모의해킹점검 결과보고서

점검 개요

점검 항목

점검 항목은 OWASP TOP 10을 기반으로 작성된 주요 웹 취약점 항목입니다.

표 1. 점검항목

순번	점검항목	설명
1	SQL Injection	신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부로 삽입되어 인터프리터로 보내질 때 발생하는 취약점으로 공격자의 악의적인 데이터가 예기치 않은 명령을 실행하거나 권한 없이 데이터에 접근 가능 여부 점검
2	운영체제 명령실행	
3	취약한 인증	로그인 후 인증 또는 세션과 관련하여 공격자가 타 사용자의 권한 획득 가능 여부 점검
4	크로스 사이트 스크립팅 (XSS)	공격자가 입력한 스크립트가 타 이용자의 사용환경(브라우저, 웹뷰 등)에서의 실행 가능 여부를 점검
5	크로스사이트 요청변조 (CSRF)	공격자가 업로드 한 악의적인 스크립트로 인해 이용자의 권한 도용이 가능한지에 대한 점검
6	파일 업로드	웹쉘 등과 같은 악성파일이 업로드 될 경우 시스템 명령어 실행 및 인접 서버에 대한 침입 가능성이 존재함에 따라, 악성파일 업로드 및 실행 가능 여부를 점검
7	파일 다운로드	파일 다운로드 인터페이스를 이용하여 서버의 주요파일 다운로드 가능 여부 점검
8	디렉토리 목록 노출	인덱스 파일로 지정한 파일이 존재하지 않거나, 디렉토리 리스팅을 허락하도록 설정했을 경우 디렉토리 리스트가 출력됨에 따라 하위 파일 및 디렉토리 목록의 출력 여부를 점검
9	관리자 페이지 노출 여부	일반 이용자에 의한 관리자 페이지 접근 가능 여부를 점검
10	불필요한 파일 노출 여부	서버 운영상 불필요한 파일의 존재 여부를 점검

SQL Injection

High (Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index.php
Method	POST
Parameter	userpw
Attack	ZAP' OR '1'='1' --
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index.php
Method	POST
Parameter	userid
Attack	ZAP' OR '1'='1' --
Instances	2
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Other information	The page results were successfully manipulated using the boolean conditions [ZAP' AND '1'='1' --] and [ZAP' OR '1'='1' --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id	89
WASC Id	19
Source ID	1

Cross-Domain Misconfiguration

Medium (Medium)	Cross-Domain Misconfiguration
Description	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server
URL	https://location.services.mozilla.com/v1/country?key=7e40f68c-7938-4c5d-9f95-e61647c213eb
Method	GET
Evidence	Access-Control-Allow-Origin: *
Instances	1
Solution	Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
Other information	The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.
Reference	http://www.hpenterprisesecurity.com/vulncat/en/vulncat/vb/html5_overly_permissive_cors_policy.html
CWE Id	264
WASC Id	14
Source ID	3

X-Frame-Options Header Not Set

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index3.html
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/community.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/image_list.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/file_list.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index1.html
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/upload_proc.php
Method	POST
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/main.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/admin.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/info.php
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index2.html
Method	GET
Parameter	X-Frame-Options
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index.php

Cross-Domain Misconfiguration

Medium (Medium)	Cross-Domain Misconfiguration
Description	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server
URL	https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular-touch.min.js
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular-animate.min.js
Method	GET
Evidence	Access-Control-Allow-Origin: *
Instances	3
Solution	Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
Other information	The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.
Reference	http://www.hpenterprisesecurity.com/vulncat/en/vulncat/vb/html5_overly_permissive_cors_policy.html
CWE Id	264
WASC Id	14
Source ID	3

Medium (Medium)	Cross-Domain Misconfiguration
Description	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu4mxKKTU1Kg.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu5mxKKTU1Kvnz.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/googlesans/v16/4UabrENHsxJlGDuGo1OIlLU94YtzCwZsPF4o.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOlCnqEu92Fr1MmSU5fBBc4AMP6lQ.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu4WxKKTU1Kvnz.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/productsans/v12/pxiDypQkot1TnFhsFMOfGShVF9eOYktMqg.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/googlesans/v16/4UaGrENHsxJlGDuGo1OIlL3Owp5eKQtG.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu7WxKKTU1Kvnz.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
URL	https://fonts.gstatic.com/s/roboto/v20/KFOmCnqEu92Fr1Mu7GxKKTU1Kvnz.woff2
Method	GET
Evidence	Access-Control-Allow-Origin: *
Instances	9
Solution	Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
Other information	The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.

