WEB/WAS 보안 분야
Stack 통합 Test 결과보고서
[WebCastellum]

2013. 07.

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

목 차

I. Stack 통합 테스트 개요

공개SW Stack 통합테스트는 여러 공개SW들의 조합으로 시스템 Stack을 구성한 후 Stack을 구성하는 공개SW의 상호운용성에 중점을 두고 기능 및 성능테스트 시나리오를 개발하여 테스트를 진행한다.

본 통합테스트를 통해 안정된 Stack 정보를 제공하여 민간 및 공공 정보시스템 도입 시 활용될 수 있도록 한다.

1. 목적

□ 공개SW Stack 통합 테스트 수행 목적

- o 공개SW로 구성된 Stack이 유기적으로 잘 동작함을 확인
- o 다양한 Stack 구성에 기반을 둔 테스트를 통해 안정된 Stack 조합 규명
- o 공개SW 시스템 도입을 위한 Stack 참조모델의 신뢰성 정보로 활용
- o 공개SW의 신뢰성과 범용성에 대한 사용자 인식 제고

Ⅱ. 웹 방화벽

1. 방화벽의 역사

□ 1세대 방화벽 : 패킷 필터

○ 패킷 자체만을 보고 미리 설정된 정책에 따라 허용 또는 거부를 결정하는 초창기 방화벽은 1세대 방화벽이라고 한다. 방화벽 내부에서 상태(세션)를 관리하지 않는 기본 형태의 방화벽이다. 이 방화벽은 특정한 IP를 허용 또는 거부하거나 특정한 포트를 허용 또는 거부하는 용도로 사용된다.

□ 2세대 방화벽 : 스테이트풀 인스펙션

○ 패킷 필터 방화벽은 매우 효율적이긴 하지만 다음과 같은 몇 가지 문제가 있다.

- 모든 패킷이 모든 정책에 해당되는지 검사하므로 정책이 많아질수록 처리 속도가 느려진다.
- 돌아오는 패킷을 허용하는 정책으로 인해 보안이 취약해질 수 있다.
- FTP와 같이 파생 세션을 만드는 일부 프로토콜을 지원하기 위해 모든 포트를 다 열어야 될 수도 있다

○ 이러한 문제들을 해결하기 위해서 고안된 것이 패킷 단위의 검사가 아닌 세션 단위의 검사를 하는 스테이트풀 검사이다. 기본적인 스테이트풀 검사는 다양한 파생 세션을 모두 처리하지 못하는 경우가 있는데, FTP의 능동적/수동적 데이터 세션 등 복잡한 파생세션을 별도의 정책 추가 없이 모두 처리할 수 있는

확장된 스테이트풀 검사를 하는 방화벽도 있다.

□ 3세대 방화벽 : 애플리케이션 방화벽

○ 초창기에 네트워크를 기반으로 하던 공격 패턴이 점차 발달하여 일상적인 트래픽과 같은 특성을 가지면서 시스템을 공격하는 형태로 발전하게 되었다. 패킷 필터 기반의 방화벽으로는 이러한 공격을 방어하기 어려워지면서 패킷의 내용을 검사하고 더 나아가서는 애플리케이션에 어떠한 영향을 미칠지를 분석하는 방화벽이 출현하기 시작한다. IPS, WAF, UTM 등으로 불리는 네트워크 장비들이 애플리케이션 방화벽이라고 할 수 있다. 다른 발전 방향으로는 IP 주소나 MAC 주소 같이 사용자가 기억하거나 이해하기 어려운 대상을 이용해서 방화벽 정책을 수립하던 과거의 방화벽과는 달리 새로운 방화벽들은 사용자에게 보다 친숙한 사람의 이름이나 도메인 주소를 이용해서 정책을 수립할 수 있게 하는 방화벽들도 속속 등장하고 있다.

(출처 : 위키피디아)

2. 웹 애플리케이션 방화벽의 필요성

□ 웹 어플리케이션

○ 소프트웨어 공학적 관점에서 웹 애플리케이션 또는 웹앱은 인터넷이나 인트라넷을 통해 웹 브라우저에서 이용할 수 있는 응용 소프트웨어를 말한다.
웹 애플리케이션은 클라이언트로서 웹 브라우저를 사용하는 사람이 많기 때문에 인기를 누리고 있다 수천만 대의 PC에 굳이 소프트웨어를 배포해서 설치하지 않아도 웹 애플리케이션을 유지 관리할 수 있다는 점이 장점 중 하나이다. 현재에 들어서는 공공, 기업, 금융 등의 분야에서 활발히 사용 중이다.

□ 웹 어플리케이션 전용 보안 제품의 등장

○ 대부분의 기업, 금융, 공공 기관 등에서 운용중인 웹 서버에는 금융 거래, 신용카드 정보, 개인정보 등을 다루는 웹 애플리케이션들이 작동하고 있다. 그리고 통상 이러한 데이터는 웹 전용 루트 80, 8080, 443을 사용해 HTTP/ HTTPS의 소통이 이뤄진다. 하지만 방화벽, IDS, IPS, VPN 등의 기존 보안제품들은 열려진 80, 443 포트에 대한 필터링, 애플리케이션 레이어에 대한 필터링이 잘 이뤄지지 않는다는 것이 문제다.
또한 웹 개발자들은 보안보다는 개발 부분에 초점을 두기 때문에 프로그램 개발 시에 나타나는 웹 프로그래밍 오류들(웹 어플리케이션 취약점)이 존재하는 것이 다반사다.
이런 현 운용상황에 대한 구조적인 문제점을 극복하고 웹 취약성에 대한 해킹을 최소화하기 위해서 웹 애플리케이션을 위한 전용 보안제품이 탄생하게 됐다.

