HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Mod security modul

CCERT-PUBDOC-2004-02-63

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

`Mod_security` je Apache modul koji djeluje kao sustav za detekciju i prevenciju neovlaštenih aktivnosti usmjerenih prema Web poslužiteljima i pripadajućim aplikacijama. Javno dostupne Web aplikacije mogu predstavljati iznimno visoki sigurnosni rizik, a dosadašnja iskustva pokazuju da je dovoljan samo jedan sigurnosni propust u izvornom kodu aplikacije koji neovlaštenim korisnicima može omogućiti pristup podacima ili u gorem slučaju preuzimanje potpune kontrole nad sustavom.
Osnovne karakteristike koje `mod_security` nudi korisnicima su sljedeće:

- Presretanje i pregledavanje svih HTTP/HTTPS zahtjeva;
- Mogućnost detekcije naprednih metoda zaobilaženja IDS sustava (engl. *Anti-evasion*);
- Mogućnost detekcije napada koji se baziraju na kodiranju URL adresa;
- Mogućnost kreiranja vlastitih pravila, pri čemu je moguće korištenje regularnih izraza;
- Izvršavanje posebno definiranih akcija nakon detekcije napada;
- Detaljno, korisnički podesivo bilježenje svih zahtjeva.

U nastavku dokumenta opisani su osnovni sigurnosni problemi koji se javljaju kod Web aplikacija, postupci instalacije i konfiguracije mod_security modula, kao i mogućnosti njegove primjene u praksi. Kako bi se ispitale karakteristike modula, dodatno je proveden skup specijalno prilagođenih testova kojima su ispitane osnovne funkcionalnosti programa. Rezultati provedenog testiranja, zajedno s kratkim opisom pojedinih testova, dani su u posljednjem poglavlju dokumenta (Poglavlje 8).

2. Sigurnost Web aplikacija

Prilikom zaštite računalnih mreža i informacijskih sustava posebnu je pažnju potrebno posvetiti upravo Web aplikacijama. Pod pojmom Web aplikacija podrazumijeva se sva programska podrška namijenjena i prilagođena izvođenju na Web poslužiteljima, pri čemu se komunikacija s korisnikom, odnosno klijentom, odvija putem HTTP/HTTPS protokola. Zaštiti Web aplikacija do sada se i nije davao preveliki značaj, međutim, kako se sve više napada odvija upravo putem HTTP/HTTPS protokola, Web aplikacije se ne smije zanemarivati. Provedena istraživanja pokazuju da je oko 92% Web aplikacija ranjivo na neki od poznatih napada (*SQL Injection*, XSS, provjera ulaza i sl.), što jasno ukazuje na ozbiljnost problema.

Web aplikacije izvor su brojnih problema iz nekoliko razloga. Iako neki od problema proizlaze iz samih karakteristika i nedostataka HTTP protokola, većina problema javlja se kao posljedica površnog i brzopletog razvoja samog programskog koda. Dodatni problem predstavlja činjenica da se HTTP/HTTPS promet redovito mora propustiti na mrežnoj opremi (vatrozidi, mrežni usmjerivači i sl.), budući da je Web servis sam po sebi predviđen upravo za javnu objavu informacija na Internetu.

Još jedna od karakteristika HTTP protokola koja na Web programere stavlja dodatnu odgovornost je beskonekcijska (engl. *stateless*) priroda protokola. Budući da HTTP protokol ne sadrži ugrađene metode za inicijalizaciju, uspostavljanje i raskidanje sjednica, upravljanje sjednica potrebno je rješavati na nivou aplikacije. Problem upravljanja sjednicama jedan je od velikih izazova za Web programere, pogotovo kada se uzme u obzir da je poznat velik broj napada koji su bazirani upravo na loše implementiranom *session managementu*.

