웹 취약점 자동화 진단 도구 제작

팀　　명 : 저희가할수있겠조

지도 교수 :　양환석 교수님

팀　　장 :　　이승재

팀　　원 :　　범채윤

이경서

이정현

이진솔

한지호

2022. 11.

중부대학교 정보보호학과

목 차

1. 서론

1.1. 연구 배경 및 필요성

코로나19 창궐 후 전 세계가 어려움을 겪는 가운데, 특히 사이버 상에서는 다양한 사건·사고가 발생했다. 랜섬웨어, APT 공격 등 다양한 공격들의 건수가 많아지는 가운데 웹 사이트도 꾸준한 타겟이 된다.

개인정보 유출이 빈번해짐에 따라 한국인터넷진흥원에 따르면 개인정보침해신고센터에 접수된 개인정보 침해 상담 개인 정보 침해 상담·신고 건수는 2016년 9만 8,210건에서 지난해 17만 7,457건으로 4년새 80% 넘게 급증했다. 이와 같이 지속적으로 증가하고 있는 개인정보 유출 사고는 해마다 급증 하고 있어 앞으로도 계속 증가할 것으로 보인다.

이러한 개인정보 유출의 이유 중 하나는, 웹 사이트 운영자의 실수로 인한 사고가 대부분이다. 이러 한 사례들은 단순 실수나 관리 미흡으로 발생하며, 이렇게 노출된 전보는 보이스피싱 등 금융범죄에 악용돼 더 큰 피해를 일으켜 사회적인 물의를 일이 킬 수 있다. 이러한 사고를 예방하기 위해서는 웹 사이트 관리자가 웹 사이트의 보안에 대해 경각심을 가지고 있어야 하며, 주기적인 취약점 진단은 필 수이다.

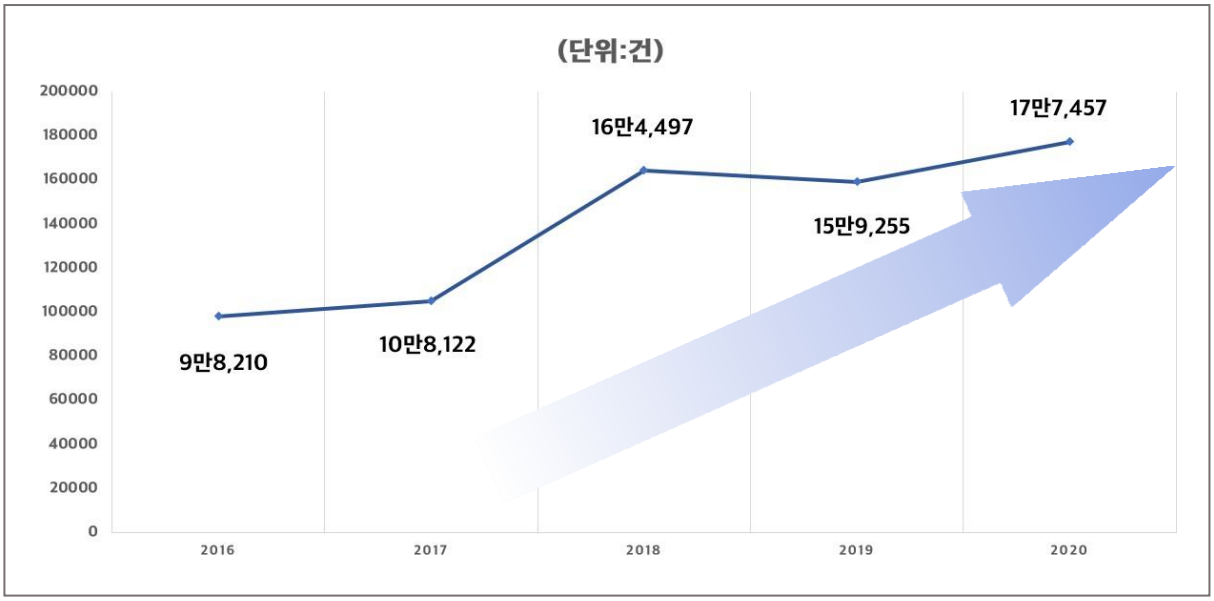


그림 1 개인정보 침해 신고•상담건수 (출처 : 한국인터넷진흥원)

2021년 상반기 과학기술정보통신부에서 230개 민간기업을 대상으로 사이버 위기대응 모의훈련을 진행했다. 실제 화이트 해커가 취약점을 이용해 기업 내부 시스템에 침투하고 대응하는 모의침투 훈련 결과, 30개 기업의 홈페이지에서 총 114개 취약점이 발견됐다. 아래와 같이 사이트 간 스크립팅, 파라미터 변조 및 조작이 가장 많이 발견된 취약점으로 나타났다.

웹으로 제공되는 서비스가 많아진 만큼 다양한 사이버 공격이 발생하고 그 발생률 또한 날이 갈수록 증가하고 있다. 그렇기 때문에 기업들과 개인들은 웹 보안에 경각심을 가져야 할 필요가 있다.

