Python 기반의 Web 스캐너 개발

Made by 박수현

Email : qkrtngus211@naver.com

A Table of Contents.

Part 1, 개요

웹 서비스의 노출되는 취약점을 진단하기 위한 프로그램입니다.

작성 언어 : Python 3.8
개발 환경 : Windows 10

필요 패키지 : BeautifulSoup, scapy, python-nmap

개발 목적 : 운영중인 웹 서비스의 취약점을 제거하고 서비스의 안정성 및 신뢰성 향상을 목표로 함

상세 내용: Response 헤더의 서버 정보 노출 여부 확인. 사용중인 SSL/TLS 암호화 프로토콜의 강도 검사. SYN 스캐닝을 활용한 Well-known 포트 오픈 여부 확인. 웹 서버 및 WAS 디폴트 페이지, 테스트, 백업, DB, 관리자 페이지, 디렉터리 인덱싱 등의 노출 여부 확인. 에러 페이지 내 서버 정보 노출 여부 확인. 내부망 서비스일 경우 외부로 노출되는 페이지가 존재하는지 구글링 검색 결과 확인.

세부적인 코드 내용과 파일은 아래 Github URL을 통해 확인하실 수 있습니다.

Github URL : https://github.com/zjsl3784/webscan_v1

Part 2, # 검사 항목

Response 값의 헤더 검사

서버에서 반환하는 Response 값의 헤더에 서버 정보가 노출되는지 여부를 검사합니다.

```
######### Response 헤더 값 검사 ##############################
def get_server_info(base_url):
    try:
        response = requests.options(base_url)

        # 서버 헤더 가져오기
        server_header = response.headers.get('Server', 'N/A')
        powered_by_header = response.headers.get('X-Powered-By', 'N/A')

        return server_header, powered_by_header
    except Exception as e:
        print("Error:", e)
        return None, None

    if server_header and powered_by_header:
        print("Server:", server_header)
        print("X-Powered-By:", powered_by_header)
    else:
        print("서버 정보를 가져오는 데 문제가 있습니다.")

############################################################
```

```
검사할 도메인 주소 ====> http://[redacted]

Response 값의 헤더 검사
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16

SSL/TLS 암호화 강도 검사
https 통신을 사용하지 않음.

Well-known 포트들에 대한 SYN 스캔 시작! (시간이 소요될 수 있음)
Well-known 포트들에 대한 SYN 스캔 종료!

오픈 된 포트 목록:
80 port is Open!
22 port is Open!
21 port is Open!

웹 스캔 시작! (다소 시간이 소요될 수 있음)
웹 스캔 종료! END
```

Part 2, 검사 항목

사용중인 SSL/TLS 암호화 프로토콜 강도 검사

Nmap의 ssl-enum-chipers.nse 스크립트를 이용하여
사용중인 웹 서비스의 SSL/TLS 암호화 프로토콜을 검사합니다.

* 참고 : Nmap 설치 필요 https://nmap.org/

```
############   SSL/TLS 암호화 강도 검사   Nmap 설치 필요 ! ###############

def run_nmap_ssl_enum_ciphers(base_url):                #####
    host1 = re.search(r'(https?://)(.*?)(?:/|$)', base_url).group(2)
    try:
        # nmap 명령어 생성
        nmap_command = ["nmap", "--script", "ssl-enum-ciphers", host1]

        # nmap 실행 및 결과 얻기
        result = subprocess.run(nmap_command, capture_output=True, text=True)

        # 결과에서 ssl-enum-ciphers 부분 추출
        ssl_enum_ciphers_result = re.search(r'ssl-enum-ciphers:(.*)', result.stdout, re.DOTALL)

        # 결과 출력
        if ssl_enum_ciphers_result:
            #print(ssl_enum_ciphers_result.group(0))  # 전체 결과 출력
            print(ssl_enum_ciphers_result.group(1))  # 'ssl-enum-ciphers' 이후의 내용만 출력
        if not ssl_enum_ciphers_result:
            print("https 통신을 사용하지 않음.")

    except Exception as e:
        print("Error:", e)

#########################################################################
```

```
검사할 도메인 주소 ====> https://www.jt---[illegible]

Response 값의 헤더 검사
Server: Pepyaka
X-Powered-By: N/A

SSL/TLS 암호화 강도 검사

|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: client
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.3:
|     ciphers:
|       TLS_AKE_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_AKE_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_AKE_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|     cipher preference: client
|_  least strength: C

Nmap done: 1 IP address (1 host up) scanned in 6.42 seconds

Well-known 포트들에 대한 SYN 스캔 시작! (시간이 소요될 수 있음)
Well-known 포트들에 대한 SYN 스캔 종료!

오픈 된 포트 목록:
80 port is Open!
443 port is Open!

웹 스캔 시작! (다소 시간이 소요될 수 있음)
웹 스캔 종료! END
```