Također, HTTP protokol je vrlo jednostavan te ga je zbog toga vrlo lako emulirati i kreirati zahtjeve po vlastitom izboru, što neovlašteni korisnici često koriste prilikom provođenja malicioznih aktivnosti. Web aplikacije postaju sve složenije i time podložnije napadima, pogotovo kada se uzme u obzir da HTTP protokol nije prvenstveno bio zamišljen za mnoga područja primjene u kojima se danas koristi. Funkcije koje Web aplikacije danas obavljaju kreću se tako od jednostavnih pretraživanja lokalnih baza podataka do različitih vrsta Web portala, foruma i *Webmail* okruženja, pa sve do kompleksnih sustava za elektroničko poslovanje i udaljeno upravljanje sustavima. Zbog navedenih funkcionalnosti koje su dostupne putem Web servisa, HTTP/HTTPS protokolom se prenose razne povjerljive informacije (brojevi kreditnih kartica, poslovni podaci, itd.).

Iz svega upravo navedenog, može se zaključiti da Web aplikacije, upravo zbog specifičnosti HTTP/HTTPS protokola na kojem se baziraju, zaista posjeduju određene sigurnosne probleme kojima je potrebno posvetiti dovoljno pažnje kako ne bi došlo do ugrožavanja cjelokupnog integriteta informacijskog sustava u sklopu kojeg se i same aplikacije pokreću.

Jedno od mogućih rješenja za unaprjeđenje sigurnosti Web aplikacija je upotreba vatrozida, odnosno njihovih funkcionalnosti koje omogućuju analizu i filtriranje prometa prema sadržaju paketa (engl. *content filtering*). Iako većina komercijalnih vatrozida danas nudi određene funkcionalnosti koje omogućuju filtriranje prometa prema sadržaju paketa, ove funkcije najčešće su vrlo jednostavne te nude nisku razinu zaštite. Za korištenje naprednijih metoda filtriranja redovito je potrebno ugraditi zasebne module, specijalno prilagođene za ovu namjenu, koji najčešće zahtijevaju postavljanje zasebnog poslužitelja na koji se preusmjeravaju zahtjevi koje je potrebno analizirati.

Osim upravo spomenutih vatrozida, kojima analiza sadržaja paketa redovito nije primarna namjena, danas postoji i velik broj specijaliziranih alata namijenjenih isključivo analizi i filtriranju sadržaja paketa. Nažalost, ovakva rješenje često su prilično skupa, a samim time i nepristupačna, pogotovo manjim i srednjim tvrtkama.

Još jedna od tehnika koja se vrlo često koristi za zaštitu Web poslužitelja od neovlaštenih aktivnosti je tehnologija reverznih poslužitelja (engl. *reverse proxy)*, gdje se sva komunikacija između klijenta i poslužitelja provodi putem *proxy* poslužitelja na kojem je moguće implementirati različite sigurnosne kontrole.

Osnovna prednost ovakvog pristupa je centralizirana kontrola mrežnog prometa, nad kojim je tada lakše provoditi sigurnosne provjere. Osim toga, *reverse proxy* poslužitelji najčešće su potpuno transparentni za krajnjeg korisnika, a osim centralizirane kontrole nude i brojne druge prednosti kao što su preusmjeravanje prometa prema opterećenju (engl. *load balancing*), prikrivanje topologije i izolacija privatne mreže i sl. Nedostatak *reverse proxy* poslužitelja je povećana kompleksnost cijelog