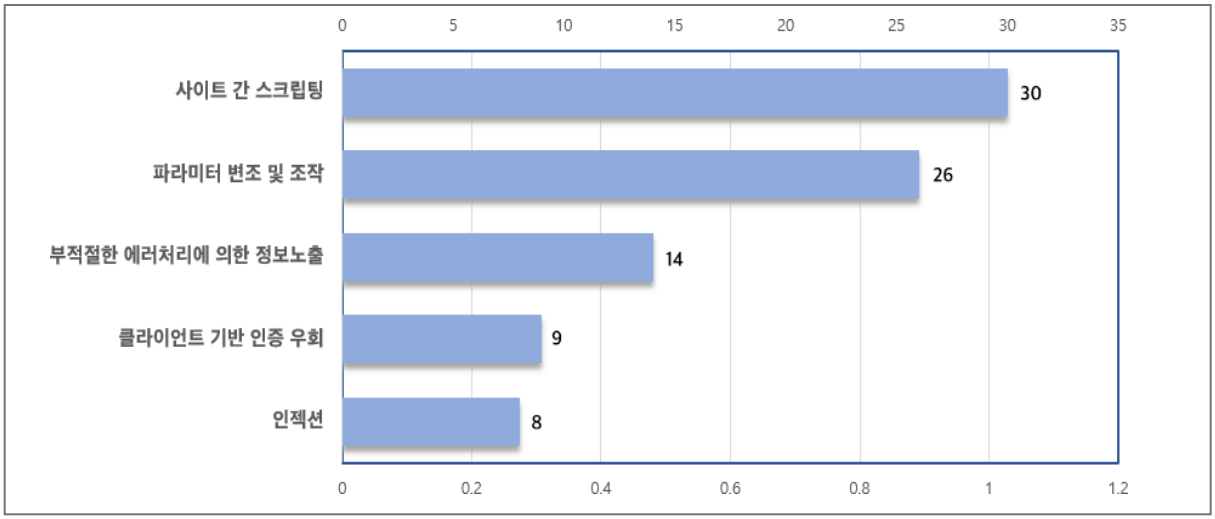


그림 2 2021 년 상반기 모의훈령 통해 발견된 취약점 항목(출처 : 과학기술정보통신부)

1.2. 연구 목적 및 주제 선정

최근 웹 취약점을 이용한 공격들이 지속적으로 발생되며 증가하고 있다. 웹서버나 웹사이트의 체계적인 보안 관리가 제대로 이루어지지 않은 회사가 다수이고, 웹사이트는 사용자들에게 항상 노출되어 있으며, 접근 방법이 매우 쉽기 때문에 보안 사고가 끊임없이 발생되고 있다.

전체 개인정보유출 피해의 75%가 외부 해킹이 원인이며, 가장 취약한 웹에서부터 시작된다. 이렇게 잦은 보안 사고로 개인정보보호법, 정보통신망법, 전자금융거래법 시행령, 신용정보법 등 법적 의무와 처벌이 강화되는 추세입니다.

개인이나 기업이 간단하게 웹 진단을 하기 위한 웹 취약점 자동 진단 도구를 제작한다. 이는 취약점 진단 시간을 단축하여 효율적인 진단이 가능하고, 사용자에게 친절한 진단 보고서와 조치방안을 제공한다.

정보통신기반 보호법에 따라 국가 사이버 안보 등을 고려하여 정보통신기반시설 중 전자적 침해 행위로부터의 보호 등이 필요하다고 인정되었다. 정부에서 특별히 지정한 시설인 주요정보통신기반시설 취약점 가이드 기반으로 진단 도구를 제작한다.

2. 관련 연구

2.1. 언어

2.1.1. Python

Python은 웹 애플리케이션, 소프트웨어 개발, 데이터 과학, 기계 학습(ML)에 널리 사용되는 프로그래밍 언어이다. 개발자는 Python이 효율적이고 배우기 쉬우며 여러 플랫폼에서 실행될 수 있으므로 Python을 사용한다. Python 소프트웨어는 무료로 다운로드할 수 있고, 모든 유형의 시스템과 원활하게 통합되며, 개발 속도를 증가시킨다.

프로젝트에서 필요한 주요 Python 모듈

- Selenium : 웹 애플리케이션 테스트를 위한 프레임워크이다. 웹에 하는 명령을 코드화 시켜서 작동시키며. 다양한 브라우저 작동을 지원하며 크롤링에도 활용된다. 현존하는 거의 모든 웹브라우저를 다양한 언어를 통해 제어 가능하다.
- PyPDF : PDF 문서를 다룰 수 있게 해주며, PdfFileReader(기존 PDF 읽기), PdfFileWriter(새 PDF 쓰기), PdfFileMerfer(새 PDF 쓰기)와 같이 다양한 기능들이 있다.
- FPDF : PHP만으로 PDF 파일을 생성할 수 있다.
- BeautifulSoup : HTML로부터 데이터를 추출하기 위해 사용할 수 있는 파싱된 페이지의 파스 트리를 만드는데, 이는 웹 스크래핑에 유용하다.
- Pyotp : OTP를 생성할 수 있다.

2.1.2. PHP

- 웹 서버에서 해석되는 스크립트 언어이다.
- 데이터베이스 연동을 편리하게 할 수 있다. (MySQL, mSQL, Oracle, Sybase 및 윈도우 ODBC 등 편리하게 연동 가능)
- 거의 모든 운영체제에서 구현이 가능하다.
- 코드 작성이 쉽고, 문법이 간단하다.
- 처리속도가 빠르다.
- 파일 업로드, 메일 전송 등의 기능은 자체적으로 지원한다.
- 문법이 C언어를 따르므로 간결하고, ASPs나 JSP에 비해 코드의 양을 많이 줄일 수 있다.
- DB 연결에 함수를 사용하기 때문에 직관적이고, 간결하다.
- 이미지를 동적으로 생성할 수 있다.
- XML, ZIP, PDF, 암호화 등에 관련된 다양한 함수를 지원한다.