Part 2, # 검사 항목

SYN 스캐닝을 활용한 Well-known 포트 검사

SYN 스캐닝을 하여 잘 알려진 포트들에 대해 오픈 되어있는지 여부를 검사합니다.

```
################ Well-known 포트 SYN 스캐닝 #################

def syn_scan(base_url, ports):
    print("Well-known 포트들에 대한 SYN 스캔 시작! (시간이 소요될 수 있음)")
    portop = []
    portcl = []
    portfi = []
    host = re.search(r'(https?://)(.*?)(?:/|$)', base_url).group(2)
    try:
        for port in ports:
            # SYN 패킷 생성
            syn_packet = IP(dst=host) / TCP(dport=port, flags='S')

            # 패킷 전송 및 응답 대기
            response = sr1(syn_packet, timeout=3, verbose=False)

            # 응답 확인
            if response and response.haslayer(TCP):
                if response[TCP].flags == 18:  # SYN-ACK 응답 확인
                    portop.append(port)
                else:
                    portcl.append(port)
            else:
                portfi.append(port)


    except Exception as e:
        print(f"Error: {e}")
    print("Well-known 포트들에 대한 SYN 스캔 종료!")
    print(" ")
    print("오픈 된 포트 목록:")
    if not portop:
        print("오픈 된 포트가 없음.")
    for port in portop:
        print(f"{port} port is Open!")
    print(" ")

#########################################################################
```

```
# 검사할 포트
ports = [80, 443, 22, 21, 20, 8080, 8443, 23, 24, 25, 37, 49, 53,
         79, 88, 109, 110, 111, 113, 123, 139, 143, 161, 162, 445, 514, 873,
         3306, 1194, 1080, 3479, 3480, 5228, 5353, 6379, 1331, 1293, 3389  ]
```

```
검사할 도메인 주소 ====> http://[redacted]

Response 값의 헤더 검사
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16

SSL/TLS 암호화 강도 검사
https 통신을 사용하지 않음.

Well-known 포트들에 대한 SYN 스캔 시작! (시간이 소요될 수 있음)
Well-known 포트들에 대한 SYN 스캔 종료!

오픈 된 포트 목록:
80 port is Open!
22 port is Open!
21 port is Open!

웹 스캔 시작! (다소 시간이 소요될 수 있음)
웹 스캔 종료! END
```