CSP Scanner: Wildcard Directive

Medium (Medium)	CSP Scanner: Wildcard Directive
Description	The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: style-src, style-src-elem, style-src-attr, img-src, connect-src, frame-src, frame-ancestors, font-src, media-src, manifest-src, prefetch-src
URL	https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/?tab%3Dwm%26ogbl&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1
Method	GET
Parameter	Content-Security-Policy
Evidence	script-src 'nonce-FWDL+s+cQZp3L9r+Hn3WmA' 'unsafe-inline' 'unsafe-eval';object-src 'none';base-uri 'self';report-uri /cspreport
Instances	1
Solution	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy http://content-security-policy.com/ https://github.com/shapesecurity/salvation
CWE Id	16
WASC Id	15
Source ID	3

X-Content-Type-Options Header Missing

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://detectportal.firefox.com/success.txt
Method	GET
Parameter	X-Content-Type-Options
URL	http://detectportal.firefox.com/success.txt?ipv6
Method	GET
Parameter	X-Content-Type-Options
URL	http://detectportal.firefox.com/success.txt?ipv4
Method	GET
Parameter	X-Content-Type-Options
Instances	3
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15
Source ID	3

Incomplete or No Cache-control and Progma HTTP Header Set

Low (Medium)	Incomplete or No Cache-control and Pragma HTTP Header Set
Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://versioncheck-bg.addons.mozilla.org/update/VersionCheck.php?reqVersion=2&id=%7Bc45c406e-ab73-11d8-be73-000a95be3b12%7D&version=2.0.5&maxAppVersion=undefined&status=userEnabled&appID=%7Bec8030f7-c20a-464f-9b0e-13a3a9e97384%7D&appVersion=79.0&appOS=Darwin&appABI=x86_64-gcc3&locale=en-US&currentAppVersion=79.0&updateType=112&compatMode=normal
Method	GET
Parameter	Cache-Control
Evidence	max-age=3600
URL	https://versioncheck-bg.addons.mozilla.org/update/VersionCheck.php?reqVersion=2&id=foxyproxy-basic@eric.h.jung&version=7.4.3&maxAppVersion=null&status=userEnabled&appID=%7Bec8030f7-c20a-464f-9b0e-13a3a9e97384%7D&appVersion=79.0&appOS=Darwin&appABI=x86_64-gcc3&locale=en-US&currentAppVersion=79.0&updateType=112&compatMode=normal
Method	GET
Parameter	Cache-Control
Evidence	max-age=3600
URL	https://versioncheck-bg.addons.mozilla.org/update/VersionCheck.php?reqVersion=2&id=wappalyzer@crunchlabz.com&version=6.0.15&maxAppVersion=undefined&status=userEnabled&appID=%7Bec8030f7-c20a-464f-9b0e-13a3a9e97384%7D&appVersion=79.0&appOS=Darwin&appABI=x86_64-gcc3&locale=en-US&currentAppVersion=79.0&updateType=112&compatMode=normal
Method	GET
Parameter	Cache-Control
Evidence	max-age=3600
URL	https://versioncheck-bg.addons.mozilla.org/update/VersionCheck.php?reqVersion=2&id=%7Ba6c4a591-f1b2-4f03-b3ff-767e5bedf4e7%7D&version=0.3.5&maxAppVersion=undefined&status=userEnabled&appID=%7Bec8030f7-c20a-464f-9b0e-13a3a9e97384%7D&appVersion=79.0&appOS=Darwin&appABI=x86_64-gcc3&locale=en-US&currentAppVersion=79.0&updateType=112&compatMode=normal
Method	GET
Parameter	Cache-Control
Evidence	max-age=3600
Instances	4
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Web_Content_Caching
CWE Id	525
WASC Id	13
Source ID	3

Cookie Without Same Site Attribute

Low (Medium)	Cookie Without SameSite Attribute
Description	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index.php
Method	POST
Parameter	user_id
Evidence	Set-Cookie: user_id
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/logout.php
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
Instances	3
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.
Reference	https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site
CWE Id	16
WASC Id	13
Source ID	3

Cookie No HttpOnly Flag

Low (Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/index.php
Method	POST
Parameter	user_id
Evidence	Set-Cookie: user_id
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
URL	http://ec2-54-180-137-70.ap-northeast-2.compute.amazonaws.com:8088/logout.php
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
Instances	3
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
CWE Id	16
WASC Id	13
Source ID	3

Absence of Anti_CSRF Tokens

Low (Medium)	Absence of Anti-CSRF Tokens
Description	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL	https://www.google.com/?gws_rd=ssl
Method	GET
Evidence	<form class="tsf nj" action="/search" style="overflow:visible" data-submitfalse="q" id="tsf" method="GET" name="f" role="search">
Instances	1