Ⅲ. 웹 어플리케이션 취약점

1. 행정안전부 취약점 고려사항

□ 2010년 국가정보원, KISA, OWASP에서 정의된 웹 어플리케이션 보안 시 고려해야 할 취약점은 아래와 같다.

점 검 항 목	국가정보원	KISA	OWASP
① 스크립트 삽입(XSS)	O	O	O
② 스크립트 요청 참조(CSRF)			O
③ 악성 파일 실행	O		O
④ SQL 구문 삽입	O	O	O
⑤ URL/파라메터 조작		O	O
⑥ 파일 업로드	O	O	O
⑦ 파일 다운로드	O	O	O
⑧ URL강제접속/인증우회		O	O
⑨ 서비스 메소드 설정	O		
⑩ 에러처리 및 기타 정보 노출		O	O
⑪ ID/PW 관리	O		
⑫ 환경설정 보안 고려사항	O	O	O

※ 위 항목은 홈페이지 개발단계에서 고려하여야 할 보안점검항목으로 보안진단 항목과는 다를 수 있음.

2. OWASP 취약점 고려사항

□ OWASP TOP10 2013 보안 취약점 변경사항

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

2010년과 2013년의 항목비교표로, 변경된 부분은 A7, A8, 그리고 2010년 버전에 존재했던 A6: Security Misconfiguration 이 삭제되고, A9: Insufficient Transport Layer Protection 이 2013년 버전의 A6 인 Sensitive Data Exposure 로 통합되었으며, 2013년에 새로 생성된 A9 Using Known Vulnerable Components 이 있음.

- 2013 중점 강화된 보안 항목 : 개인정보보안, 권한관리, CSRF 방지

3. 보안 분야 주요 공개SW

보안 분야 주요 공개SW 중 ESAPI_WAF, ModSecurity, WebCastellum을 테스트 SW로 선정하고 공개SW 선정지표를 통하여 점수가 두 번째로 높은 WebCastellum를 사용하여 테스트를 진행 한다.

[표 III-1. 분석 분야 주요 공개SW]

제품명	Stack 환경	홈페이지	비고
ESAPI_WAF	Linux/Window/Mac	https://www.owasp.org/index.php/ESAPI_Swingset	
ModSecurity	Linux/Window/Mac	http://www.modsecurity.org/	
WebCastellum	Linux/Window/Mac	http://www.webcastellum.org/	

[표 III-2. 분석 분야 주요 공개SW 선정지표 점수]

분야	세부분야	대상	항목[배점]				총점 [100]
			Document [25]	Support [25]	Product [30]	Community [20]	
보안	WEB/WAS Security	ESAPI_WAF	17.4	12.5	18.8	6.7	55.3
		ModSecurity	23.4	18.3	23.8	10.0	75.5
		WebCastellum	22.2	15.8	22.6	6.7	67.2

※ 공개SW 선정지표에 의해 선정된 공개SW가 품질/성능의 우수성을 뜻하는 것은 아님

※ [별첨1]공개SW WebCastellum 선정지표 테스트 결과

Ⅳ. 테스트 대상 소개

1. WebCastellum 소개

WebCastellum은 독일 WebCastellum사에 의해 2009년 7월 20일 WebCastellum 1.8.1버전을 시작으로 1.8.3버전까지 제작된 공개SW 웹 어플리케이션 방화벽 프로그램이다.(WAF - Web Application FireWall)

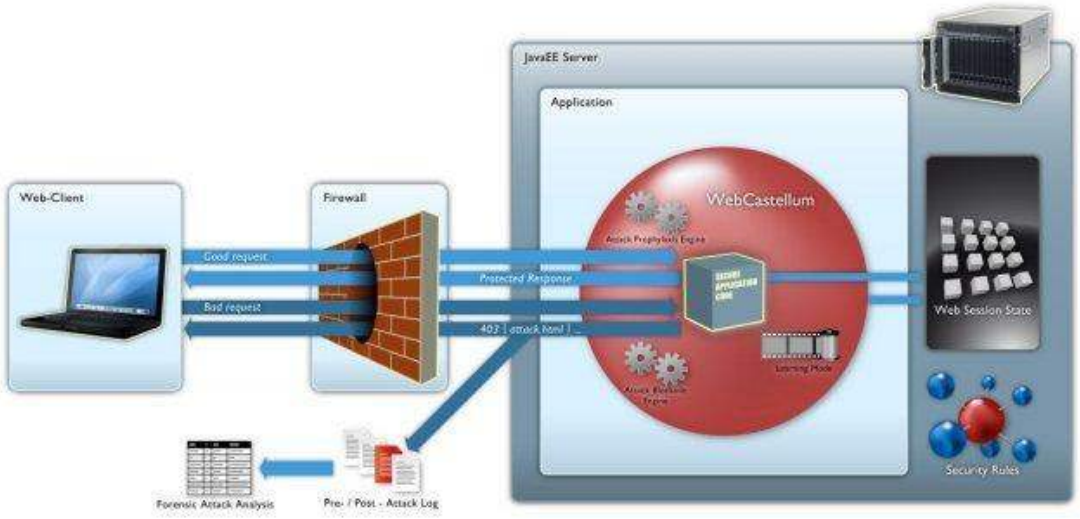
WebCastellum은 100% JAVA 언어로 개발 되었으며 경량 소스(1.9MB)로 설치가 매우 빠르고 쉬우며 독립적으로 구성 되어있다. Tomcat, JBoss ,WebLogic, WebSphere등 일반적인 JAVA EE 서버와 호환이 가능하다.