Part 2, # 검사 항목

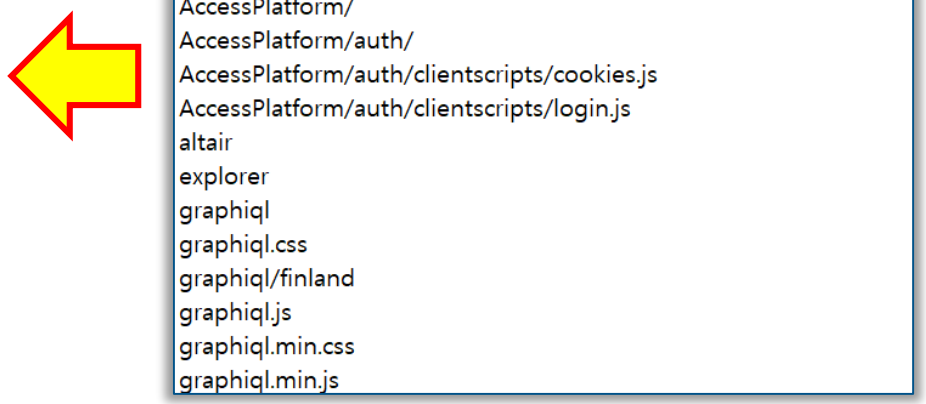
웹 스캔

WordList를 대입하여 웹 서버 및 WAS 디폴트 페이지, 테스트, 백업, DB, 관리자 페이지, 디렉터리 인덱싱 등의 노출 여부 확인합니다.

```
######### Word List 기반 잘 알려진 관리자페이지 및 서버 디폴트 페이지 등 스캔 ##################
def check_url_existence(base_url, additional_paths):
    """
    주어진 URL이 존재하는지 확인하는 함수
    """
    requests_count = 10     #타이머
    requests_per_second = 5     #타이머
    interval = 1 / requests_per_second  #타이머

    urls_200 = []  # 200 응답 리스트
    urls_302 = []  # 302 응답 리스트
    urls_403 = []  # 403 응답 리스트
    urls_404 = []  # 404 응답 리스트
    urls_ano = []  # 기타 응답 리스트
    urls_chk = []  # 체크가 필요한 리스트
    serverinfo = [] # 서버정보 체크용

    print("웹 스캔 시작! (다소 시간이 소요될 수 있음)")
    try:
        for path in additional_paths:
            url = base_url + path

            headers = {
    'Cache-Control': 'max-age=0',
    'Sec-Ch-Ua': '"Chromium";v="123", "Not:A-Brand";v="8"',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
    'Sec-Ch-Ua-Mobile': '?0',
    'Upgrade-Insecure-Requests': '1',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
    'Cookie': 'a'
}
            # time.sleep(interval) #타이머
            prepared_request = request.prepare()

            response = requests.get(url, headers=headers)

            if response.status_code == 200:
                page_content = response.text
                if "404" in page_content or "페이지가 존재" in page_content or "에러" in page_content or
                    if "not" in page_content and "found" in page_content:
                        urls_404.append(url)
                else:
                    urls_200.append(url)
                    urls_chk.append(url)

            elif response.status_code == 403:
                page_content = response.text
                if "404" in page_content or "페이지가 존재" in page_content or "에러" in page_content or
                    urls_404.append(url)
```

웹 사이트에 로그인이 필요한 경우 headers 값에 로그인 후 세션 값이 포함된 쿠키 값을 넣음

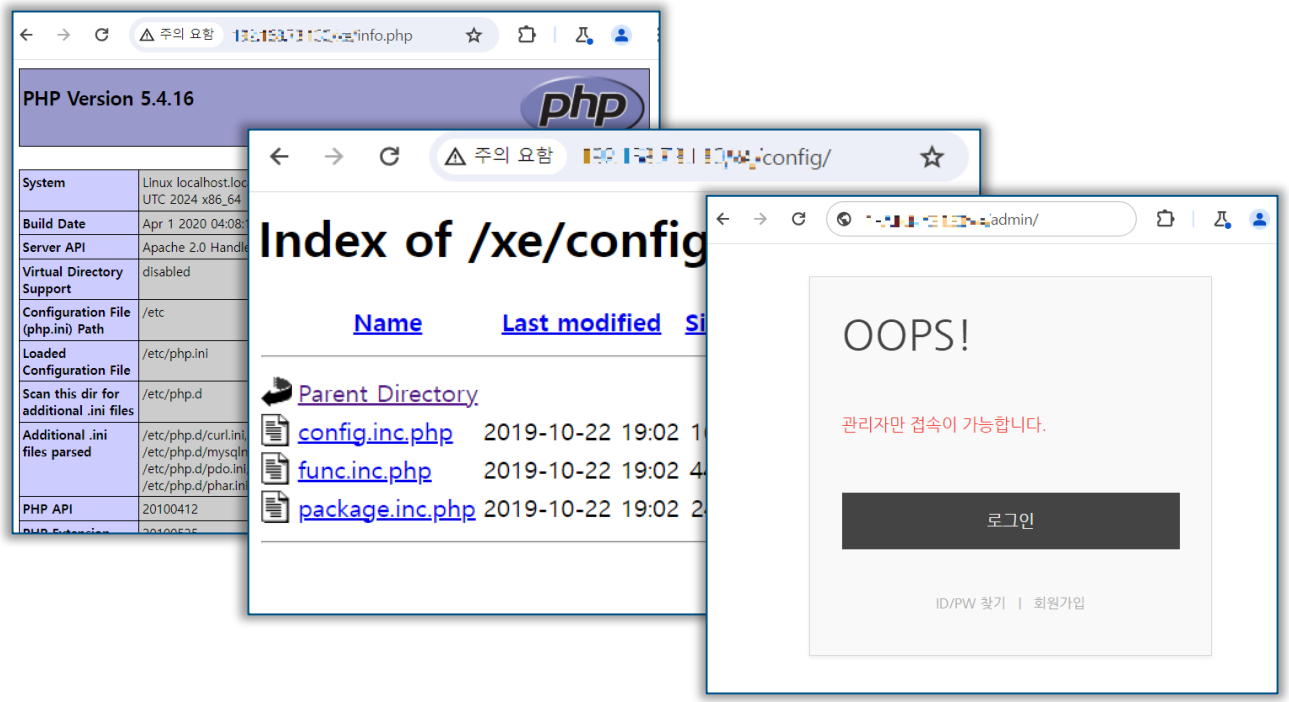
Apache, Tomcat, Nginx, IIS, WebLogic, Jboss, Wordpress, mysql, mssql, graphql, PHP, Jsp 등을 포함한 약 1천개의 주요 설정 파일 WordList.



