifrado. Una contraseña débil también puede ser adivinada fácilmente por alguien que perfila al usuario, como un cumpleaños, apodo, dirección, nombre de una mascota o pariente, o una palabra común como Dios, amor, dinero o contraseña. Aquí es donde entra en uso Cupp, ya que puede usarse en situaciones como pruebas de penetración legales o investigaciones de delitos forenses. Aquí, estamos creando una lista de palabras específica para una persona llamada Raj. Ingresamos los detalles y, al enviarlos, tenemos una lista de palabras que se genera especialmente para este usuario.

GitHub: [CUPP – Perfilador de contraseñas de usuarios comunes](#)

Más información: [Guía completa sobre Cupp: una herramienta para generar listas de palabras](#)

```
(root@kali)-[~]
# cupp -i

cupp.py!

# Common
# User
# Passwords
# Profiler

[ Muris Kurgas | j0rgan@remote-exploit.org ]
[ Mebus | https://github.com/Mebus/ ]

[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked ;)

> First Name: raj
> Surname: chandel
> Nickname:
> Birthdate (DDMMYYYY):

> Partners) name:
> Partners) nickname:
> Partners) birthdate (DDMMYYYY):

> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):

> Pet's name:
> Company name:

> Do you want to add some key words about the victim? Y/[N]:
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]:

[+] Now making a dictionary ...
[+] Sorting list and removing duplicates ...
[+] Saving dictionary to raj.txt, counting 150 words.
[+] Now load your pistolero with raj.txt and shoot! Good luck!

(root@kali)-[~]
# cat raj.txt
Chandel
Chandel2008
Chandel2009
Chandel2010
Chandel2011
Chandel2012
```


Elaboración de listas de palabras: Pydictor

Pydictor es una de esas herramientas que tanto los principiantes como los profesionales pueden apreciar. Es una herramienta de creación de diccionarios que es excelente tener en su arsenal cuando se trata de pruebas de seguridad de contraseñas. La herramienta ofrece una gran cantidad de funciones que se pueden utilizar para crear el diccionario perfecto para prácticamente cualquier tipo de situación de prueba. Aquí, definimos la base y la longitud como 5 y luego creamos una lista de palabras. La lista de palabras contiene números de hasta 5 dígitos.

GitHub: [pydictor](#)

Más información: [Guía completa sobre Pydictor: una herramienta para generar listas de palabras](#)

```
(root@kali)~[/pydictor]
# python pydictor.py --len 5 5 -base d -o dict.txt

[+] A total of :100000 lines
[+] Store in :/root/pydictor/results/dict.txt
[+] Cost :0.3021 seconds

(root@kali)~[/pydictor]
# head /root/pydictor/results/dict.txt
00000
00001
00002
00003
00004
00005
00006
00007
00008
00009
```

Elaboración de listas de palabras: Bopscrk

Bopscrk (Before Outset PaSsword CRackIng) es una herramienta para generar listas de palabras inteligentes y potentes para ataques dirigidos. Es parte de Black Arch Linux desde que tenemos memoria. Introduce información personal relacionada con el objetivo y combina cada palabra y la transforma en posibles contraseñas. También contiene un módulo de pase de letras que le permite buscar letras relacionadas con el artista favorito del objetivo y luego incluirlas en las listas de palabras.

GitHub: [Bopscrk](#)

Fields can be left empty. You can use accentuation in your words. If you enable case transforms, won't matter the lower/uppercases in your input. In "others" field (interactive mode), you can write several words comma-separated (e.g.: 2C,Flipper).

For advanced usage and documentation:
<https://github.com/r3nt0n/bopscrk>

[illegible]

```
[?] Passwords min length [4] >>> 4
[?] Password's max length [12] >>> 8
[?] First name >>> raj
[?] Surname >>> chandel
[?] Last name >>>
[?] Birth date (DD/MM/YYYY) >>>
[?] Some other relevant words (comma-separated) >>> ignite
[?] Do yo want to make leet transforms? [y/n] >>> n
[?] Do yo want to make case transforms? [y/n] >>> y
[?] How much words do you want to combine at most [2] >>>
[?] Artist names to search song lyrics (comma-separated) >>>
[?] Exclude words from other wordlists? >>>
[?] Output file [tmp.txt] >>> dict.txt
```

```
[+] Appending words provided (base wordlist length: 3)...
```

```
[+] Time elapsed: 0:00:00
[+] Output file: dict.txt
[+] Words generated: 531
```

Aquí, podemos ver que la lista de palabras que se creó a partir de los detalles que proporcionamos es ordenada y elaborada, con una alta probabilidad de ser la contraseña real del usuario de Raj.

```
(root@kali)-[~/bopscrk]
# cat dict.txt
chandel
ignite
raj.
.raj
raj_
_raj
raj-
-raj
raj$
$raj
raj%
%raj
raj&
&raj
raj#
#raj
raj@
@raj
raj123
123raj
rajxXx
xXxraj
raj !!
!! raj
chandel.
.chandel
chandel_
_chandel
chandel-
```