sustava, a njegovom centralizacijom dobiva se jedinstvena točka čijim ugrožavanjem dolazi do potencijalnog ispada cijelog sustava (npr. prilikom provođenja *DoS* napada).
Većina postojećih sigurnosnih alata ne koriste specifičnosti HTTP protokola u svojim operacijama. Upravo zato dolazi do potrebe novih alata koji će koristiti HTTP protokol i razumjeti ga vrlo dobro. Jedno takvo rješenje je dodatni modul za Apache Web poslužitelje *mod_security*. Mod_security programski paket Apache Web poslužiteljima dodaje mogućnosti detekcije neovlaštenih aktivnosti, što pridonosi sigurnosti Apache Web poslužitelja i aplikacija koje se na njemu pokreću.
Osim detekcije malicioznih aktivnosti, *mod_security* pruža i prevenciju, odnosno zaštitu od napada, što je također jedna od njegovih kvaliteta. Budući da se način rada *mod_security* modula bazira na presretanju komunikacije između klijenta i poslužitelja, svi zahtjevi za koje se smatra da sadrže zlonamjerni sadržaj odbacuju se prije nego što ih sam poslužitelj primi. Ovisno o definiranim pravilima, program pri detekciji malicioznih paketa može obaviti i niz unaprijed definiranih akcija kojima se može spriječiti daljnje provođenje malicioznih aktivnosti.
Poput svakog drugog Apache modula, *mod_security* se može lako uključiti u Apache instalaciju. Nešto teži dio je njegova konfiguracija, budući da je, za maksimalno iskorištavanje njegovih mogućnosti, potrebno ne samo poznavati način rada i konfiguracijske parametre modula, već i arhitekturu sustava koji se štiti, kako bi se pravila filtriranja paketa mogla prilagoditi specifičnostima aplikacije. Sigurnosna razina koju *mod_security* nudi prvenstveno ovisi o kvaliteti definiranih sigurnosnih pravila. Ukoliko su pravila pogrešno ili površno implementirana, postoji mogućnost da neki od napada prođu nezabilježeno. Međutim, incidenti do kojih može doći uslijed napada na sustav daleko su skuplji od utrošenog vremena na efektivnu konfiguraciju i dodatno procesiranje zahtjeva.

3. Instalacija

Način instalacije `mod_security` modula razlikuje se ovisno o inačici Apache Web poslužitelja, kao i o platformi na kojoj se Web poslužitelj koristi. U ovom poglavlju bit će opisan postupak instalacije na Windows i Linux operacijskim sustavima.
Važno je napomenuti da `mod_security` modul radi sa obje verzije Apache Web poslužitelja (1.3.x i 2.x), te da ne zahtjeva određene inačice poslužitelja za ispravan rad.
`Mod_security` modul za Apache poslužitelj moguće je dohvatiti sa URL adrese http://www.modsecurity.org/download/ u dva oblika; kao izvorni kod (tar.gz paket) ili kao izvršnu datoteku.

3.1. Instalacija iz izvornog koda

Prilikom instalacije modula iz izvornog koda moguća su dva scenarija. Prvu opciju predstavlja integriranje modula u sam Web poslužitelj (statička instalacija), čime se postiže nešto brže izvršavanje, no i postupak instalacije je nešto složeniji, budući da zahtjeva ponovno prevođenje Apache Web poslužitelja.
Drugi, mnogo jednostavniji način, je prevesti *mod_security* modul kao dinamički dijeljeni objekt (*engl. Dynamic Shared Object, DSO*), te ga kao takvog uključiti putem konfiguracijske datoteke Apache poslužitelja (direktive LoadModule i AddModule). Ovakav pristup omogućava znatno jednostavniju instalaciju i integraciju modula u postojeći sustav te se kao takav može preporučiti svim administratorima, pogotovo onim manje iskusnima.
Postupak instalacije `mod_security` programa kao DSO modula opisan je u nastavku poglavlja. Postupak se bazira na korištenju apxs programa koji dolazi sa Apache poslužiteljem i koji je predviđen isključivo za ovu namjenu:
Instalacija modula u Web poslužitelj pomoću apxs alata:

```
/apache/bin/apxs -cia mod_security.c
```

Ponovno pokretanje Web poslužitelja:

```
/apache/bin/apachectl restart
```

U slučaju statičke instalacije `mod_security` modula postupak instalacije razlikuje se ovisno o inačici Apache Web poslužitelja.
Za Apache poslužitelj inačice 1.x, potrebno je izvršiti ponovnu instalaciju Apache poslužitelja uz neke dodatne opcije:

1. Kopiranje `mod_security.c` datoteke u direktorij `/src/modules/extra`
2. Konfiguracija Apache poslužitelja uz dvije dodatne opcije

```
--activate-module=src/modules/extra/mod_security
--enable-module=security
```