Part 2, # 검사 항목

웹 스캔

WordList를 대입하여 서버의 응답 값을 분류 합니다.
200 또는 302 응답을 받는 경우 추가로 디렉터리 인덱싱 존재여부를 확인합니다.
403, 404 및 기타 응답을 받는 경우, 해당 에러 페이지 내 서버 정보 존재여부를 확인합니다.




```
웹 스캔 시작! (다소 시간이 소요될 수 있음)
웹 스캔 종료! END

200 응답을 받은 URL:
http://[redacted]/COPYRIGHT
http://[redacted]/LICENSE
http://[redacted]/README.md
http://[redacted]/admin
http://[redacted]/config
http://[redacted]/classes

302 응답을 받은 URL:
http://[redacted]/test.php

403 응답을 받은 URL:
http://[redacted]/sys

기타 응답을 받은 URL:
http://[redacted]/info.php

디렉터리 인덱싱 의심항목 :
http://[redacted]/config 페이지에 'index of' 문자열이 포함됨.
http://[redacted]/classes 페이지에 'index of' 문자열이 포함됨.

서버정보 노출 의심항목 :
http://[redacted]/info.php 페이지에 서버정보로 의심되는 문자열이 포함됨.
```

Part 2, 검사 항목

구글링

내부망에서 운영중인 서비스인 경우,

구글 검색을 통해 외부로 노출되는 페이지가 있는지 검사합니다.

```
############ 구글링 ##############################################
    googlestr1 = 'https://www.google.com/search?start=1&q=site%3A'
    googlechk1 = googlestr1 + base_url
    response = requests.get(googlechk1, headers=headers)

    if response.status_code == 200:
        print('구글링 요청이 성공했습니다.')
        print(" ")
        print('구글링 응답 내용:')

        # BeautifulSoup을 사용하여 HTML 파싱
        soup = BeautifulSoup(response.text, 'html.parser')

        # h3 태그에 해당하는 요소 추출
        h3_tags = soup.find_all('h3', class_='LC20lb')

        if not h3_tags:
            print("검색 결과가 없습니다.")
            print(" ")

        # 각 h3 태그의 텍스트 출력
        for h3_tag in h3_tags:
            print(h3_tag.text)
            print(" ")
    else:
        print('구글링 요청이 실패했습니다. 상태 코드:', response.status_code)

    googlestr2 = 'https://www.google.com/search?start=8&q=site%3A'
    googlechk2 = googlestr2 + base_url
    response = requests.get(googlechk2, headers=headers)

    if response.status_code == 200:
        # BeautifulSoup을 사용하여 HTML 파싱
        soup = BeautifulSoup(response.text, 'html.parser')

        # h3 태그에 해당하는 요소 추출
        h3_tags = soup.find_all('h3', class_='LC20lb')
```