2.1.2.1. 테스트 사이트 제작

PHP를 이용한 테스트 쇼핑몰 사이트 주요 기능

- Main 페이지

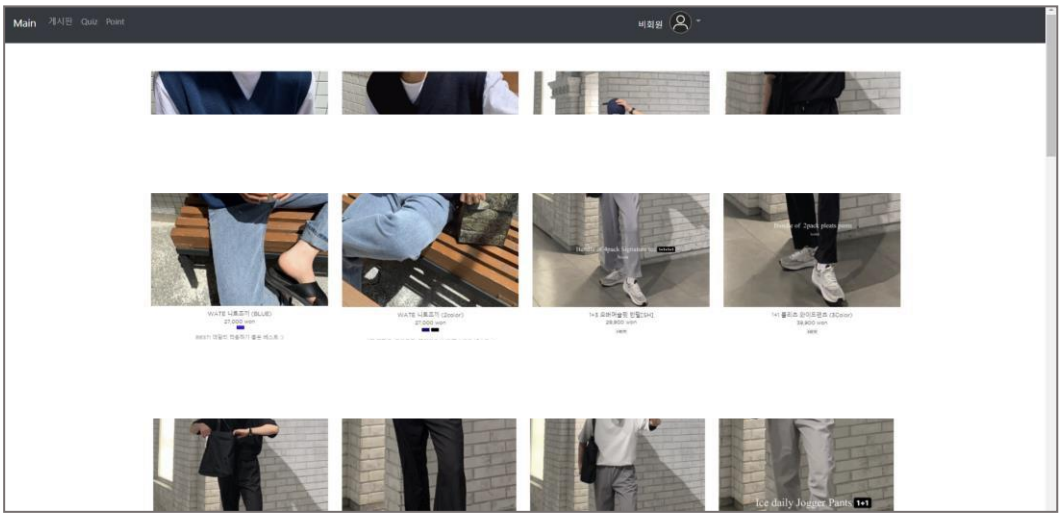


그림 3 PHP 쇼핑몰 사이트 메인페이지

- 회원가입 및 로그인 기능

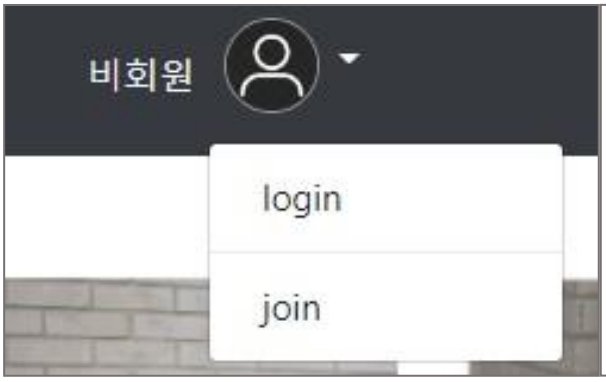


그림 4 로그인/회원가입 기능

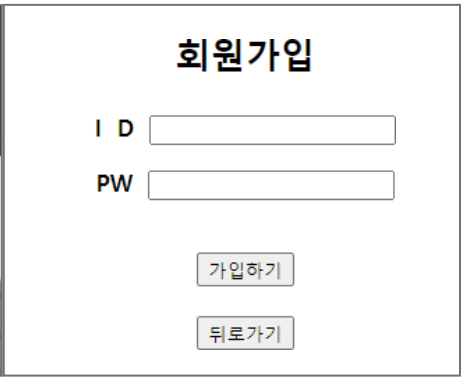
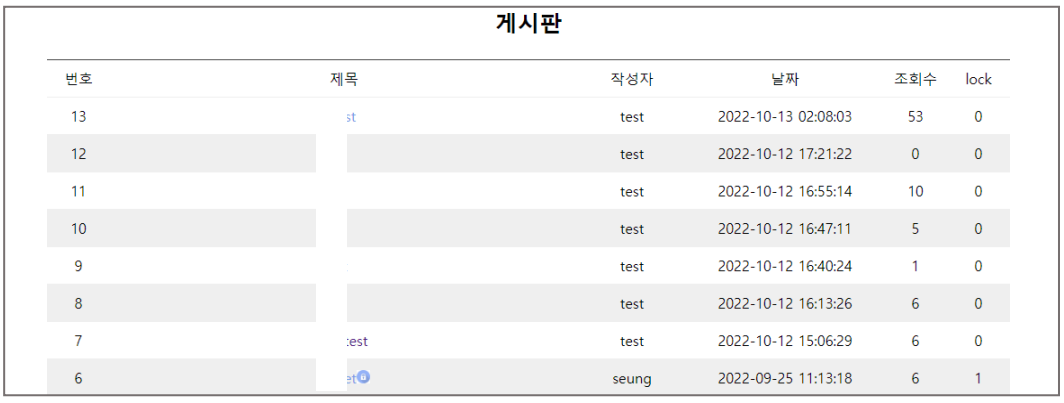


그림 5 로그인 화면

- 게시판 기능(파일 업로드)



게시판

번호	제목	작성자	날짜	조회수	lock
13	st	test	2022-10-13 02:08:03	53	0
12		test	2022-10-12 17:21:22	0	0
11		test	2022-10-12 16:55:14	10	0
10		test	2022-10-12 16:47:11	5	0
9		test	2022-10-12 16:40:24	1	0
8		test	2022-10-12 16:13:26	6	0
7	est	test	2022-10-12 15:06:29	6	0
6		seung	2022-09-25 11:13:18	6	1