3. Nastavak prevođenja i instalacije Apache poslužitelja na uobičajen način

Kako trenutno ne postoji dostupna dokumentacija za integriranje modula u Apache Web poslužitelj 2.x, ukoliko se na računalu nalazi upravo ta verzija, proces instalacije je moguće izvesti na sljedeći način:

1. Konfiguracija poslužitelja sa naredbom da uključi *mod_security* modul:

```
./configure --prefix=(prefiks) --with-module=mappers:
security
```

2. Kopiranje `mod_security.c` u direktorij `modules/mappers`
3. Dodavanje sljedećih linija u datoteku `modules/mappers/modules.mk`:

```
mod_security.la: mod_security.lo
   $ (MOD_LINK) mod_security.lo
   …
static = mod_negotiation.la mod_security.la mod_dir.la …
```

4. Nastavak prevođenja i instalacije Apache poslužitelja na uobičajen način

3.2. Instalacija pomoću izvršne datoteke

Prilikom instalacije modula kada je izvršna verzija već dostupna, te nije potrebno prethodno prevođenje izvornog koda, instalacija se za obje inačice Apache Web poslužitelja odvija na sljedeći način:

1. Kopiranje `mod_security.so` datoteke (Linux računala), odnosno `mod_security.dll` datoteke (Windows računala) u direktorij `libexec/`
2. Dodavanje nove opcije u `httpd.conf` datoteku

```
LoadModule security_module libexec/mod_security.so
```

4. Mehanizmi zaštite

Konfiguracija mod_security modula u potpunosti se odvija preko httpd.conf datoteke. Da bi Apache Web poslužitelj koristio mod_security modul potrebno ga je u konfiguracijskoj datoteci uključiti pomoću <IfModule> direktive:

```
IfModule mod_security.c>
    #popis pravila
    …
/IfModule>
```

Nakon što je u httpd.conf datoteci uključena upravo navedena direktiva potrebno je popuniti tijelo direktive popisom mod_security sigurnosnih pravila. U nastavku poglavlja dane su gotovo sve direktive koje mod_security podržava. Potrebno je unijeti samo one direktive koje se žele koristiti, dok za one koje se ne koriste postoje inicijalne, unaprijed pretpostavljene, vrijednosti.

4.1. Konfiguracijske direktive

4.1.1. Uključivanje filtriranja

Prema inicijalnoj konfiguraciji mod_security programa filtriranje prometa je isključeno. Kako bi se ono omogućilo potrebno je navesti sljedeću direktivu:

```
SecFilterEngine On
```

Moguće vrijednosti SecFilterEngine parametra su:

On – analiza svih dolazećih zahtjeva

Off – ne analizira se niti jedan zahtjev

DynamicOnly – analiza samo onih zahtjeva koji su generirani dinamički za vrijeme izvođenja. Korištenje ove opcije štedi procesorsko vrijeme za provjeru pristupa statičkim datotekama.

4.1.2. Provjera POST sadržaja

Sadržaji koji se poslužitelju prosljeđuju POST metodom ne analiziraju se sve dok se ne uključi odgovarajuća direktiva. Budući da se POST metodom najčešće prosljeđuju vrijednosti koje korisnici unose putem Web formi, ovu opciju svakako se preporučuje uključiti. Osnovni razlog je taj što se većina napada provodi upravo na način da se poslužitelju putem Web formi proslijedi specijalno prilagođeni maliciozni niz, koji će iskoristiti ranjivosti unutar aplikacije.

```
SecFilterScanPOST On
```

4.1.3. Nasljeđivanje filtra

Poddirektoriji Apache poslužitelja nasljeđuju filtre koji su definirani u direktorijima koji su u lancu iznad njih, tzv. direktorija roditelja. Ovakvo ponašanje je prihvatljivo i potrebno u većini slučajeva, međutim postoje određene iznimke u kojima ovakvo ponašanje nije prihvatljivo. Ponekad je u određenim dijelovima strukture Web poslužitelja potrebno definirati drukčiju sigurnosnu politiku kojom će se zadovoljiti potrebe sustava. Kako bi se onemogućilo nasljeđivanje sigurnosnih postavki iz nadređenog direktorija potrebno je koristiti sljedeću direktivu:

```
<Location /web/direktorij>
    SecFilterInheritance Off
</Location>
```

U navedenom primjeru koristi se i direktiva <Location> za definiranje direktorija u kojem je potrebno obaviti navedene akcije, u ovom slučaju isključiti nasljeđivanje filtra.

