#PassRipperHydra

PassRipperHydra is an advanced AI-powered password cracking tool designed for red team operations. It leverages machine learning models like PassGAN for password generation, integrates with tools like John the Ripper and Hydra for offline and online attacks, and includes advanced captcha-solving capabilities using a pretrained CNN model. The tool features a cyberpunk-themed Streamlit GUI for ease of use.
Features

AI-Powered Password Generation: Uses PassGAN to generate realistic passwords.
Offline Hash Cracking: Integrates with John the Ripper for cracking hashes with GPU acceleration.
Online Brute-Force Attacks: Supports multiple protocols (SSH, HTTP-Form, FTP, SMB, RDP, Telnet, MySQL, LDAP, PostgreSQL, SNMP, SIP) using Hydra.
Advanced Captcha Solving: Solves image and audio captchas using a pretrained CNN model and third-party APIs (2Captcha, Anti-Captcha).
Proxy Rotation: Enhances anonymity during online attacks using BrightData and ProxyRack.
Export Options: Exports results in JSON, CSV, and PDF formats.
Visualizations: Displays attack success rates and password complexity distributions using Plotly.
Cyberpunk UI: A visually stunning Streamlit interface with light/dark theme toggle.

#Prerequisites

Python 3.7: Required for compatibility with tensorflow==1.13.1.
Hashcat: For offline hash cracking (optional but recommended for GPU acceleration).
API Keys:
2Captcha or Anti-Captcha for captcha solving.
BrightData or ProxyRack for proxy rotation.

#Hardware:
GPU (optional, for faster ML inference and hash cracking).
At least 8GB RAM and 10GB free disk space (for model training and storage).

Project Structure

```
PassRipperHydra/
├── assets/
│ └── logo.png # Logo for the Streamlit app
├── core/
│ ├── passgan.py # PassGAN model for password generation
│ ├── john_wrapper.py # Wrapper for John the Ripper
│ └── hydra_wrapper.py # Wrapper for Hydra brute-force attacks
├── utils/
│ ├── exporter.py # Export results to JSON/CSV/PDF
│ ├── helpers.py # Utility functions (captcha solving, proxy rotation)
│ └── logger.py # Logging setup
├── fine_tuned/
│ └── checkpoints/ # PassGAN checkpoint directory
```

```
├── models/
│   └── captcha_cnn.h5 # Pretrained CNN model for captcha solving
├── output/
│   ├── logs/ # Logs directory
│   └── results/ # Results directory
├── main.py # Streamlit app entry point
├── .env # Environment variables
├── requirements.txt # Python dependencies
└── README.md # Project documentation
```

Setup Instructions

Step 1: Install Python 3.7

PassRipperHydra requires Python 3.7 due to tensorflow==1.13.1 compatibility.

Windows:

Download Python 3.7 from python.org.
Install, ensuring "Add Python to PATH" is checked.
Verify: py -3.7 --version

macOS:

Install Homebrew: /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
Install Python 3.7: brew install python@3.7
Verify: python3.7 --version

Linux:

Update package list: sudo apt update
Install dependencies: sudo apt install -y build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev curl libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
Install pyenv: curl https://pyenv.run | bash
Add pyenv to shell: echo 'export PATH="$HOME/.pyenv/bin:$PATH"' >> ~/.bashrc && echo 'eval "$(pyenv init --path)"' >> ~/.bashrc && source ~/.bashrc
Install Python 3.7: pyenv install 3.7.0
Set global version: pyenv global 3.7.0
Verify: python --version

Step 2: Clone the Repository

git clone https://github.com/SunnyThakur25/PassRipperHydra.git
cd PassRipperHydra

Step 3: Set Up Virtual Environment

Windows:py -3.7 -m venv venv
venv\Scripts\activate

macOS/Linux:python3.7 -m venv venv
source venv/bin/activate

Step 4: Install Dependencies

pip install -r requirements.txt

Windows Note: If PyAudio fails to install:pip install pipwin
pipwin install pyaudio

Linux Note: Install system dependencies for PyAudio:sudo apt install -y portaudio19-dev