모든 HTTP 통신을 확인하고 응답과 요청을 모니터링 할 수 있고 정규 표현식을 통해 필터 규칙을 적용시킨다. 또한 개별적으로 핸들러를 설정하여 보안 적용을 각각 별도로 지정 할 수 있다.

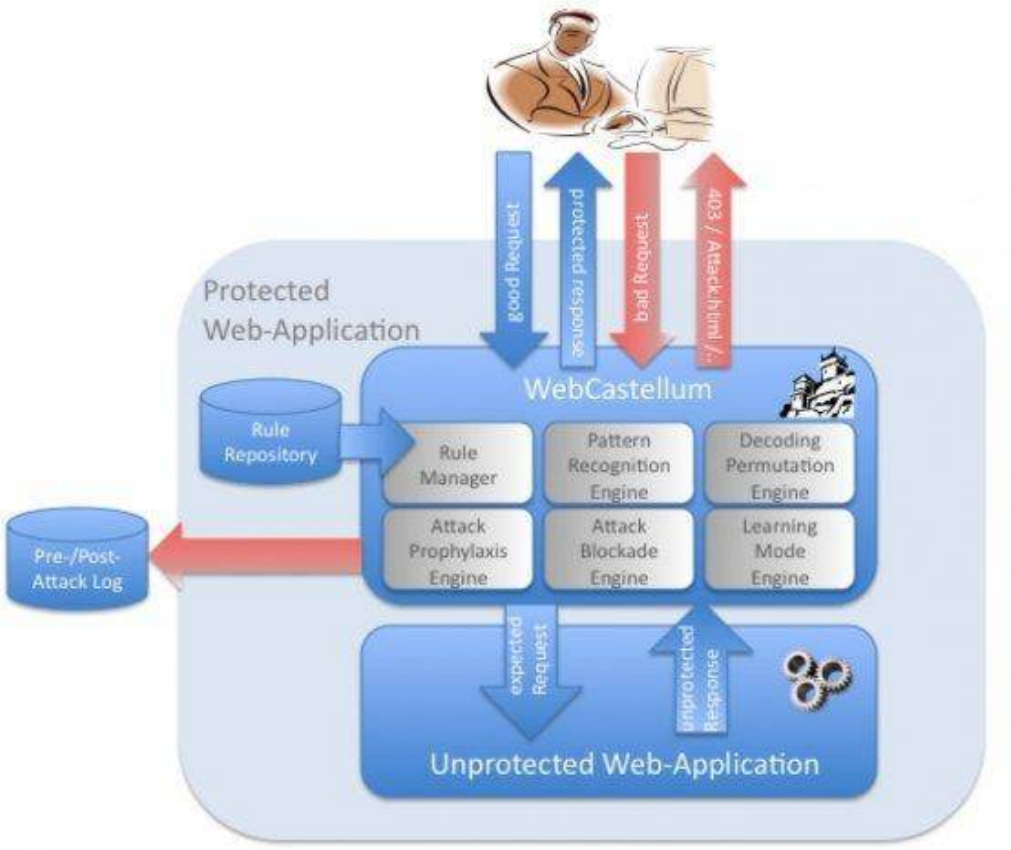
세션 및 쿠키를 관리하고 웹 어플리케이션에 입력되는 모든 값을 필터링하여 보안성 검증을 하도록 구성되어 있다.

개발자가 웹 어플리케이션 개발에만 집중 할 수 있게 해준다.

2. WebCastellum 구조 및 디자인 구성



[그림 IV-1. WebCastellum 오버뷰]



[그림 IV-2. WebCastellum 디자인 구조]

□ WebCastellum 웹 방화벽 기능

- o SQL Injection
- o Cross Site Scripting (XSS)
- o Cross-Site Request Forgery (CSRF)
- o Command Injection
- o CRLF Injection
- o Session fixation
- o Parameter manipulation
- o Privilege Escalation
- o Directory Traversal
- o System File Disclosure
- o Fingerprinting & probing
- o Brute force or denial of service attacks
- o etc

□ WebCastellum 지원 OS

JAVA 1.6이상 버전을 지원하는 Linux, Windows, Mac계열의 모든 OS를 지원한다.

※ 추가적인 자세한 정보는 아래의 링크 정보 참조

- http://www.webcastellum.org/

V. Stack 통합 테스트

1. 테스트 환경

□ WebCastellum 환경

[표 V-1. 테스트 SW]

SW	Version
Apache Tomcat	6.0
Test Web Application(JAVA)	1.6
WebCastellum	1.8.3

□ Stack 환경

[표 V-2. Stack 환경]

Stack	Clinet OS	IP	Server OS	IP
A	Ubuntu 12.04	121.162.249.88	CentOS 6.2	121.162.249.18

□ HW 환경

[표 V-3. HW 환경]

제조사	모델명	CPU	MEM	Disk	NIC
HP	DL360G6	Quad-Core 2.40 GHz-2P	8GB	750GB	Gigabit 1Port

2. 주요 테스트 방법

□ 시나리오 테스트

시나리오 테스트 기법은 단일 기능에 대한 결함 여부를 확인하는 것이 아니라, 서로 다른 컴포넌트 사이의 상호작용과 간섭으로 발생할 수 있는 결함을 발견하기 위한 기법이다.

본 테스트에서는 사용자 시나리오 테스트 기법을 적용하여 WebCastellum을 사용하여 보안 취약점 방어 시나리오를 도출하였다. 각각의 항목에서 도출한 세부 시나리오는 사용자가 일반적으로 수행할 수 있는 시나리오를 추출하여 테스트케이스로 작성하였다.