4.1.4. Provjera URL kodiranja

Kako bi se omogućilo korištenje specijalnih znakova unutar URL adrese, HTTP specifikacija dozvoljava da se znakovi unutar URL adrese prikazuju u heksadecimalnom obliku. U tom slučaju znakove je potrebno prikazati kao kombinaciju tri znaka %XY, gdje je XY kod znaka u heksadecimalnom obliku. Heksadecimalni znakovi dozvoljavaju slova od A do F, međutim, napadači vrlo često koriste i druge

znakove, s ciljem zaobilaženja definiranih sigurnosnih mjera te sustava za detekciju neovlaštenih aktivnosti. `Mod_security` program provjerava sve podržane tipove kodiranja kako bi se osiguralo da su svi upiti ispravno kodirani. Provjera valjanosti kodiranja URL niza uključuje se direktivom:

```
SecFilterCheckURLEncoding On
```

Bitno je napomenuti da navedena direktiva neće provjeravati POST dijelove upita ukoliko se koristi "*multipart/form-data*" kodiranje.

4.1.5. Provjera *Unicode* kodiranja

Ova vrsta provjere prvi se put pojavila u 1.6 inačici `mod_security` modula. Ukoliko se direktiva za provjeru *unicode* kodiranja ne uključi, provjera ove vrste kodiranja neće biti obavljena. Ovu direktivu je potrebno uključiti ukoliko aplikacija ili operacijski sustav na kojem se aplikacija odvija razumije *unicode* skup znakova:

```
SecFilterCheckUnicodeEncoding On
```

Navedena postavka pretpostavlja da se radi o UTF-8 kodiranju, te će *mod_security* tražiti sljedeće vrste pogrešaka:

- Nedovoljan broj okteta. UTF-8 podržava kodiranje pomoću dva, tri, četiri, pet i šest okteta. Ukoliko se otkrije da nedostaje jedan ili više okteta, *mod_security* će upozoriti na taj nedostatak.
- Neispravno kodiranje. Dva najznačajnija bita u većini znakova moraju biti postavljeni na 0x80. Napadači često koriste ovu činjenicu kako bi zavarali *unicode* dekodere.
- Predugi znakovi. ASCII znakovi se u *unicode* kodiranju predstavljaju sa jednim oktetom. Međutim, većinu ASCII znakova je moguće kodirati i sa dva, tri, četiri, pet i šest okteta, pri čemu se može prisiliti dekoder na pretpostavku da se radi o znaku koji to zapravo i nije.

4.1.6. Provjera raspona okteta

Moguće je unaprijed zadati duljinu okteta zahtjeva koje poslužitelj prihvaća. Ovakva ograničenja mogu biti vrlo praktična za zaštitu od napada prepisivanjem spremnika (engl. *buffer overflow*). Ograničavanjem upita na određenu veličinu koja odgovara potrebama aplikacije u većini će je slučajeva zaštititi od napada ovog tipa. Sljedeća direktiva dozvoljava samo one zahtjeve u rasponu od 32 do 128 okteta:

```
SecFilterForceByteRange 32 128
```

Ukoliko se ova direktiva ne koristi, raspon je od 0 do 255 okteta, odnosno dozvoljeni su zahtjevi svih veličina.

Bitno je napomenuti da ova direktiva neće provjeriti POST sadržaj ako se koristi "*multipart/form-data*" kodiranje.