그림 6 게시판 기능

그림 7 게시판 기능 – 파일업로드 가능

- 비밀글 기능

그림 8 비밀글 기능

- 검색 기능

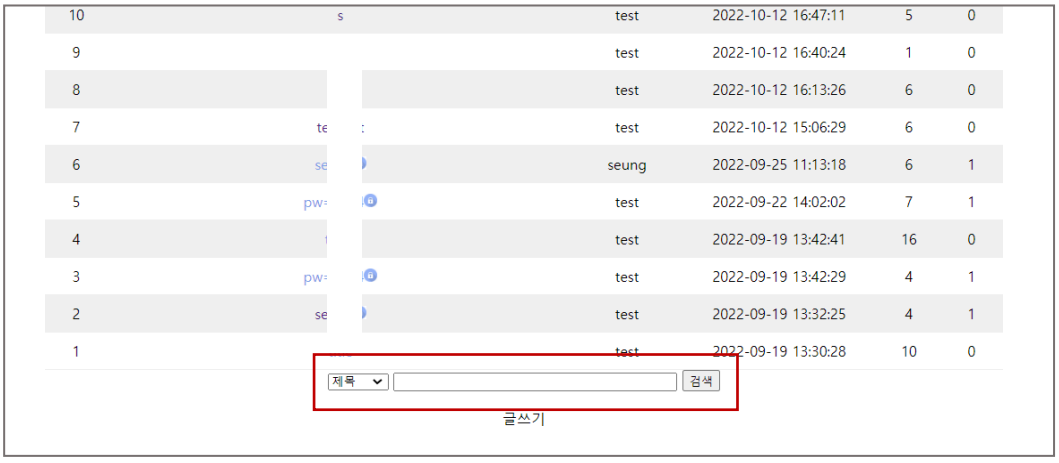


그림 9 검색 기능

2.2. 웹 취약점

2.2.1. LDAP Injection

사용자 입력을 기반으로 LDAP(Lightweight Directory Access Protocol)구문을 구축하여 웹 기반 응용 프로그램을 악용하는 데 사용되는 공격이다.

응용 프로그램이 사용자 입력 값에 대한 적절한 필터링 및 유효성 검증을 하지 않아 공격자는 로컬 프록시를 사용함으로 LDAP 문의 변조가 가능하다. 공격 성공 시 승인되지 않은 쿼리에 권한을 부여하고, LDAP 트리 내의 내용 수정이나 임의의 명령 실행을 가능하게 하므로 적절한 필터링 로직을 구현하여야 한다.

2.2.2. SQL Injection

사용자의 입력 값으로 웹 사이트 SQL 쿼리가 완성되는 약점을 이용하며, 입력 값을 변조하여 비정상적인 SQL 쿼리를 조합하거나 실행하는 공격이다.

개발자가 생각지 못한 SQL문을 실행되게 함으로써 데이터베이스를 비정상적으로 조작 가능하며, 해당 취약점이 존재하는 경우 비정상적인 SQL 쿼리로 DBMS 및 데이터(Data)를 열람하거나 조작 가능하므로 사용자의 입력 값에 대한 필터링을 구현하여야 한다.

2.2.3. XPath Injection

XML 구조에 악의적인 행위를 일으키는 내용을 삽입하거나 Xpath를 조작하여 XML의 내용을 노출하는 취약점이다.

해당 취약점이 존재할 경우 프로그래머가 의도하지 않았던 문자열을 전달하여 쿼리문의 의미를 왜곡시키거나 그 구조를 변경하고 임의의 쿼리를 실행하여 인가되지 않은 데이터를 열람할 수 있으므로 적절한 필터링 로직 구현이 필요하다.

2.2.4. XSS

악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법으로 공격 방식은 크게 stored 공격 방식과 reflected 공격 방식으로 나누어진다.

웹 애플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우, 공격자는 사용자 입력 값을 받는 게시판, URL 등에 악의적인 스크립트(Javascript, VBScript, ActiveX, Flash 등)를 삽입하여 게시글이나 이메일을 읽는 사용자의 쿠키(세션)를 탈취하여 도용하거나 악성코드 유포 사이트로 Redirect 할 수 있다.

2.2.5. 약한 문자열 강도

웹 사이트에서 취약한 패스워드로 회원가입이 가능할 경우 공격자는 추측 및 주변 정보를 수집하여 작성한 사전 파일로 대입을 시도하여 사용자 계정을 탈취할 수 있는 취약점이다.

해당 취약점 존재 시 유추가 용이한 계정 및 패스워드의 사용으로 인한 사용자 권한 탈취 위험 이 존재하며, 해당 위험을 방지하기 위해 값의 적절성 및 복잡성을 검증하는 로직을 구현하여야 한다.

2.2.6. 불충분한 인증

민감한 데이터에 취약한 인증 절차가 취약한 경우 나타나는 취약점이다.

중요정보(개인정보 변경 등) 페이지에 대한 인증 절차가 불충분할 경우 권한이 없는 사용자가 중요정보 페이지에 접근하여 정보를 유출하거나 변조할 수 있으므로 중요정보 페이지에는 추가적인 인증 절차를 구현하여야 한다.

2.2.7. 불충분한 인가

페이지 접근을 위한 인증기능이 구현되지 않을 경우, 인가되지 않는 사용자가 페이지에 접근 및 중요 정보의 변조를 할 수 있는 취약점이다.

접근제어가 필요한 중요 페이지의 통제수단이 미흡한 경우, 비인가자가 URL 파라미터 값 변경 등의 방법으로 중요 페이지에 접근하여 민감한 정보 열람 및 변조 가능하다.

2.2.8. 세션 고정

로그인 시 발급된 세션 ID가 전과 동일한 취약점이다.

사용자 로그인 시 항상 일정하게 고정된 세션 ID가 발행되는 경우 세션 ID를 도용한 비인가자의 접근 및 권한 우회가 가능하다.

2.2.9. 자동화 공격

웹사이트를 다운시키거나 무차별 대입 공격으로 인해 사용자 계정을 탈취할 수 있거나, 데이터 를 등록할 수 있는 취약점이다.

웹 애플리케이션의 특정 프로세스에 대한 반복적인 요청을 통제하지 않을 경우 무차별 대입 공격으로 인해 사용자 계정을 탈취할 수 있고, 자동화 공격으로 게시글 등록 또는 SMS 발송 요청을 반복하여 웹 애플리케이션 자원을 고갈시킬 수 있다.

2.2.10. 프로세스 검증 누락

인증이 필요한 페이지에 대해 인가된 인원인지를 확인하는 기능이 존재하지 않는 경우에 해당 정보를 변조하거나 탈취할 수 있는 취약점이다.