□ 상호 운용성 기반 테스트

상호 운용성은 서로 다른 기술로 이루어진 제품이나 서비스가 상호작용 상의 오류가 없는지 검증하는 기법으로, 본 테스트에서는 애플리케이션이 지원하는 Stack을 구성하여 애플리케이션과 Stack 환경 사이의 상호작용 상의 동작여부를 검증하였다.

3. 기능 테스트 수행 결과

기능 테스트 수행 관련 세부 시나리오는 별첨「WebCastellum 테스트 케이스」문서를 참고한다.

□ 테스트 시나리오 현황

[표 V-4. 테스트 시나리오 현황]

기능	테스트 시나리오	테스트 케이스
Security	15	65
모니터링	1	1
합 계	16	66

□ 테스트 결과

기능 테스트 시나리오를 통한 테스트 수행 결과 시나리오 상의 대부분의 기능이 예상 결과와 동일하게 동작함을 확인하였으나 Spoofing, Fingerprinting & probing, Brute force or denial of service attacks, System File Disclosure, Privilege Escalation 등 5개는 기능 구조상 확인이 불가능하여 N/A처리를 하였다.

[표 V-5. 테스트 결과]

분류		PASS	FAIL	N/A
기능	개수			
Security	15	10	0	5
모니터링	1	1	0	0

4. 성능 테스트 수행 결과

성능 테스트의 경우 하드웨어 사양뿐 아니라, OS 및 애플리케이션 환경 구성에 따라 측정 결과가 상이하므로, 실제 운영 시스템 환경에 따라 테스트 결과가 다를 수 있다.

본 성능 테스트는 Web Application에 jMeter를 사용하여 부하에 대한 시나리오를 재현하고, 테스트 대상 SW를 적용하였을 때와 적용하지 않을 때의 자원 사용률을 측정한다.

□ 테스트 시나리오

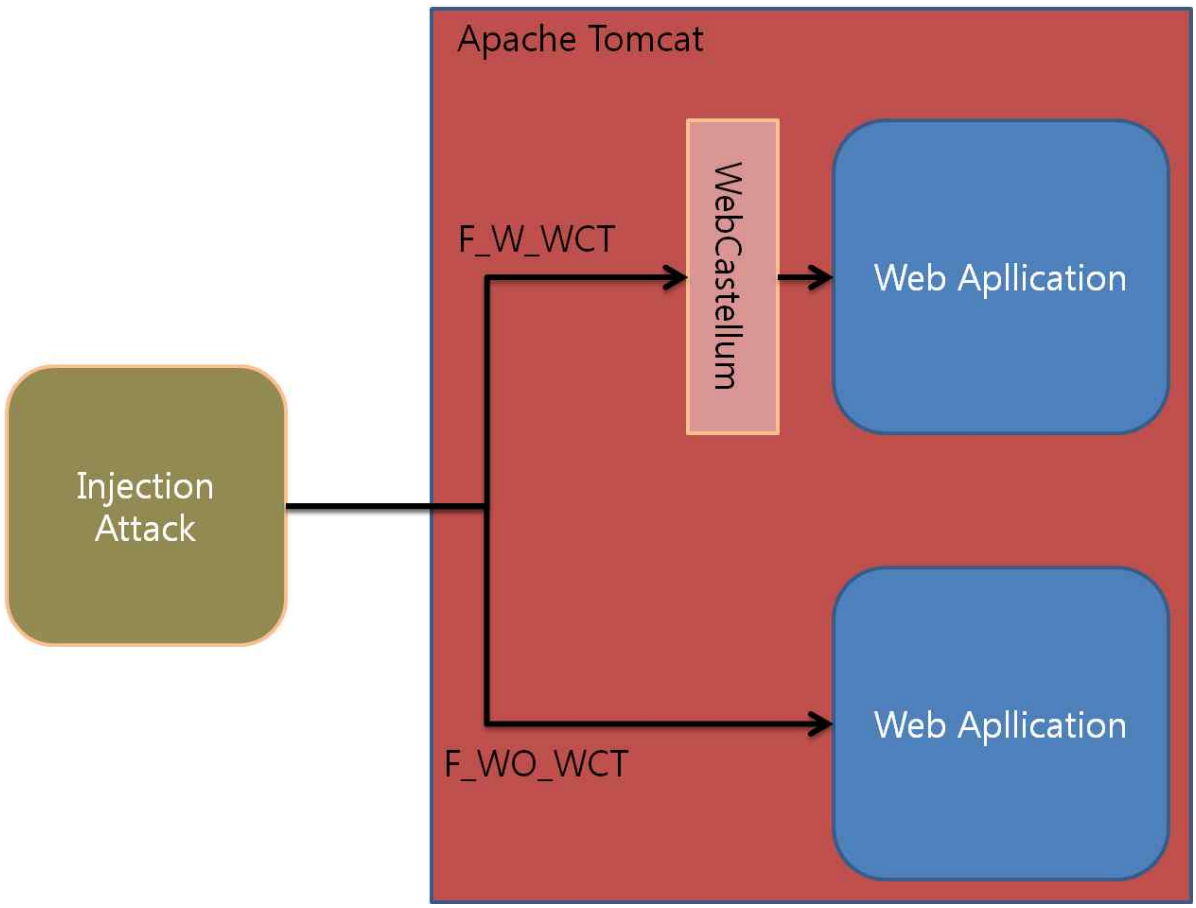
[표 V-6. 테스트 시나리오]

분류	시나리오ID	시나리오
F	F_W_WCT	테스트 대상 SW를 적용한 WAS에 부하를 주기위해 Injection을 사용한다.
	F_WO_WCT	테스트 대상 SW를 적용하지 않은 WAS에 부하를 주기위해 Injection을 사용한다

