[별첨]

WebCastellum 테스트 케이스

2013. 07

nipa 정보통신산업진흥원 National IT Industry Promotion Agency
nipa 공개SW역량프라자 Open Source Software COMPENTENCE PLAZA

Stack A

순번	대분류	중분류	소분류	시나리오명	시나리오 개요	시나리오 흐름	케이스 번호	케이스	입력데이터	예상결과	결과	오류증상	비고
1	Monitering			모니터링	Debug로그 확인		1	WAS 구동시 tomcat 로그를 tail한다.	tail -f catalina.out	어플리케이션 로그가 실시간 모니터링된다.	pass		
2	SQL Injection			SQL Injection 취약점 검증	SQL인젝션 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	SQL인젝션 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 재질의한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸다.	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 재질의를 수행한다.	- http://[TEST_DOMAIN]/list1_list.do?searchkeyorg=1&searchtextorg='%20OR%201=1&pageNo=&listdiv=4&pagediv=&searchkey=1&searchtext='%20OR%201=1 -http://[TEST_DOMAIN]/page.jsp?select * from board where seq = 1 'Or 1' 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 HTTP 질의의 결과에서 SQL 인젝션 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
3	Cross Site Scripting (XSS)			Cross Site Scripting (XSS) 취약점 검증	크로스 사이트 스크립팅 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	크로스사이트스크립팅 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	http://[TEST_DOMAIN]/list1_list.do?searchkeyorg=1&searchtextorg=%22%3E%3Cscript%3Ealert%28%27XSS%27%29%3B%3C%2Fscript%3E&pageNo=&listdiv=4&pagediv=&searchkey=1&searchtext=%22%3E%3Cscript%3Ealert%28%27XSS%27%29%3B%3C%2Fscript%3E – http://[TEST_DOMAIN]/page.do?page=00602&seq	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 변조된 HTTP 질의의 결과에서 크로스사이트스크립팅 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
4	Cross-Site Request Forgery (CSRF) 취약점 검증			Cross-Site Request Forgery (CSRF) 취약점 검증	크로스 사이트 요청위조 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	크로스사이트 요청 위조 취약점을 검증할 수 있는 보조서버를 구성한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	http://[TEST_DOMAIN]/../../../../bin/ls%20-la http://[TEST_DOMAIN]/../../../../bin/rm%20-rf%20 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 변조된 HTTP 질의의 결과에서 CSRF 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
5	Command Injection			Command Injection 취약점 검증	운영체제명령실행 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	운영체제명령실행 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	http://[TEST_DOMAIN]/../../../../bin/ls%20-la http://[TEST_DOMAIN]/../../../../bin/rm%20-rf%20 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 변조된 HTTP 질의의 결과에서 운영체제 명령실행 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
6	Session fixation			Session fixation 취약점 검증	세션 고정 공격 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	세션 고정 공격 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- http://[TEST_DOMAIN]/images/ - http://[TEST_DOMAIN]/dokdoAd/ http://[TEST_DOMAIN]/admin/ http://[TEST_DOMAIN]/console/ - http://[TEST_DOMAIN]/mobile/ - http://[TEST_DOMAIN]/mobile/css/ 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 변조된 HTTP 질의의 결과에서 세션 고정 공격 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			

7	Directory Traversal				디렉토리 탐색 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	디렉토리 탐색 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다.	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는 지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- http://[TEST_DOMAIN]/images/ - http://[TEST_DOMAIN]/dokdoAd/ http://[TEST_DOMAIN]/admin/ http://[TEST_DOMAIN]/console/ - http://[TEST_DOMAIN]/mobile/ - http://[TEST_DOMAIN]/mobile/css/ 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 는 변조된 HTTP 질의의 결과에서 디렉토리 탐색 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
8	upload & filedownload attack				파일 다운로드 취약점을 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	파일 다운로드 취약점에 해당하는 CRS ID 를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는 지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- POST /filedown.jsp pagediv=filedownload&userfileorgname=../../../etc/passwd - POST /fileupload.jsp pagediv=fileupload&userfileorgname=../../../../etc/attack.jsp 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 변조된 HTTP 질의의 결과에서 파일다운로드&업로드 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
9	Parameter manipulation				파라메터 변조 취약점 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	파라메터 취약점에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는 지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- POST /boardview.jsp & pagenum=1 - POST /index.jsp& file=http://[TEST_DOMAIN]/main.html 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 변조된 HTTP 질의의 결과에서 파라메터 변조 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
10	CRLF Injection				Carrage Return Attack 취약점 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	Carrage Retun Attack에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는 지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- http://[Test_Domain]/index.jsp?page=%0d%0aContent-Type: text/html%0d%0aHTTP/1.1 200 OK%0d%0aContent-Type: text/html%0d%0a%0d%0a%3Chtml%3EHacker Content%3C/html%3E 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 변조된 HTTP 질의의 결과에서 파라메터 변조 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
11	Script Injection				스크립트 주입 공격 취약점 검증하기 위해서 변조된 HTTP 질의를 수행하고, 해당 취약점이 검증되는 지를 확인한다.	스크립트 주입 공격 에 해당하는 데이터를 적용한다. 기존 Tomcat 로그를 이용해 변조된 HTTP 질의를 수행한다. WebCastellum Debug로그를 통해서 해당 취약점의 WebCastellum 감지/대응 여부를 확인한다	1	Web apllication의 web.xml을 설정한다.	설정파일 편집	설정파일이 변경된다.	pass		
							2	Tomcat 필터 설정 적합성을 확인한다.	Tomcat 설정확인 명령	Tomcat 설정구문이 정상적으로 동작하는 지를 확인한다.			
							3	취약점 검증대상의 URL 을 Tomcat 접근기록에서 뽑아낸	Tomcat 접근기록	취약점 검증대상의 URL 들이 출력된다.			
							4	검증대상의 URL 로 변조된 HTTP 질의를 수행한다.	- http://[Test_Domain]/index.jsp?page=<Sc<Script>ript>alert("test")</Sc</Script>ript> 외 HTTP 질의 다수	입력데이터에 대한 HTTP 질의가 수행된다.			
							5	WebCastellum 변조된 HTTP 질의의 결과에서 스크립트 주입 공격 취약점 패턴을 확인한다.	HTTP 질의 결과	HTTP 질의 결과에 대해서 WebCastellum 가 검증을 수행한다.			
							6	WebCastellum Debug 로그를 통해서 변조된 HTTP 질의의 감지/대응여부를 확인한다.	일반	해당 취약점의 감지/대응결과가 Debug 로그에 나타난다.			
12	spoofing										N/A		
13	Fingerprinting & probing										N/A		
14	Brute force or denial of service attacks										N/A		
15	System File Disclosure										N/A		
16	Privilege Escalation										N/A		