인증이 필요한 웹 사이트의 중요(관리자 페이지, 회원변경 페이지 등) 페이지에 대한 접근 제어가 미흡할 경우 하위 URL 직접 접근, 스크립트 조작 등의 방법으로 중요한 페이지에 대한 접근이 가능하다.

2.2.11. 파일 다운로드

웹에서 파일 다운로드 시 파일의 경로 및 파일명을 파라미터로 받아 처리하는 경우 이를 적절히 필터링 하지 않으면 공격자가 이를 조작하여 허용되지 않은 파일을 다운 받을 수 있고 임의의 위치에 있는 파일을 열람하거나 다운받는 것을 가능케 하는 취약점이다.

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 중요

한 설정 파일(passwd, shadow 등)을 다운받을 수 있다.

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 가능하다.

3. 본론

3.1. 시스템 구성도

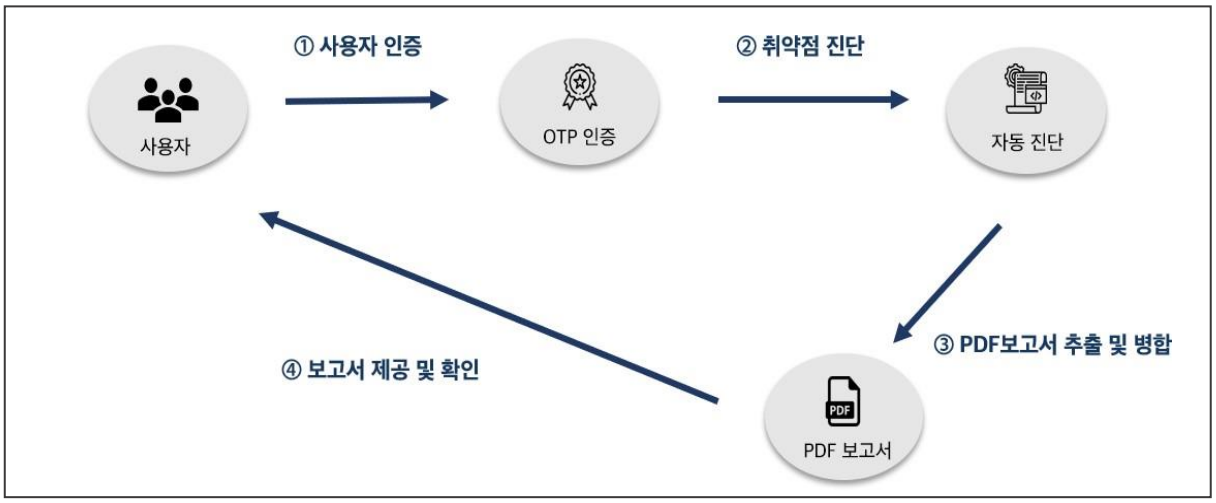


그림 10 시스템 구성도

① 사용자 인증 : OTP 메일을 통해 사용자 인증 한다.

② 취약점 진단 : 기업이나 개인은 해당 프로그램을 사용하여 취약점 자동 진단을 실시한다.

③ PDF 보고서 추출 및 병합 : 취약점 보고서 파일과 조치방안 보고서를 추출한다.

④ 보고서 제공 및 확인 : 사용자의 이메일로 전송된 보고서를 확인한다.

3.2. 웹 취약점 자동 진단 도구

해당 웹 취약점 자동화 진단 도구는 주요정보통신기반시설 기술적 취약점 분석평가 방법 상세가이드를 토대로 개발했다.

진단 항목은 LDAP Injection, SQL Injection, XPath Injection, XSS, 약한 문자열 강도, 불충분한 인증, 불충분한 인가, 세션 고정, 자동화 공격, 프로세스 검증 누락, 파일 다운로드, 관리자 페이지 노출로 총 12가지 취약점은 진단한다.

해당 도구는 취약점을 진단하는 도구로써, 웹 해킹 공격 기법을 시도한다. 따라서 인증되지 않은 사용자가 진단을 허가하지 않은 사이트에 도구를 사용할 수 없도록 사용자 인증을 해야 한다. 사용자 인증은 사용자 Email로 OTP 번호를 보내서 OTP 인증을 받아야 한다.

3.3. 진단 기준

3.3.1. 인젝션(LDAP, SQL, XPath)

Injection 취약점의 경우 Payload를 작성하여 해당 구문으로 사용자 권한, 로그인 성공 여부 등으로 공격 성공, 실패 여부를 판단한다.

```
def LI(domain): #LDAP Injection(3)
    Inspection_Items = "LDAP Injection(3)"
    contents = ""
    cve = "Safety"

    urls = domain+"/login.php"

    XPATH_id = "/html/body/div/form/p[1]/input"
    XPATH_pw = "/html/body/div/form/p[2]/input"
    XPATH_click = "/html/body/div/form/input"

    lines = [
            "*",
            "admin)(&))",
            "*)(&",
            ")(cn=*))",
            "*()|&'",
            "*(|(objectclass=*))",
            "*)(uid=*))(|(uid=*",
            "admin*)((|userpassword=*)"
            ]

    count = 0
    for payload in lines:
        driver.get(urls)
        input_box = driver.find_element(By.XPATH, XPATH_id)
```

그림 11 LDAP Injection 소스코드 일부

3.3.2. XSS

XSS 취약점의 경우 reflected XSS 공격 기법으로 진단했다. Payload 에 공격 구문을 작성하고 공격을 시도했을 때, 공격에 대한 반응이 나타나면 취약하다고 판단했다.

```
def XS(domain): # XSS(11)
    Inspection_Items = "XSS(11)"
    contents = ""
    cve = "Safety"

    urls = domain+"/board.php"
    lines = ["<script>alert('XSS Risk')</script>",
             '"><script>alert(XSS Risk)</script>']
    count = len(lines)
    for payload in lines:
        try:
            driver.get(urls)
            input_box = driver.find_element(By.NAME, "search")
            input_box.send_keys(payload)
            driver.find_element(By.XPATH, '/html/body/div/form/button').click()
        except UnexpectedAlertPresentException:
            time.sleep(2)
```