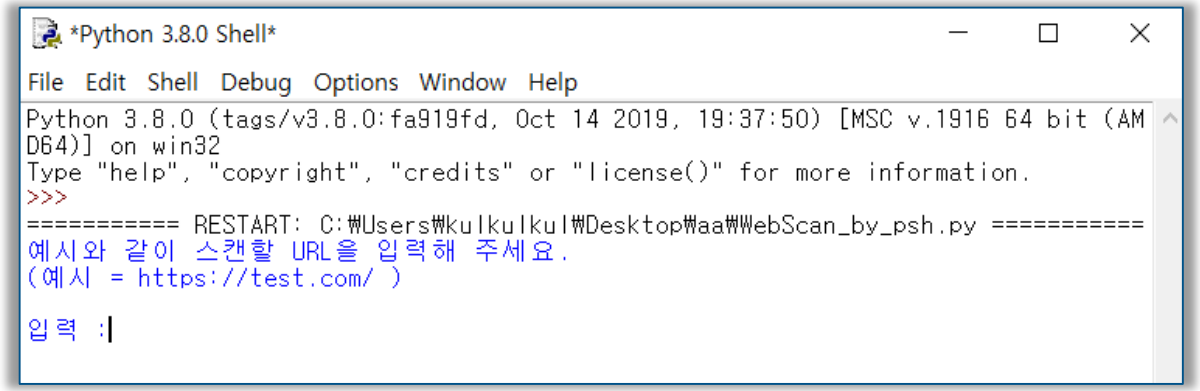
```
기타 응답을 받은 URL:
http:// [redacted] /info.php

디렉터리 인덱싱 의심항목 :
http:// [redacted] /config 페이지에 'index of' 문자열이 포함됨.
http:// [redacted] /classes 페이지에 'index of' 문자열이 포함됨.

서버정보 노출 의심항목 :
http:// [redacted] /info.php 페이지에 서버정보로 의심되는 문자열이 포함됨.


구글링 요청이 성공했습니다.

구글링 응답 내용:
검색 결과가 없습니다.

종료 END
```