Elaboración de listas de palabras: BEWGor

Para empezar, comencemos con la pronunciación. Se pronuncia como moco. Sé que no es fácil entenderlo. BEWGor está diseñado para ayudar a garantizar la seguridad de las contraseñas. Es un script de Python que solicita al usuario datos biográficos sobre una persona, denominada Sujeto. Estos datos luego se utilizan para crear contraseñas probables para ese Asunto. BEWGor se basa en gran medida en Cupp, pero son diferentes en algunos aspectos, ya que presenta información detallada mucho mayor sobre el tema principal, incluye soporte para una cantidad arbitraria de miembros de la familia y mascotas. Los usuarios pueden usar permutaciones para generar posibles contraseñas. Además, BEWGor puede generar una gran cantidad de contraseñas, crear variaciones superior/inferior/inversa de los valores ingresados, guardar los valores ingresados sin procesar en un archivo de términos antes de que se generen las variaciones, establecer límites superior e inferior en la longitud de la línea de salida y verificar que un cumpleaños ingresado es válida. Los cumpleaños no deben ser el futuro, un falso día bisiestro, el 32 de junio, etc.

 GitHub: [BEWGor - Generador de listas de palabras en forma de diana](#)



You will be asked a series of questions about your Subject.
Your answers will be used to generate a wordlist.
The lists are generated using all permutations of inputted numbers and words.

If you are unable or uninterested in providing input for a specific prompt...

ANY PROMPT CAN BE LEFT BLANK BY PRESSING ENTER

If you do not know how to answer a prompt, more research may be needed.
Use --- <http://www.osintframework.com/> --- to find many useful tools.

PAY ATTENTION

Many prompts include specific details about input - read carefully.
Failing to do so will result in a poor quality wordlist!

Let's begin!

Press enter to continue ...

Section A: Main Subject Information

> Enter The Main Subject's Full Name, separated by spaces - or as much as you have >:Raj

> Enter Raj's Maiden Name - if applicable >:

For nicknames, think about common name shortenings,
such as 'Michael' into 'Mike'
Also enter usernames and online handles.

> Enter one of Raj's Nicknames or usernames, or simply press enter to move on >:

Be Aware BEWGor uses DDMM formatting!
Winter Solstice falls on 21/12, 22/12, or 23/12 in this format.

> Enter Raj's Birthday (without year, DDMM) >:

> Enter Raj's Birth year (YYYY) >:2020

Would you like to include Raj's Chinese Zodiac Sign? (Y/N) >:

---Raj's Gender Identity, in all relevant forms---

MALE and FEMALE are the most commonly identified genders.
BEWGor can include (English) synonyms for these two choices if
they are entered. You may wish to include synonyms in the
Subject's native language. For example, a Spanish speaker might
include "chico" in their password. A German speaker may include "Frau"
Transgendered or transsexual people may identify as male,

Después de trabajar un rato, vemos que tenemos una lista de palabras refinada para el usuario Raj. Ahora se puede utilizar para aplicar fuerza bruta a las credenciales de Raj.


```
(root@kali)-[~]
# cat 1.txt
00001
00002
00003
00004
00005

(root@kali)-[~]
# cat 2.txt
00010
00020
00030
00040
00050
```

Al ejecutar DyMerge, proporcionamos result.txt como la lista de palabras que se creará fusionando 1.txt y 2.txt. Se puede observar que result.txt tiene 10 entradas de ambas listas de palabras.

```
(root@kali)~/dymerge# ./dymerge.py ~/1.txt ~/2.txt -o result.txt
```

DyMerge 0.2 Nikolaos Kamarinakis (nikolaskama.me)

Made with <3 by k4m4

```
[+] Starting Dictionary Merge Task  
[+] Reading Dictionaries  
[+] Merging Dictionaries  
[+] Task Successfully Complete  
[+] Final Dictionary Saved As → result.txt  
Comp/tional Time Elapsed: 0.018297
```

```
(root@kali)~/dymerge# cat result.txt
```

```
00001  
00002  
00003  
00004  
00005  
00010  
00020  
00030  
00040  
00050
```

Elaboración de listas de palabras: mentalista

Es una herramienta GUI para crear listas de palabras personalizadas. Utiliza paradigmas humanos comunes para crear listas de palabras basadas en contraseñas. Puede crear la lista completa de palabras con contraseñas, pero también puede crear reglas compatibles con Hashcat y John the Ripper.

