Bug Bounty Tool

A simple Python-based **Bug Bounty Tool** for automated reconnaissance and basic vulnerability checks (SQLi, XSS, security headers, port scanning). This repository is intended as a starting point for a larger bug-bounty automation project.

Repository Structure

```
bug-bounty-tool/
├─ README.md                # Project overview (this file)
├─ LICENSE                  # MIT License
├─ .gitignore               # Common ignores for Python
├─ requirements.txt         # Python dependencies
├─ src/
│  └─ main.py               # Main scanner script
└─ examples/
   └─ sample-output.txt     # Example scan output
```

Features

- Basic reachability check
- Simple SQL injection detection with common payloads
- Reflected XSS test using a basic payload
- HTTP security header checks (CSP, X-Frame-Options, HSTS)
- Lightweight port scanner for common ports

Installation

1. Clone the repository:

```
git clone https://github.com/yourusername/bug-bounty-tool.git
cd bug-bounty-tool
```

1. Create a virtual environment and install requirements:

```
python3 -m venv venv
source venv/bin/activate    # Linux / macOS
```

```
venv\Scripts\activate     # Windows
pip install -r requirements.txt
```

Usage

Run the main script and provide a target URL when prompted:

```
python src/main.py
```

Example target input: `https://example.com`

Important: Only scan targets for which you have explicit permission. Unauthorized scanning may be illegal.

Files (to create)

`src/main.py`

```python
import requests
from urllib.parse import urljoin
import socket

# --- Function 1: Check if website is reachable ---
def check_website(url):
    try:
        response = requests.get(url, timeout=5)
        print(f"[+] Website reachable: {url} (Status Code:
{response.status_code})")
    except requests.exceptions.RequestException:
        print(f"[-] Could not reach {url}")

# --- Function 2: Check for SQL Injection ---
def check_sql_injection(url):
    payloads = ["'", "' OR '1'='1", '" OR "1"="1']
    vulnerable = False

    for payload in payloads:
        test_url = url + payload
        try:
            response = requests.get(test_url, timeout=5)
            if any(error in response.text.lower() for error in ["sql", "syntax",
"mysql", "ora-", "error"]):
```

```
                print(f"[!] Possible SQL Injection vulnerability at:
{test_url}")
                vulnerable = True
                break
        except requests.exceptions.RequestException:
            continue
    if not vulnerable:
        print("[+] No SQL Injection detected.")

# --- Function 3: Check for XSS (Cross Site Scripting) ---
def check_xss(url):
    payload = "<script>alert('XSS')</script>"
    try:
        response = requests.get(url, params={"q": payload}, timeout=5)
        if payload in response.text:
            print(f"[!] Possible XSS vulnerability found at {url}")
        else:
            print("[+] No XSS vulnerability detected.")
    except requests.exceptions.RequestException:
        print("[-] XSS test request failed")

# --- Function 4: Check HTTP Security Headers ---
def check_security_headers(url):
    try:
        response = requests.get(url, timeout=5)
        headers = ["Content-Security-Policy", "X-Frame-Options", "Strict-
Transport-Security"]
        print("\n[+] Checking Security Headers:")
        for h in headers:
            if h in response.headers:
                print(f"   [+] {h}: Found")
            else:
                print(f"   [-] {h}: Missing")
    except Exception as e:
        print("Error checking headers:", e)

# --- Function 5: Simple Port Scanner ---
def port_scan(host):
    print(f"\n[+] Scanning ports for {host}")
    open_ports = []
    common_ports = [21, 22, 23, 25, 80, 443, 3306]
    for port in common_ports:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        socket.setdefaulttimeout(1)
        try:
            result = sock.connect_ex((host, port))
            if result == 0:
                open_ports.append(port)
        except Exception:
            pass
        finally:
            sock.close()
    if open_ports:
        print(f"[!] Open ports found: {open_ports}")
    else:
        print("[+] No common open ports found.")

# --- Main Function ---
if __name__ == "__main__":
    print("=== Simple Bug Bounty Tool ===")
    target = input("Enter target URL (e.g., https://example.com): ").strip()
    host = target.replace("https://", "").replace("http://", "").split("/")[0]

    check_website(target)
    check_sql_injection(target)
    check_xss(target)
    check_security_headers(target)
    port_scan(host)

    print("\nScan Complete ✅")
```

`.gitignore`

```
__pycache__/
*.pyc
venv/
.env
.DS_Store
```

`requirements.txt`

```
requests>=2.28.0
```

Contributing

- Open issues for feature requests or bugs.
- Create pull requests against the `main` branch.
- Follow a clear commit style and include tests for new functionality.

License

Notes

LEGAL: Only use this tool against systems you own or have explicit permission to test. Unauthorized scanning can be illegal and unethical.