그림 12 XSS 소스코드 일부

3.3.3. 약한 문자열 강도

약한 문자열 강도 취약점의 경우 payload 에 유추하기 쉬운 아이디와 패스워드를 작성한 후, 모든 값을 입력하여 로그인 성공, 실패 여부를 판단한다.

```
def BF(domain): # 약한 문자열 강도(12)
    Inspection_Items = "BF(12)"
    contents = ""
    cve = "Safety"

    urls = domain+"/login.php"

    XPATH_id = "/html/body/div/form/p[1]/input"
    XPATH_pw = "/html/body/div/form/p[2]/input"
    XPATH_click = "/html/body/div/form/input"

    idz = ["adminstrator", "manager","guest", "admin", "test"]

    passwds = ["Abcd",
               "aaaa",
               "admin",
               "test",
               "1234",
               "1111",
               "password"]

    for i in idz:
        driver.get(urls)
        input_box = driver.find_element(By.XPATH, XPATH_id)
        input_box.send_keys(i)
```

그림 13 약한 문자열 강도 소스코드 일부

3.3.4. 불충분한 인증

추측 가능한 아이디/패스워드를 입력했을 시, 로그인 가능한 경우 취약하다 판단

```
def IA(domain): # 불충분한 인증(13)
    Inspection_Items = "IA(13)"
    contents = ""
    cve = "Safety"

    urls = "http://"+domain+"/login.php"

    XPATH_id = "/html/body/div/form/p[1]/input"
    XPATH_pw = "/html/body/div/form/p[2]/input"
    XPATH_click = "/html/body/div/form/input"
    XPATH_Mypage = '//*[@id="collapsibleNavbar"]/ul/li[6]/div/a[4]'
    X = '//*[@id="navbarDropdown"]'
    count = 0

    driver.get(urls)
    input_box = driver.find_element(By.XPATH, XPATH_id)
    input_box.send_keys("admin")
    input_box2 = driver.find_element(By.XPATH, XPATH_pw)
    input_box2.send_keys("admin")
    driver.find_element(By.XPATH, XPATH_click).click()

    try:
        driver.find_element(By.XPATH, X).click()
        driver.find_element(By.XPATH, XPATH_Mypage).click()

    except UnexpectedAlertPresentException:
        time.sleep(1)
        count += 1
```

그림 14 불충분한 인증 소스코드 일부

3.3.5. 불충분한 인가

불충분한 인가 취약점의 경우 다른 사용자의 비밀글에 인증을 하지 않고 접근할 수 있는지에 대한 판단으로, 공개글에서 비밀글로 넘어갈 때 URL 에서 파라미터 값을 변조하여 공격 성공, 실패 여부를 판단했다.

```
def IN(domain): #불충분한 인가(17)
    Inspection_Items = "IN(17)"
    contents = "This Website \"Risk\" from IN"
    cve = "Risk"
    msg = "불충분한 인가 취약"

    urls = domain+"/board.php"
    sourcecode = urllib.request.urlopen(urls).read()
    soup = BeautifulSoup(sourcecode, "html.parser")
    li=[0 for i in range(3)]

    for href in soup.find("tr", class_="even").find_all("tbody"):
        attr = href.find("a")["href"]
        if "number" in attr:
            num = li.split('=')
            li.append(num[1])
```

그림 15 불충분한 인가 소스코드 일부

3.3.6. 세션 고정

세션 고정 취약점은 로그인 시 발급받은 세션 ID 가 로그인 전/후 모두 동일하게 사용된 경우 취약하다고 판단한다. 사용자 로그인 시 고정된 세션 ID 가 발급되는 경우 비인가자의 접근 및 권한 우회가 가능한 취약점이 발생한다.

```
def SF(domain): # 세션 고정(19)
    Inspection_Items = "SF(19)"
    contents = "This website is \"SAFETY\" from SF"
    cve = "Safety"

    urls = domain+"/login.php"

    XPATH_id = "/html/body/div/form/p[1]/input"
    XPATH_pw = "/html/body/div/form/p[2]/input"
    XPATH_click = "/html/body/div/form/input"
    XPATH_nav = '//*[@id="navbarDropdown"]'
    XPATH_logout = '//*[@id="collapsibleNavbar"]/ul/li[5]/div/a[1]'

    driver.get(urls)
    driver.maximize_window()
    input_box = driver.find_element(By.XPATH, XPATH_id)
    input_box.send_keys('test')
    input_box2 = driver.find_element(By.XPATH, XPATH_pw)
    input_box2.send_keys('test')
    driver.find_element(By.XPATH, XPATH_click).click()
    alert = driver.switch_to.alert
    alert.accept()

    for cookie in driver.get_cookies():
```

그림 16 세션 고정 소스코드 일부

3.3.7. 자동화 공격

자동화 공격 취약점은 반복적인 대입 공격을 진행하여 로그인 실패가 5 회 이상이 넘어가면 취약하다고 판단했다.

```
def AD(domain): # 자동화 공격(20)
    Inspection_Items = "Auto Attack(20)"
    contents = "This Website is \"SAFETY\" from Auto Attack"
    cve = "Safety"

    urls = domain+"/login.php"

    XPATH_id = "/html/body/div/form/p[1]/input"
    XPATH_pw = "/html/body/div/form/p[2]/input"
    XPATH_click = "/html/body/div/form/input"
    count = 0
    for payload in range(5):
        driver.get(urls)
        input_box = driver.find_element(By.XPATH, XPATH_id)
        input_box.send_keys("admin")
        input_box2 = driver.find_element(By.XPATH, XPATH_pw)
        input_box2.send_keys('aaaa1234!@')
        driver.find_element(By.XPATH, XPATH_click).click()
        alert = driver.switch_to.alert
        alert_text = alert.text
        alert.accept()
        if "확인해주세요" in alert_text:
```