Se genera uniendo nodos que a su vez toman forma de cadena. El nodo inicial de la cadena se denomina nodo Palabras base. Luego, cada palabra base se pasa al siguiente nodo de la cadena como

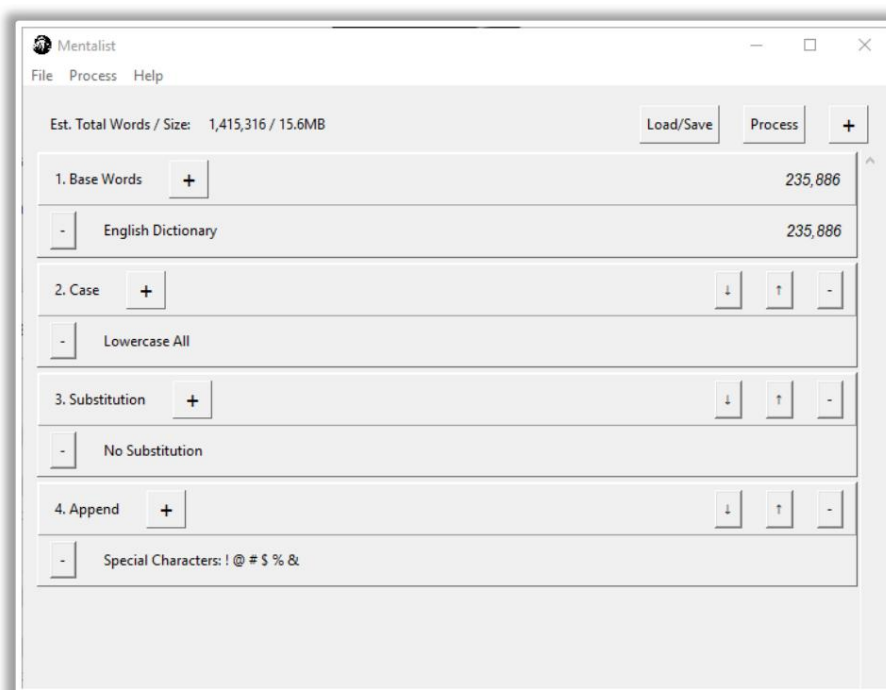
está procesado. Así es como se modifican las palabras en las listas de palabras. Después de trabajar en la cadena, finalmente escribe el resultado de la cadena en el archivo especificado o lo convierte en reglas según la solicitud del usuario.

Reglas de Hashcat/John

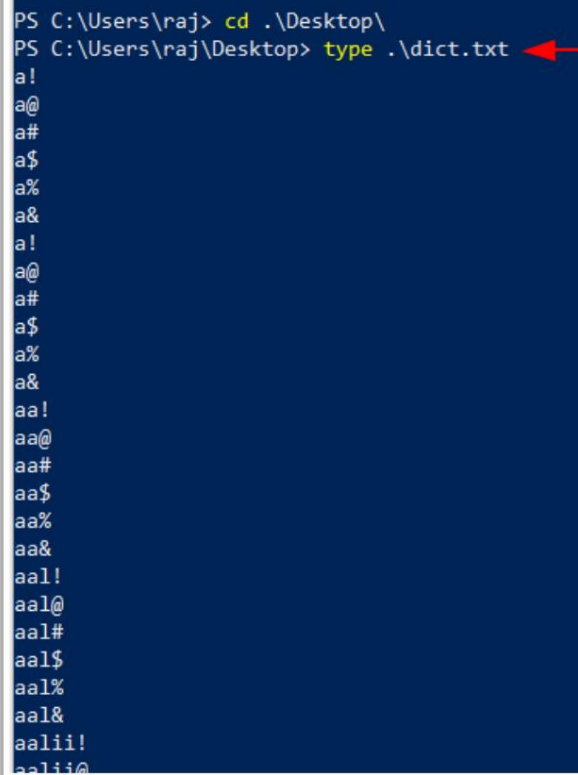
Para el craqueo fuera de línea, hay ocasiones en las que la lista completa de palabras es demasiado grande para generarla en su totalidad. En este caso, tiene sentido generar reglas para que Hashcat o John puedan generar mediante programación la lista de palabras completa. Descargue la versión de GitHub.

GitHub: [mentalista](#)

Estamos utilizando el sistema operativo Windows aquí para demostrar la capacidad de Mentalist. Hemos elegido el Diccionario de Inglés como palabras base. Calcula que se pueden manipular 235.886 posibles palabras clave en las contraseñas tomando como base diccionarios de inglés. Luego proporcionamos algunas opciones adicionales como Caso y si queremos sustituir entradas y Si queremos agregar un carácter especial después de cada entrada.



Después de ejecutarse por un tiempo, creó un archivo de texto con el nombre dict.txt. Contiene todas las contraseñas que fueron posibles de crear según nuestros requisitos.



```
PS C:\Users\raj> cd .\Desktop\  
PS C:\Users\raj\Desktop> type .\dict.txt  
a!  
a@  
a#  
a$  
a%  
a&  
a!  
a@  
a#  
a$  
a%  
a&  
aa!  
aa@  
aa#  
aa$  
aa%  
aa&  
aal!  
aal@  
aal#  
aal$  
aal%  
aal&  
aalii!  
aalii@
```

Conclusión

El punto que intentamos transmitir a través de este artículo es que la lista de palabras es uno de los activos más importantes que puede tener un evaluador de penetración. Hay múltiples recursos para obtener una lista de palabras y múltiples herramientas para crear su propia lista de palabras. Queríamos que este artículo le sirviera de guía siempre que intente aprender o utilizar una lista de palabras o cualquiera de las herramientas para elaborar una lista de palabras.

Referencias

- <https://www.hackingarticles.in/wordlists-for-pentester/>

ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