[표 V-7. 측정항목]

항목	내용
CPU 사용률	프로세스에서 CPU(Central Processing Unit)를 사용한 비율(%)
메모리 사용률	Physiscal 메모리 사용량

□ 성능 테스트 서버환경 정보



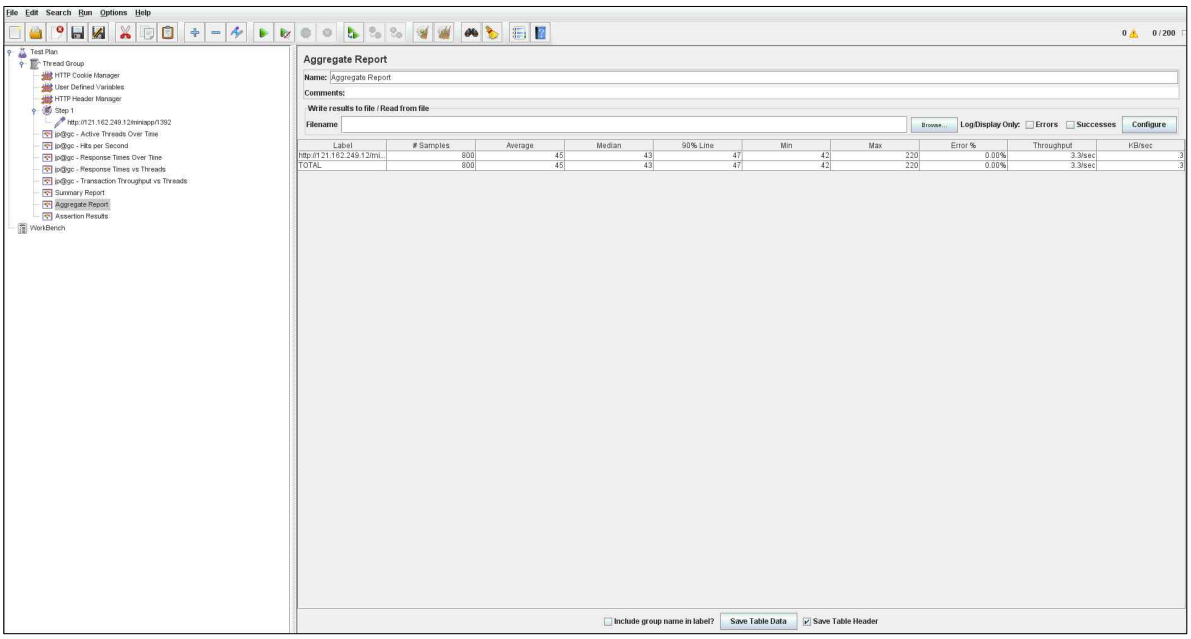
[그림 V-1. 성능 테스트 환경 정보]

□ 테스트 결과

o 분류 F_W_WCT 성능 테스트 결과

- 수행정보

수행 조건	- 테스트 대상 SW를 적용한 WAS에 부하를 주기위해 Injection을 사용한다. - 200명의 동시접속자가 5초안에 모두 접속하여 12Kb의 Post데이터를 4분동안 Injection하도록 스케쥴링을 한다.

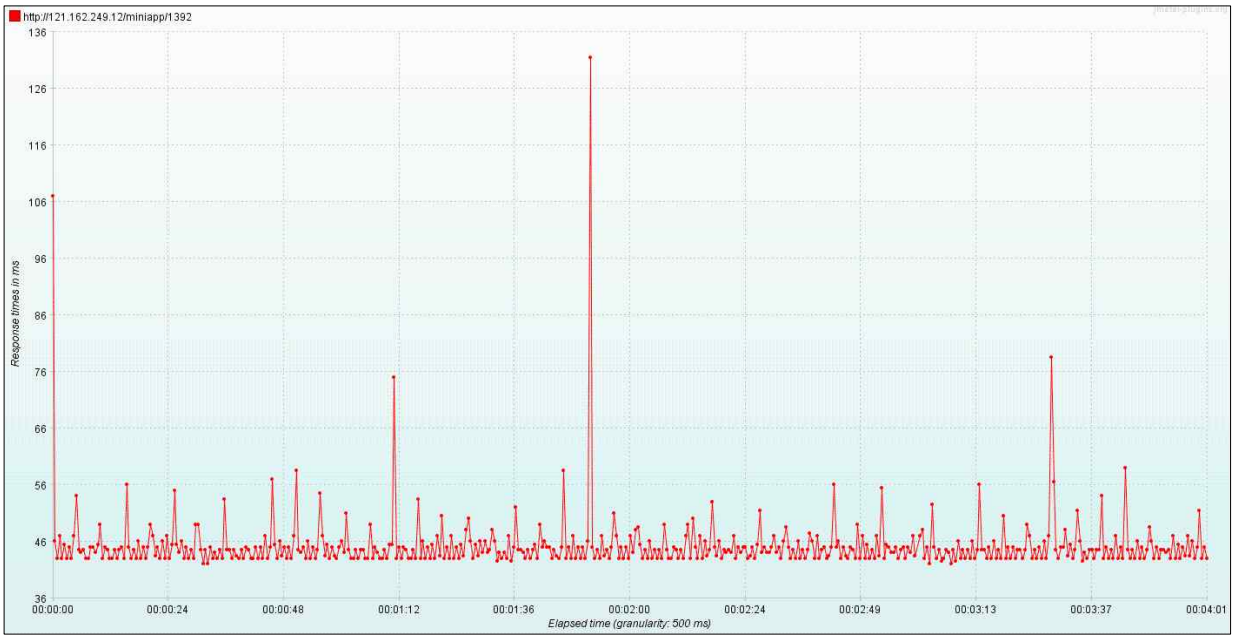


[그림 V-2. Aggregate Report]