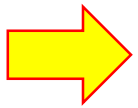
Part 3, 사용 예시

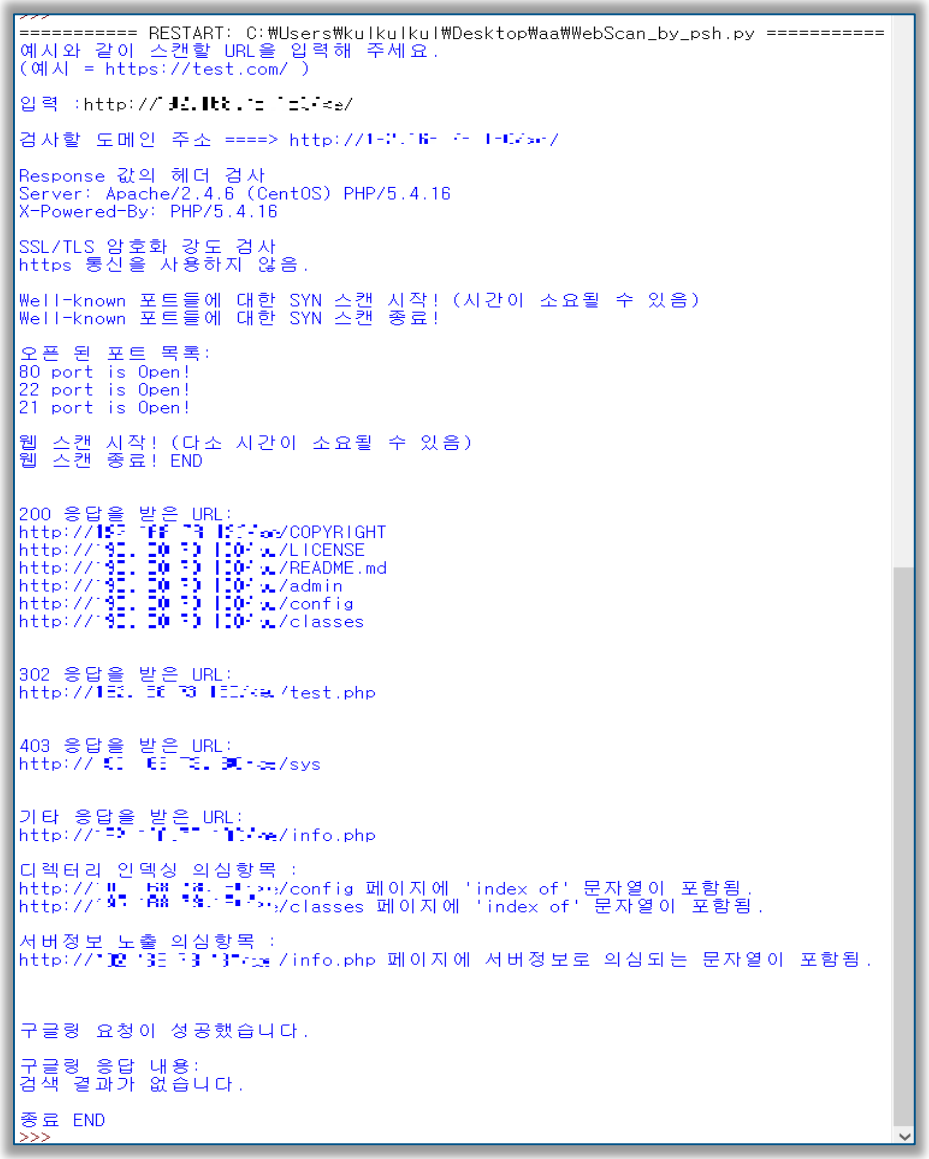
사용법

스캔할 URL을 입력합니다. 5분~ 10분 정도 시간이 소요될 수 있습니다.
SSL/TLS 암호화 프로토콜 점검 시 별도의 Nmap 프로그램 설치가 필요합니다.
* Github 참조

입력 예시 :
https://test.com/ o
https://test.com x
192.168.0.25 x
http://192.168.0.25/ o







참고 자료

2021 주요정보통신기반시설 기술적 취약점 분석/평가 방법 상세 가이드

웹(Web)

DI (상)	8. 디렉터리 인덱싱
취약점 개요	
점검내용	■ 웹 서버 내 디렉터리 인덱싱 취약점 존재 여부 점검
점검목적	■ 디렉터리 인덱싱 취약점을 제거하여 특정 디렉터리 내 불필요한 파일 정보의 노출을 차단
보안위협	■ 해당 취약점이 존재할 경우 브라우저를 통해 특정 디렉터리 내 파일 리스트를 노출하여 응용시스템의 구조를 외부에 허용할 수 있고, 민감한 정보가 포함된 설정 파일 등이 노출될 경우 보안상 심각한 위험을 초래할 수 있음
참고	※ **디렉터리 인덱싱 취약점:** 특정 디렉터리에 초기 페이지 (index.html, home.html, default.asp 등)의 파일이 존재하지 않을 때 자동으로 디렉터리 리스트를 출력하는 취약점
점검대상 및 판단기준	
대상	■ 웹 서버
판단기준	**양호 :** 디렉터리 파일 리스트가 노출되지 않는 경우
	취약 : 디렉터리 파일 리스트가 노출되는 경우
조치방법	웹 서버 설정을 변경하여 디렉터리 파일 리스트가 노출되지 않도록 설정

웹(Web)

IL (상)	9. 정보 누출
취약점 개요	
점검내용	■ 웹 서비스 시 불필요한 정보가 노출되는지 여부 점검
점검목적	■ 웹 서비스 시 불필요한 정보가 노출되는 것을 방지함으로써 2차 공격에 활용될 수 있는 정보 노출을 차단하기 위함
보안위협	■ 웹 사이트에 중요정보(개인정보, 계정정보, 금융정보 등)가 노출되거나 에러 발생 시 과도한 정보(애플리케이션 정보, DB 정보, 웹 서버 구성 정보, 개발 과정의 코멘트 등)가 노출될 경우 공격자들의 2차 공격을 위한 정보로 활용될 수 있음
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	■ 웹 애플리케이션 소스코드, 웹 서버
판단기준	**양호 :** 웹 사이트에 중요정보가 노출되지 않고, 에러 발생 시 과도한 정보가 노출되지 않는 경우
	취약 : 웹 사이트에 중요정보가 노출되거나, 에러 발생 시 과도한 정보가 노출되는 경우
조치방법	웹 사이트에 노출되는 중요정보는 마스킹을 적용하여야 하며, 발생 가능한 에러에 대해 최소한의 정보 또는 사전에 준비된 메시지만 출력함

참고 자료

2021 주요정보통신기반시설 기술적 취약점 분석/평가 방법 상세 가이드

웹(Web)