그림 17 자동화 공격 소스코드 일부

3.3.8. 프로세스 검증 누락

프로세스 검증 누락 취약점은 인증이 필요한 페이지에 대해 비인가자가 접근할 경우 로그인 페이지로 보내는지 확인한다. 로그인 하지 않고 직접 접근이 가능한 경우 취약하다고 판단했다.

```
def PV(): #프로세스 검증 누락(21)
    Inspection_Items = "PV(21)"
    contents = "This Website is \"SAFETY\" from PV"
    cve = "Safety"

    f = open("./payload/url.txt",'r')
    line = f.readlines()
    length = []
    i = 0
    cnt = 0
    for u in line:
        line[i] = line[i].strip('\n')
        i=i+1

    for url in line:
        response = requests.get(url = url)
        status = response.status_code
        res = response.text
        length.append(len(res))
```

그림 18 프로세스 검증 누락 소스코드 일부

3.3.9. 파일 다운로드

파일 다운로드 취약점의 경우 파일을 다운로드할 때 패킷에서 해당 파일의 경로를 다른 파일의 경로로 바꾸었을 때 파일이 다운로드지에 대한 성공, 실패 여부로 판단했다.

```
def FD(domain): #파일 다운로드(23)
    Inspection_Items = "File Download(23)"
    contents = "This Website is \"SAFETY\" from File Downloads"
    cve = "Safety"

    urls = domain+"/board.php"
    driver.get(urls)
    a_tag = driver.find_element(By.TAG_NAME, "a")
    href_text = a_tag.get_attribute('href')
    urls2 = href_text
    driver.get(urls2)
    a = driver.find_element(By.TAG_NAME, "a")
    href = a.get_attribute('href')

    down_url = href[:-1]
    down_url2 = down_url + "../../../../../../../../etc/passwd"
    driver.get(down_url2)
    time.sleep(1)
```

그림 19 파일 다운로드 소스코드 일부

3.3.10. 관리자 페이지 노출

관리자 페이지 노출 취약점은 URL 경로에 유추하기 쉬운 경로를 payload 로 작성하여 해당 페이지의 HTTP 상태 코드 값으로 공격 성공, 실패 여부를 판단했다.

```
def AE(domain): # 관리자 페이지 노출(24)
    Inspection_Items = "Admin Page Exposure(24)"
    contents = "This Website is \"SAFETY\" from Admin Page Exposure"
    cve = "Safety"

    page= ["/admin", "/manager", "/master", "/system", "/administart", "/admin/admin.php"]
    urls = domain
    for pages in page:
        try:
            res = urlopen(urls+pages)
            if res.status == 200 :
                cve = "Risk"
                writer.addPage(report.getPage(68))
                writer.addPage(report.getPage(69))
                writer.addPage(report.getPage(70))
                print("Admin_Page 경로 취약 --> ", pages)
                contents = urls
                return (Inspection_Items, contents.strip(), cve)
            else :
                print("Admin_Page 경로 안전")
                return (Inspection_Items, contents.strip(), cve)
        except HTTPError as e:
            err = e.read()
            code = e.getcode()
            if code != 200 : continue #print(code) ## 404
```

그림 20 관리자 페이지 노출 소스코드 일부

4. 결론

4.1. 실행 화면

① 실행 화면

Main.py 실행 시 배너와 주의사항, 진단 도구 서비스를 출력한다.

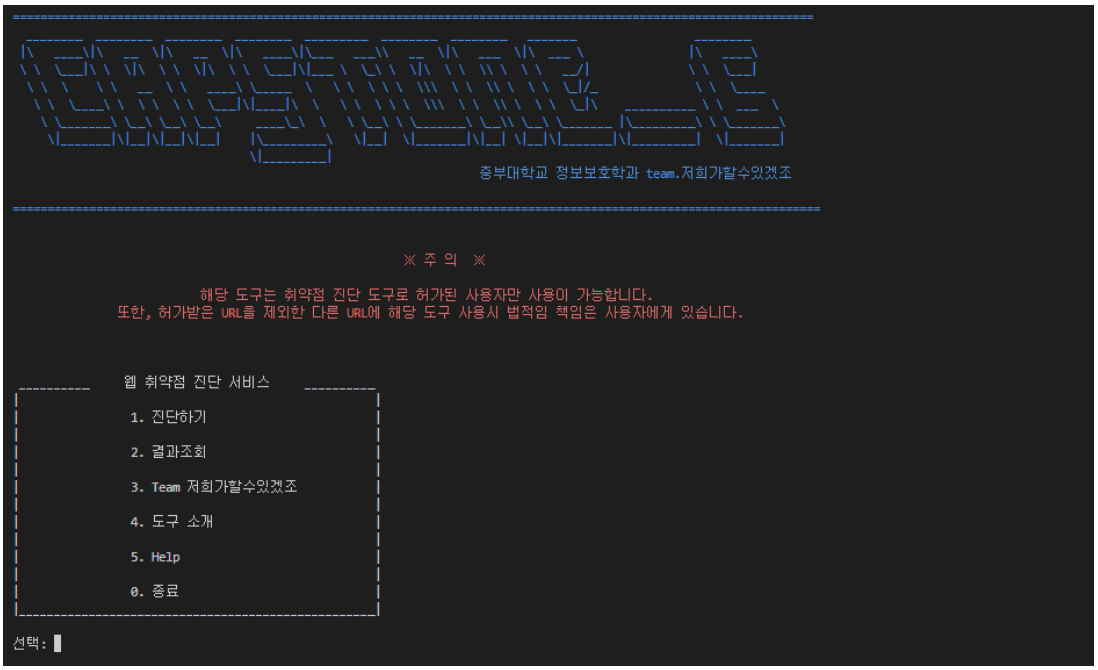


그림 21 실행화면

② Select : 1 (진단하기)

진단을 시작하면 진단 대상의 URL을 입력하고, 사용자 Email을 통해 OTP 인증을 받아야 한다.