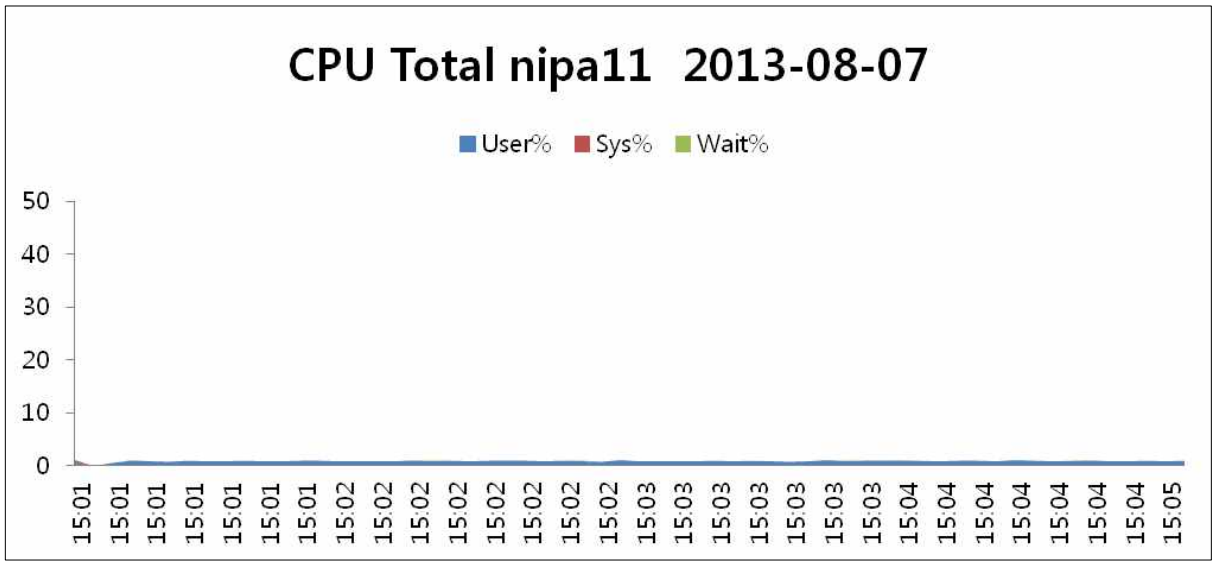
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
http://121.162.249.12/mi...	800	45	43	47	42	220	0.00%	3.3/sec	.3
TOTAL	800	45	43	47	42	220	0.00%	3.3/sec	.3

Sample	Average	Min	Max	KB/sec
800	45	42	220	.3

- 최소 42, 최대 220 , 평균 45

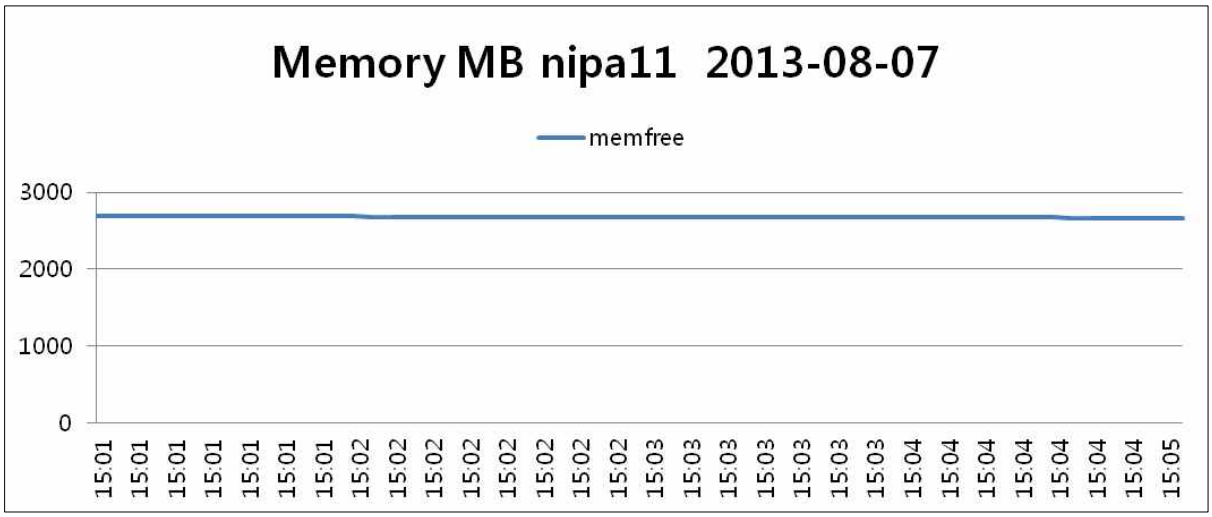


[그림 V-3. Response Time Over Time 그래프]



[그림 V-4. CPU 사용률]