4.1.7. Vidljivost modula

U normalnim okolnostima napadač ne može znati da li Web poslužitelj koristi *mod_security* modul ili ne. U inicijalnim postavkama *mod_security* ne ispisuje nikakve informacije o svojem postojanju, već dozvoljava Apache poslužitelju da korisniku vrati standardne poruke. Ukoliko se korisniku želi dati do znanja da se u okviru Apache Web poslužitelja koristi *mod_security* modul, potrebno je zadati sljedeću direktivu:

```
SecServerResponseToken On
```

Na administratoru poslužitelja je da odluči da li će dozvoliti vidljivost modula ili će ga ostaviti prikrivenog. Postoje dva načina razmišljanja u ovom slučaju. Po prvom, dobro je iznijeti potencijalnim napadačima činjenicu da se na poslužitelju nalazi instaliran `mod_security` program ili bilo koja druga slična kontrola. U tom slučaju, napadač će pretpostaviti da se poslužitelj nadzire te će vjerojatno potražiti neku jednostavniju metu. Drugi način razmišljanja je taj da će napadač vidjeti zaštićeni poslužitelj kao veći izazov te će pokušati zaobići sigurnosne kontrole kako bi ostvario pristup sustavu.

Ukoliko se administrator odluči na uključivanje ove direktive, rezultat izravnog spajanja telnet servisom biti će sličan ovakvom ispisu:

```
[root@linux conf]# telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 23 Mar 2004 14:35:13 GMT
Server: Apache/1.3.29 (Unix) mod_security/1.7.5
Last-Modified: Tue, 23 Mar 2004 14:32:41 GMT
ETag: "2411f-a2-40604a89"
Accept-Ranges: bytes
Content-Length: 162
Connection: close
Content-Type: text/html
<html>
<head>
<title>Testing Document</title>
<meta       http-equiv="Content-Type"       content="text/html;
charset=iso-8859-1">
</head>
<body>
Test...
</body>
</html>

Connection closed by foreign host.
```

Ukoliko je Apache poslužitelj pomoću `ServerTokens` i `ServerSignature` direktiva konfiguriran tako da ne ispisuje nikakve informacije o sebi, nije bitno da li je `SecServerResponseToken` direktiva uključena ili ne.

4.1.8. Otkrivanje grešaka

Prilikom otkrivanja grešaka (*engl. Debugging*) koriste se dvije direktive, jedna za određivanje datoteke u koju će biti zapisan izlaz otkrivanja grešaka,

```
SecFilterDebugLog logovi/mod_security_debug_log
```

te direktiva koja određuje koliko će detaljan taj ispis biti:

```
SecFilterDebugLevel 0
```

Nivoi koji određuju koliko je ispis detaljan se kreću u rasponu od 0 do 3, te označavaju što će biti zabilježeno u datoteku na sljedeći način:

- 0 – ne bilježi se ništa,
- 1 – bilježe se samo značajni događaji,
- 2 – bilježe se sve poruke,
- 3 – bilježe se sve poruke sa dodatnim informacijama.

4.2. Filtriranje zahtjeva

Svaki nadolazeći zahtjev `mod_security` će presresti i analizirati prije nego što se proslijedi poslužitelju na izvršavanje. U prvom koraku provodi se niz ugrađenih provjera kako bi se utvrdila valjanost samog zahtjeva. Ove provjere moguće je kontrolirati konfiguracijskim direktivama programa kao što je to ranije opisano. Drugi dio analize sastoji se od niza sigurnosnih pravila definiranih od strane administratora sustava. Primljeni zahtjevi uspoređuju se sa nizom definiranih pravila, a u slučaju pozitivne detekcije provode se korisnički definirane akcija kojima je moguće kontrolirati način procesiranja primljenog paketa.