이후, 주요정보통신기반시설 웹 취약점 가이드에 속해 있는 12가지 항목에 대한 진단을 시작하고, 결과와 조치방안에 대한 PDF 보고서가 제공된다.

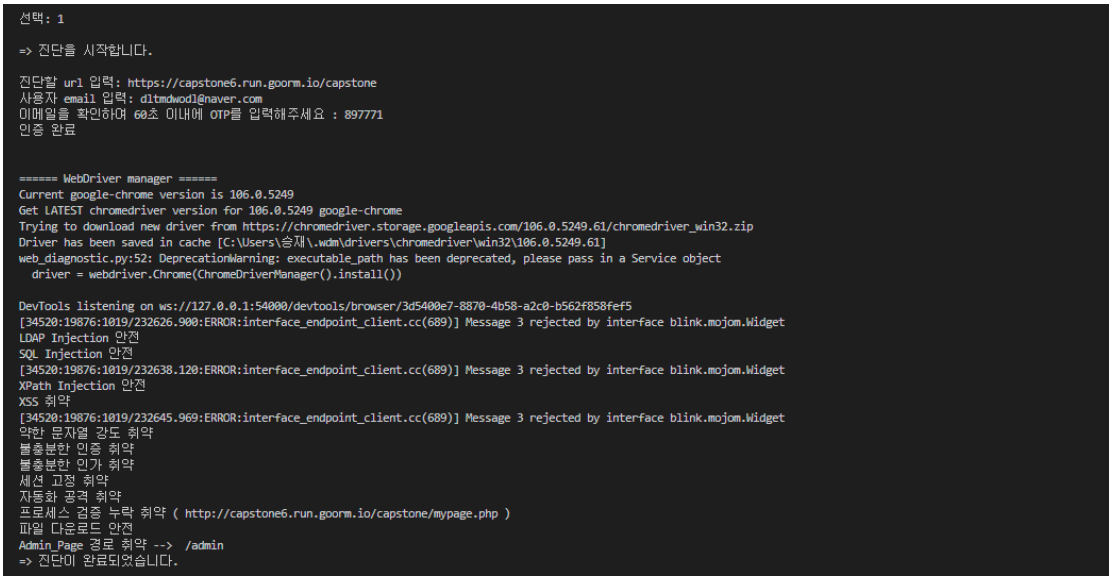


그림 22 실행화면-1

그림 23 실행화면-1(OTP)

③ select : 2 (결과 조회)

사용자가 원하는 경우, 진단 결과에 대한 보고서를 사용자 Email로 받을 수 있다.

그림 24 실행화면-2

그림 25 실행화면-2(보고서 제공)

④ Select : 3 (팀원소개)

도구 제작자를 소개한다.

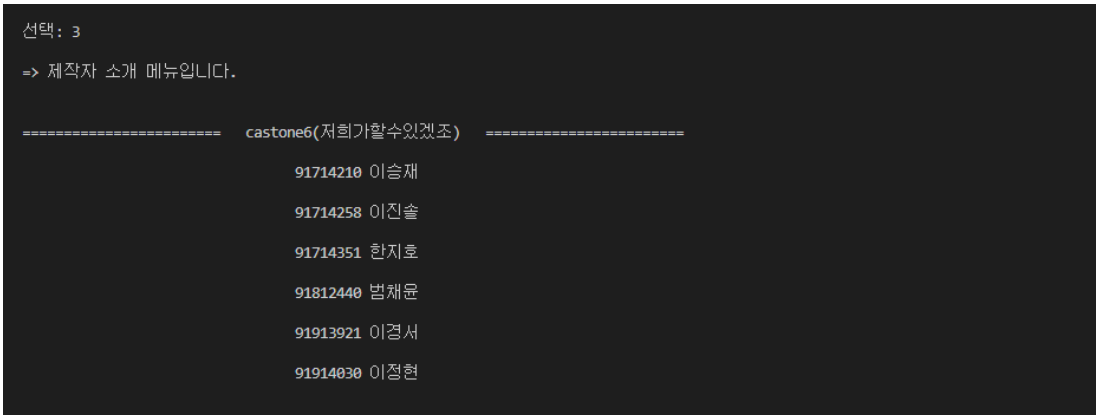


그림 26 실행화면-3

⑤ Select : 4 (도구 소개)

도구에 대한 설명과 점검 항목에 대한 설명이 포함되어있다.

그림 27 실행화면-4

⑥ Select : 5 (Help)

도구에 대한 사용 설명이 포함되어 있다.

그림 28 실행화면-5

⑦ Select : 0 (실행 종료)

4.2. 기대효과

- 취약점 점검 효율성 증대

  자동 진단 도구 활용으로 인한 취약점 진단 시간 단축
- 취약점 유형 연구

  식별된 취약점을 통한 주요정보통신기반 공격 유형 연구
- 빠른 취약점 위치 식별

  잠재적 취약점 발생 위치 파악 및 대처
- 학생들의 보안 역량 강화

  모의해킹 웹 사이트를 직접 제작하고 연구함으로써 학생들의 실습 경험 제공

5. 별첨

- github

  https://github.com/98sseung/capstone6

6. 참고

https://blog.lgcns.com/m/2787

https://www.boannews.com/media/view.asp?idx=104160

https://www.sharedit.co.kr/posts/5716

https://aws.amazon.com/ko/what-is/python/

https://lts0606.tistory.com/m/555