- Cpu 사용률이 매우 낮음



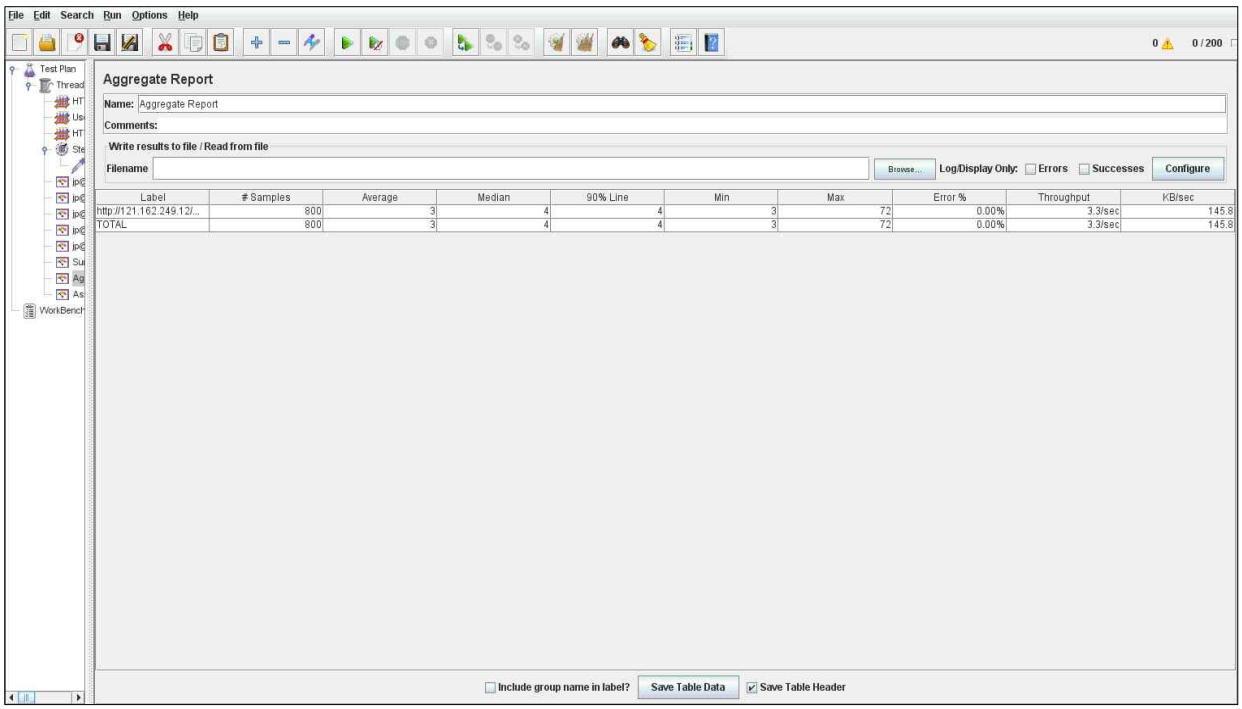
[그림 V-5. Memory 사용률]

- Memory 사용률이 매우 낮음

o 분류 F_WO_WCT 성능 테스트 결과

- 수행정보

수행 조건	- 테스트 대상 SW를 적용하지 않은 WAS에 부하를 주기위해 Injection을 사용한다. - 200명의 동시접속자가 5초안에 모두 접속하여 12Kb의 Post데이터를 4분동안 Injection하도록 스케쥴링을 한다.

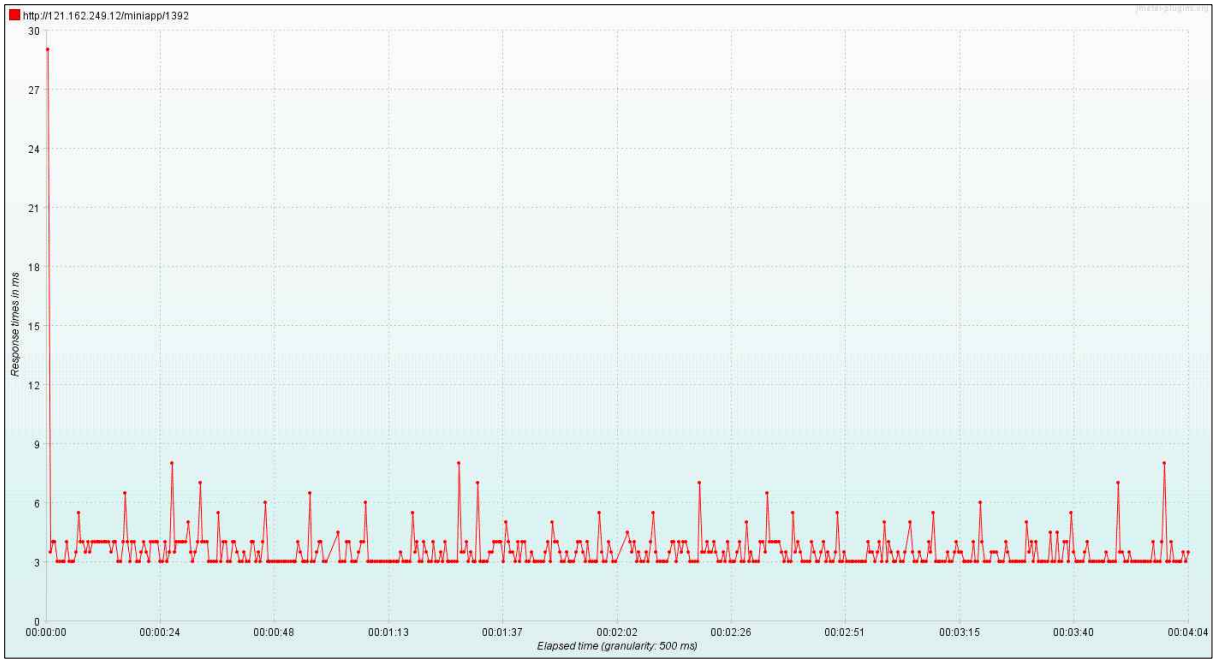


[그림 V-6. Aggregate Report]