Najjednostavniji način filtriranja paketa je sljedeći:

```
SecFilter KLJUCNA_RIJEC
```

KLJUCNA_RIJEC pritom predstavlja znakovni niz koji *mod_security* traži u svakom upitu koji se prosljeđuje poslužitelju na izvršavanje. Pretraživanje se provodi samo nad prvom linijom HTTP upita (ona u kojoj se nalazi sam upit; npr. `GET index.html HTTP/1.0`), a u slučaju POST upita pretražuje se i sadržaj upita pod uvjetom da je omogućena `SecFilterScanPOST` direktiva.
Osim mogućnosti pretraživanja HTTP upita prema definiranim ključnim riječima, moguće je unutar istog pravila definirati i niz akcija koje će biti poduzete ukoliko upit odgovara definiranom pravilu. Sintaksa je sljedeća:

```
SecFilter KLJUČNA_RIJEČ [NIZ_AKCIJA]
```

Više informacija o akcijama koje je moguće provoditi nad upitima dano je u nastavku dokumenta.
Treba napomenuti da se definirana sigurnosna pravila (filtri) ne primjenjuju izravno na upite u onom obliku kakvom su primljeni od strane klijenta. Prije primjene definiranih sigurnosnih pravila, *mod_security* provodi normalizaciju upita, kako bi se onemogućilo zaobilaženje sigurnosnih mjera korištenjem različitih tehnika poznatih kod provođenja malicioznih aktivnosti usmjerenih prema Web poslužiteljima (kodiranje upita, različite varijacije korištenja *slash* i *backslash* znakova i sl.). Nakon provedene normalizacije provodi se filtriranje upita prema definiranoj sigurnosnoj politici.
U posljednjoj stabilnoj verziji `mod_security` modula prilikom normalizacije upita provode se sljedeće pretvorbe:

- pretvaranje \ u / (samo na Windows operacijskim sustavima);
- reduciranje /./ u /;
- provjera URL kodiranja (samo ukoliko je određena direktiva uključena);
- propuštanje zahtjeva određene duljine (samo ukoliko je određena direktiva uključena);
- dekodiranje URL načina kodiranja;
- reduciranje // u /.

Opisane transformacije spriječiti će određene vrste napada koje neovlašteni korisnici tipično provode, a ujedno će se i korisnicima `mod_security` modula olakšati njegova konfiguracija, budući da ne trebaju voditi računa o ovakvim problemima.
Primjer u kojem normalizacija upita može pomoći u detekciji nelegitimnih HTTP upita je implementacija filtara koji će zabraniti izvođenje naredbi ljuske. U konfiguracijskoj datoteci programa dovoljno je definirati pravilo koje će tražiti pojavljivanje `/bin/sh` niza u primljenom zahtjevu. Ukoliko napadač pokuša zaobići definirane sigurnosne mjere korištenjem kombinacije *slash* znakova i točke (npr. `/bin/./sh`), upit će biti uspješno blokiran s obzirom na prethodno provedenu normalizaciju upita.
Za aplikacije pisane u C/C++ programskim jezicima poznati su propusti koji ih mogu činiti osjetljivima na tzv. "*null byte*" napade. Radi se o napadima koji ranjivu aplikaciju pokušavaju zavarati ranijim prekidanjem znakovnog niza korištenjem NULL znakova. Ovakav tip napada moguće je spriječiti korištenjem `SecFilterByteRange` direktive kojom je moguće ograničiti duljinu dozvoljenih zahtjeva. Rješenje je ponuđeno od 1.7 inačice `mod_security` programa na taj način da se NULL okteti pretvaraju u praznine.
Ukoliko se koristi inačica ranija od 1.7, te se ne koristi direktiva `SecFilterByteRange`, aplikacija koja koristi pravilo u obliku:

```
SecFilter skrivena_informacija
```

bit će ranjiva na napad zahtjevom:

```
GET /dir/varijabla?x=vidljivo%00skrivena_informacija
```