TensorFlow Note: If tensorflow==1.13.1 fails, download a compatible wheel from Christoph Gohlke's site and install manually:pip install path\to\tensorflow-1.13.1-cp37-cp37m-.whl

step 5: Configure Environment Variables

Copy the example .env file:cp .env.example .env

Edit .env with your API keys and paths:2CAPTCHA_API_KEY=your_2captcha_api_key
ANTI_CAPTCHA_API_KEY=your_anti_captcha_api_key
BRIGHTDATA_API_KEY=your_brightdata_api_key
PROXYRACK_API_KEY=your_proxyrack_api_key
CAPTCHA_MODEL_PATH=models/captcha_cnn.h5
LOG_DIR=output/logs
OUTPUT_DIR=output/results
STREAMLIT_SERVER_PORT=8501
STREAMLIT_SERVER_HEADLESS=true

Obtain API keys from 2Captcha, Anti-Captcha, BrightData, and ProxyRack.
Ensure CAPTCHA_MODEL_PATH points to the pretrained captcha model (train it if missing—see below).

Step 6: Install Hashcat (Optional, for GPU Acceleration)

Windows:
Download Hashcat from hashcat.net.
Extract to C:\hashcat and add to PATH.

macOS:brew install hashcat

Linux:sudo apt install -y hashcat

Ensure GPU drivers (e.g., NVIDIA CUDA) are installed for acceleration.

Pretrained Models Setup

PassGAN Model
PassRipperHydra uses a pretrained PassGAN model for password generation. Follow these steps to set up the model:

Step 1: Clone the Original PassGAN Repository

git clone https://github.com/brannondorsey/PassGAN.git
cd PassGAN

Step 2: Adjust Paths in core/passgan.py

Update weights_path and checkpoint_path in main.py:passgan = PassGANModel(
weights_path="core/passgan_weights.pkl",
checkpoint_path="fine_tuned/checkpoints/checkpoint_200000.ckpt",
charset=st.session_state.passgan_config["charset"],
min_len=st.session_state.passgan_config["min_len"],
max_len=st.session_state.passgan_config["max_len"]
)

If using the cloned PassGAN weights:
Copy the pretrained weights/checkpoints from PassGAN/checkpoints/ to PassRipperHydra/fine_tuned/checkpoints/.
Update checkpoint_path to point to the copied checkpoint (e.g., PassRipperHydra/fine_tuned/checkpoints/checkpoint_200000.ckpt).
If passgan_weights.pkl doesn't exist, generate it by saving the model weights after loading the checkpoint:# In core/passgan.py, after loading checkpoint
with open(weights_path, 'wb') as f:
pickle.dump(self.model.get_weights(), f)

Step 3: (Optional) Pretrain PassGAN

If the pretrained weights are unavailable or you want to retrain:

Prepare a password dataset (e.g., RockYou dataset, available from various sources—ensure legal usage).
Follow the training instructions in the PassGAN repository:python train.py --dataset path/to/passwords.txt --output-dir fine_tuned/checkpoints

Copy the trained checkpoint to PassRipperHydra/fine_tuned/checkpoints/.
Update checkpoint_path in main.py to the new checkpoint.

CAPTCHA CNN Model

The CAPTCHA CNN model (models/captcha_cnn.h5) is used for solving image captchas.
Step 1: Prepare a CAPTCHA Dataset

Collect a dataset of CAPTCHA images (e.g., scrape from a target site or use a public dataset—ensure ethical usage).
Label the images with their corresponding text (e.g., captcha_001.png -> "abcd").

Organize the dataset:captcha_dataset/

```
├── images/
│ ├── captcha_001.png
│ ├── captcha_002.png
│ └── ...
└── labels.csv # Format: filename,text (e.g., captcha_001.png,abcd)
```

Step 2: Pretrain the CAPTCHA CNN Model

Use the following script to train the CNN model:

```
import tensorflow as tf
import numpy as np
import pandas as pd
from PIL import Image
import os
```

=== Constants ===

```
IMG_HEIGHT, IMG_WIDTH = 50, 200 # Adjust to your CAPTCHA dimensions
NUM_CLASSES = 36 # 0-9 + a-z
CHARSET = '0123456789abcdefghijklmnopqrstuvwxyz'
CHAR_TO_IDX = {c: i for i, c in enumerate(CHARSET)}
```