FD (상)	23. 파일 다운로드
취약점 개요	
점검내용	■ 웹 사이트에서 파일 다운로드 시 허용된 경로 외 다른 경로의 파일 접근이 가능한지 여부 점검
점검목적	■ 파일 다운로드 시 허용된 경로 외 다른 경로의 파일 접근을 방지하여 공격자가 임의의 위치에 있는 파일을 열람하거나 다운받는 것을 불가능하게 하기 위함
보안위협	■ 해당 취약점이 존재할 경우 공격자는 파일 다운로드 시 애플리케이션의 파라미터 값을 조작하여 웹 사이트의 중요한 파일(DB 커넥션 파일, 애플리케이션 파일 등) 또는 웹 서버 루트에 있는 중요한 설정 파일(passwd, shadow 등)을 다운받을 수 있음 ■ cgi, jsp, php 등 파일 다운로드 기능을 제공해주는 애플리케이션에서 입력되는 경로를 검증하지 않는 경우 임의의 문자(../.. 등)나 주요 파일명의 입력을 통해 웹 서버의 홈 디렉터리를 벗어나서 임의의 위치에 있는 파일을 열람하거나 다운받는 것이 가능함
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	■ 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽
판단기준	**양호 :** 다운로드 파일이 저장된 디렉터리 이외에 접근이 불가능한 경우
	취약 : 다운로드 파일이 저장된 디렉터리 이외에 접근이 가능한 경우
조치방법	다운로드 시 허용된 경로 이외의 디렉터리와 파일에 접근할 수 없도록 구현

웹(Web)

AE (상)	24. 관리자 페이지 노출
취약점 개요	
점검내용	■ 유추하기 쉬운 URL로 관리자 페이지 및 메뉴 접근의 가능 여부 점검
점검목적	■ 관리자 페이지 URL이 유추하기 쉬운 이름(admin, manager 등) 및 웹 사이트 설계 오류를 수정하여 비인가자의 관리자 메뉴 접근을 방지하고자 함
보안위협	■ 웹 관리자의 권한이 노출될 경우 웹 사이트의 변조뿐만 아니라 취약성 정도에 따라서 웹 서버의 권한까지도 노출될 수 있음
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	■ 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽
판단기준	**양호 :** 유추하기 쉬운 URL로 관리자 페이지 접근이 불가능한 경우
	취약 : 유추하기 쉬운 URL로 관리자 페이지 접근이 가능한 경우
조치방법	유추하기 어려운 이름(포트 번호 변경 포함)으로 관리자 페이지를 변경하여 비인가자가 관리자 페이지에 접근할 수 없도록 하고 근본적인 해결을 위해 지정된 IP만 관리자 페이지에 접근할 수 있도록 제한하여야 함 단, 부득이하게 관리자 페이지를 외부에 노출해야 하는 경우 관리자 페이지 로그인 시 2차 인증(otp, vpn, 인증서 등) 적용 필요함

참고 자료

2021 주요정보통신기반시설 기술적 취약점 분석/평가 방법 상세 가이드

웹(Web)

PT (상)	25. 경로 추적
취약점 개요	
점검내용	■ 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리의 접근 통제 여부 점검
점검목적	■ 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터의 접근 및 실행을 방지하고자 함
보안위협	■ 웹 서버와 웹 애플리케이션의 파일 또는 디렉터리 접근이 통제되지 않아 웹 서버 또는 웹 애플리케이션의 중요한 파일과 데이터에 접근을 허용하는 취약점으로 웹 루트 디렉터리에서 외부의 파일까지 접근하여 이를 실행할 수 있음
참고	※ 소스코드 및 취약점 점검 필요
점검대상 및 판단기준	
대상	■ 웹 애플리케이션 소스코드, 웹 서버, 웹 방화벽
판단기준	**양호** : 웹 루트 디렉터리보다 상위 디렉터리(예. /root)에 접근이 불가능한 경우
	취약 : 웹 루트 디렉터리보다 상위 디렉터리에 접근이 가능한 경우
조치방법	사용자가 임의로 접근할 수 있는 최상위 디렉터리를 웹 루트 디렉터리로 설정하여 웹 서버의 시스템 루트 디렉터리로 접근하지 못하게 제한

웹(Web)

