```
import requests
import re
import csv
import time
from datetime import datetime

# ================================================================================
# PROJECT 4: THREAT INTELLIGENCE AGGREGATOR
# AUTHOR: Raj Bharti
# PURPOSE: Fetch and normalize IOCs (Indicators of Compromise) from public feeds.
# ================================================================================

# --- CONFIGURATION ---
# We will use URLHaus (Abuse.ch) - A safe, public source for malware URLs
SOURCE_URL = "https://urlhaus.abuse.ch/downloads/csv_recent/"
OUTPUT_FILE = "ioc_blocklist.csv"

def fetch_threat_data():
    print("==================================================")
    print("   THREAT INTELLIGENCE AGGREGATOR - STARTED       ")
    print("==================================================")
    print(f"[*] Connecting to Threat Feed: {SOURCE_URL}...")

    try:
        response = requests.get(SOURCE_URL)
        if response.status_code == 200:
            print("[+] Connection Successful! Downloaded data.")
            return response.text
        else:
            print(f"[-] Error: Failed to fetch data. Status Code:
{response.status_code}")
            return None
    except Exception as e:
        print(f"[-] Network Error: {e}")
        return None

def process_data(raw_data):
    print("[*] Processing and Normalizing Data...")

    iocs = []
    lines = raw_data.splitlines()

    # URLHaus CSV format is usually:
id,dateadded,url,url_status,threat,tags,urlhaus_link,reporter
    # We will skip comments (#) and extract the URL and Threat Type

    count = 0
    for line in lines:
        # Skip comments
        if line.startswith("#") or not line.strip():
            continue

        parts = line.split(',')
        if len(parts) > 3:
            # Clean up the data (remove quotes)
            malicious_url = parts[2].replace('"', '')
            threat_type = parts[4].replace('"', '')

            # Add to our list
            iocs.append([malicious_url, threat_type, "High", 
datetime.now().strftime("%Y-%m-%d")])
            count += 1

            # Limit for demonstration (keep file size small)
            if count >= 50:
                break

    print(f"[+] Successfully extracted {len(iocs)} IOCs.")
    return iocs

def save_to_csv(iocs):
    try:
        with open(OUTPUT_FILE, mode='w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            # Write Header
            writer.writerow(["Indicator (IOC)", "Threat Type", "Severity", "Date 
Detected"])
            # Write Data
            writer.writerows(iocs)

        print(f"[+] IOC Correlation Output saved to: {OUTPUT_FILE}")
        print("[*] TASK COMPLETE. Blocklist ready for firewall integration.")

    except Exception as e:
        print(f"[-] File Error: {e}")

if __name__ == "__main__":
    raw_text = fetch_threat_data()
    if raw_text:
        parsed_iocs = process_data(raw_text)
        save_to_csv(parsed_iocs)
```