=== Load Dataset ===

```
def load_dataset(data_dir: str, labels_file: str):
labels_df = pd.read_csv(labels_file)
images = []
labels = []
```

```
    for _, row in labels_df.iterrows():
        img_path = os.path.join(data_dir, 'images', row['filename'])
        img = Image.open(img_path).convert('L')  # Grayscale
        img = img.resize((IMG_WIDTH, IMG_HEIGHT))
        img_array = np.array(img) / 255.0  # Normalize
        images.append(img_array)

        # Encode label to fixed length (4 chars max)
        label = row['text'].lower()
        label_encoded = [CHAR_TO_IDX[c] for c in label]
        label_encoded += [0] * (4 - len(label_encoded))  # Padding
        labels.append(label_encoded)

    return np.array(images), np.array(labels)
```

=== Build CNN Model ===

def build_model():
model = tf.keras.Sequential([
tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)),
tf.keras.layers.MaxPooling2D((2, 2)),
tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
tf.keras.layers.MaxPooling2D((2, 2)),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(4 * NUM_CLASSES, activation='softmax') # For 4 char positions
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model

=== Training Pipeline ===

def train(data_dir: str, labels_file: str, save_path: str):
print("[*] Loading dataset...")
images, labels = load_dataset(data_dir, labels_file)
images = images.reshape(-1, IMG_HEIGHT, IMG_WIDTH, 1)
labels = labels.reshape(-1, 4, 1) # 4 labels per image

```
    print("[*] Building model...")
    model = build_model()

    print("[*] Training model...")
    model.fit(images, labels, epochs=10, batch_size=32, validation_split=0.2)
```

```
print("[*] Saving model...")
model.save(save_path)
print(f"[+] Model saved to {save_path}")
```

=== Entry Point ===

if **name** == "**main**":
data_dir = "captcha_dataset"
labels_file = os.path.join(data_dir, "labels.csv")
save_path = "models/captcha_cnn.h5"
train(data_dir, labels_file, save_path)
Run the training script:
python train_captcha_cnn.py

Ensure captcha_dataset/ exists with the structure described above.
The trained model will be saved to models/captcha_cnn.h5
Step 3: Update .env

Ensure CAPTCHA_MODEL_PATH in .env points to the trained model:

CAPTCHA_MODEL_PATH=models/captcha_cnn.h5

Running the Tool

```
Activate the virtual environment:
    Windows: venv\Scripts\activate
    macOS/Linux: source venv/bin/activate
Start the Streamlit app:


streamlit run main.py
Access the app at http://localhost:8501.
Use the GUI to:
    Upload hash files for offline cracking.
    Perform online brute-force attacks with proxy rotation and captcha solving.
    Generate passwords and export results.
```

Troubleshooting

```
TensorFlow Installation Fails:
    Ensure Python 3.7 is used.
    Try a pre-built wheel from Christoph Gohlke’s site.
Hashcat Not Found:
    Install Hashcat and add it to your PATH.
```

```
Model Loading Fails:
    Verify model files exist at the specified paths.
    Retrain models if necessary (see above).
Logs:
    Check output/logs/passripperhydra.log for detailed error messages.
```

Future Improvements

```
Upgrade to TensorFlow 2.x for better performance and compatibility.
Support longer passwords (up to 32 characters) with PassGAN.
Integrate transformer-based models (e.g., GPT-2) for password generation.
Add multi-target batch processing with parallel execution.
Deploy on cloud platforms (AWS, GCP) with distributed computing.
Export results to SIEM systems (Splunk, ELK).
```

License

This project is for educational and authorized red team use only. Ensure compliance with all applicable laws and regulations.
Contributors

Developed by sunny thakur | June 2025
Key Sections Explained

```
Project Overview:
    Describes the tool, its features, and its purpose for red team operations.
Prerequisites:
    Lists required software, API keys, and hardware.
Project Structure:
    Provides a clear directory layout for users to understand the codebase.
Setup Instructions:
    Detailed steps for Windows, macOS, and Linux, including Python 3.7
installation, virtual environment setup, dependency installation, and
environment configuration.
Pretrained Models Setup:
    Instructions for setting up PassGAN (cloning, adjusting paths, optional
retraining) and training the CAPTCHA CNN model with a provided script.
Running the Tool:
    Steps to launch the Streamlit app and use its features.
Troubleshooting:
    Common issues and solutions (e.g., TensorFlow installation, missing
models).
Future Improvements:
    Lists planned enhancements (aligned with previous conversations).
License and Contributors:
    Includes a legal disclaimer and credits.
```