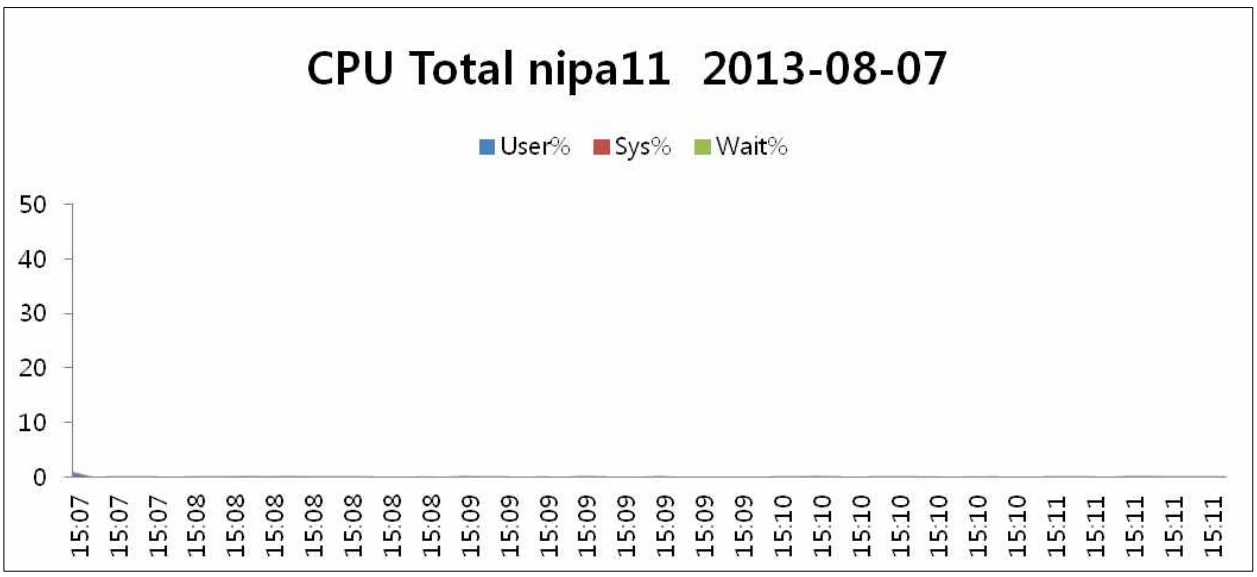
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
http://121.162.249.12/...	800	3	4	4	3	72	0.00%	3.3/sec	145.8
TOTAL	800	3	4	4	3	72	0.00%	3.3/sec	145.8

Sample	Average	Min	Max	KB/sec
800	3	3	72	145.8

- 최소 3, 최대 72, 평균 3

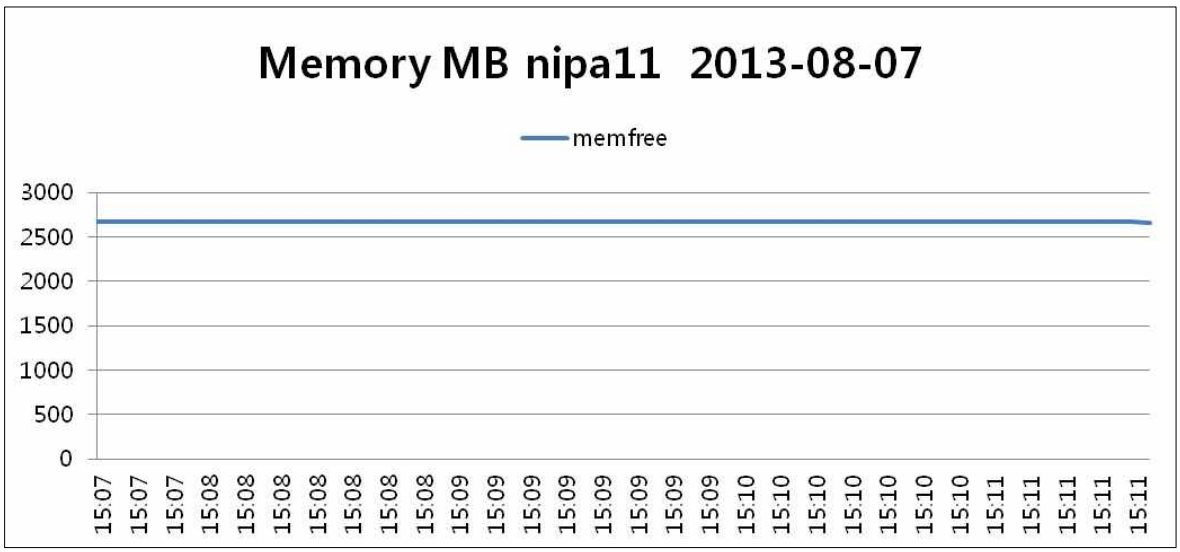


[그림 V-7. Response Time Over Time 그래프]



[그림 V-8. CPU 사용률]

- cpu 사용률이 매우 낮음



[그림 V-8. Memory 사용률]

- Memory 사용률이 매우 낮음

Ⅵ. 종합

□ WebCastellum 기능 테스트 수행 결과 공개SW로 구성된 Stack 상에서 각 기능 시나리오 수행 시 치명적 오류 또는 심각한 장애가 발생하지 않았으며, Stack을 구성하는 각 공개SW가 유기적으로 동작함을 확인하였다.

□ 성능 테스트 수행 결과 WebCastellum이 적용된 시나리오가 적용되지 않은 시나리오보다 응답 속도의 지연을 보였으며, CPU 및 Memory 자원 사용률은 차이가 없다는 것이 확인되었다.

※ 참고 자료

[1] http://www.webcastellum.org/

[2] https://www.owasp.org/index.php/Korea

[3] WebCastellumGuideEnglish.pdf