PL (상)	26. 위치 공개
취약점 개요	
점검내용	■ 예측 가능한 폴더의 위치 사용 여부 및 불필요한 파일의 존재 여부 점검
점검목적	■ 공격자가 폴더의 위치를 예측하여 파일 및 정보 획득을 방지하고자 함
보안위협	■ 폴더나 파일명의 위치가 예측 가능하여 쉽게 노출될 경우 공격자는 이를 악용하여 대상에 대한 정보를 획득하고 민감한 데이터에 접근 가능
참고	-
점검대상 및 판단기준	
대상	■ 웹 서버
판단기준	**양호** : 불필요한 파일이 존재하지 않고, 샘플 페이지가 존재하지 않을 경우
	취약 : 불필요한 파일이 존재하거나, 샘플 페이지가 존재하는 경우
조치방법	웹 루트 디렉터리 이하 모든 불필요한 파일 및 샘플 페이지 삭제

참고 자료

2021 주요정보통신기반시설 기술적 취약점 분석/평가 방법 상세 가이드

UNIX 서버

U-18 (상)	2. 파일 및 디렉토리 관리 > 2.14 접속 IP 및 포트 제한
취약점 개요	
점검내용	■ 허용할 호스트에 대한 접속 IP 주소 제한 및 포트 제한 설정 여부 점검
점검목적	■ 허용한 호스트만 서비스를 사용하게 하여 서비스 취약점을 이용한 외부자 공격을 방지하기 위함
보안위협	■ 허용할 호스트에 대한 IP 및 포트제한이 적용되지 않은 경우, Telnet, FTP같은 보안에 취약한 네트워크 서비스를 통하여 불법적인 접근 및 시스템 침해 사고가 발생할 수 있음
참고	▪ 접속 IP 및 포트제한 애플리케이션 종류 예시 ※ **TCP Wrapper:** 네트워크 서비스에 관련한 트래픽을 제어하고 모니터링 할 수 있는 UNIX 기반의 방화벽 툴 ※ **IPFilter:** 유닉스 계열에서 사용하는 공개형 방화벽 프로그램으로써 Packet Filter로 시스템 및 네트워크 보안에 아주 강력한 기능을 보유한 프로그램 ※ **IPtables:** 리눅스 커널 방화벽이 제공하는 테이블들과 그것을 저장하는 체인, 규칙들을 구성할 수 있게 해주는 응용프로그램
점검대상 및 판단기준	
대상	■ SOLARIS, LINUX, AIX, HP-UX 등
판단기준	**양호 :** 접속을 허용할 특정 호스트에 대한 IP 주소 및 포트 제한을 설정한 경우
	취약 : 접속을 허용할 특정 호스트에 대한 IP 주소 및 포트 제한을 설정하지 않은 경우
조치방법	OS에 기본으로 제공하는 방화벽 애플리케이션이나 TCP Wrapper와 같은 호스트별 서비스 제한 애플리케이션을 사용하여 접근 허용 IP 등록

UNIX 서버

U-38 (상)	3. 서비스 관리 > 3.20 웹서비스 불필요한 파일 제거
취약점 개요	
점검내용	■ Apache 설치 시 기본으로 생성되는 불필요한 파일의 삭제 여부 점검
점검목적	■ Apache 설치 시 디폴트로 설치되는 불필요한 파일을 제거함을 목적으로 함.
보안위협	■ Apache 설치 시 htdocs 디렉터리 내에 매뉴얼 파일은 시스템 관련정보를 노출하거나 해킹에 악용될 수 있음
참고	※ 불필요한 파일: 샘플 파일, 매뉴얼 파일, 임시 파일, 테스트 파일, 백업 파일 등
점검대상 및 판단기준	
대상	■ SOLARIS, LINUX, AIX, HP-UX 등
판단기준	**양호 :** 기본으로 생성되는 불필요한 파일 및 디렉터리가 제거되어 있는 경우
	취약 : 기본으로 생성되는 불필요한 파일 및 디렉터리가 제거되지 않은 경우
조치방법	불필요한 파일 및 디렉터리 제거 ("/[Apache_home]/htdocs/manual", "/[Apache_home]/manual" 파일 제거 등)

감사합니다.

Made by 박수현

Email : qkrtngus211@naver.com

URL : https://github.com/zjsl3784/webscan_v1