jer prethodno navedeno pravilo neće detektirati željenu riječ u ovako formuliranom zahtjevu.
Kao filtar, umjesto ključne riječi koja se traži u zahtjevu, moguće je postaviti i regularni izraz (engl. *regular expression*). Ovo je ujedno i jedna od najvećih prednosti modula, budući da obrana od napada filtriranjem samo točno određenih riječi može biti prilično neefikasna, a u isto vrijeme i vrlo zahtjevna za administratora sustava. Korištenjem regularnih izraza pravila filtriranja nisu ograničena na jednu riječ, već je moguće definirati složene izraze koji će objediniti različite kombinacije nelegitimnih upita. Iako su regularni izrazi iznimno moćan alat, za potpuno iskorištavanje njihovih mogućnosti potrebno je prilično znanja i iskustva u njihovom korištenju. Pogrešno implementirani izrazi mogu ostaviti lažan dojam sigurnosti, budući da nelegitimni upiti mogu prolaziti potpuno neopaženo. Upravo se iz tog razloga preporučuje detaljno ispitivanje definiranih pravila prije nego što se sustav pusti u rad. Regularni izrazi s obzirom na svoju složenost nisu tema ovog dokumenta, a za čitatelje

koji žele nešto više saznati o ovoj temi mogu se poslužiti literaturom i adresama navedenim u poglavlju Reference (Poglavlje 8).

`SecFilter` direktiva će se naprednijim korisnicima vrlo brzo pokazati kao nedovoljno precizna i preširoka tehnika pretraživanja upita. Direktiva:

```
SecFilterSelective LOKACIJA KLJUCNA_RIJEC [NIZ_AKCIJA]
```

dozvoljava izbor točne lokacije nad kojom će se provoditi pretraživanje. KLJUCNA_RIJEC je već objašnjena, dok će o parametru NIZ_AKCIJA biti nešto više riječi u nastavku dokumenta. Ovdje će biti detaljnije opisan parametar LOKACIJA i njegov značaj u smislu pretraživanja upita.

Parametar LOKACIJA sastoji se od niza identifikatora lokacija odvojenih znakom | (engl. *pipe*). Primjer koji će pretraživati samo IP adrese klijenata i korisnička imena dan je u nastavku:

```
SecFilterSelective "REMOTE_ADDR|REMOTE_USER" KLJUČNA_RIJEČ
```

Parametar LOKACIJA može sadržavati sve CGI varijable koje su i inače dostupne prilikom izrada Web aplikacija. U nastavku su navedene neke od varijabli:

- REMOTE_ADDR;
- REMOTE_HOST;
- REMOTE_USER;
- REQUEST_METHOD;
- PATH_INFO;
- QEURY_STRING;
- AUTH_TYPE;
- DOCUMENT_ROOT;
- SERVER_NAME;
- SERVER_ADDR;
- TIME_YEAR;
- TIME;
- API_VERSION;
- THE_REQUEST.

Osim upravo navedenih parametara, postoje i određene specijalne lokacije koje je moguće koristiti prilikom definiranja sigurnosnih pravila:

- POST_PAYLOAD – filtriranje sadržaja POST zahtjeva;
- ARGS – filtriranje argumenata;
- ARGS_NAMES – filtriranje samo imena varijabli/parametara;
- ARG_VALUES – filtriranje samo vrijednosti varijabli/parametara;
- COOKIES_NAMES - filtriranje samo naziva kolačića;
- COOKIES_VALES - filtriranje samo vrijednosti kolačića;
- HTTP_zaglavlje – pretraživanje zaglavlja zahtjeva "zaglavlje";
- ENV_varijabla – pretraživanje varijable okoline "varijabla";
- ARG_varijabla – pretraživanje varijable zahtjeva "varijabla";
- COOKIE_naziv – pretraživanje kolačića pod nazivom "naziv".

Parametar ARG_varijabla moguće je koristiti i u inverznom obliku kada se koristi u kombinaciji sa ranije spomenutim ARG parametrom. Primjer:

```
SecFilterSelective "ARGS|!ARG_parametar" KLJUČNA_RIJEČ
```

Ovakav filtar pretražiti će sve argumente osim onog pod imenom "parametar".

4.2.1. Izlazno filtriranje

Od 1.7 inačice `mod_security` programa za Apache serije 2.x, modul podržava i izlazno filtriranje. Ovu opciju moguće je uključiti i u seriju Apache poslužitelja 1.x, međutim postupak je nešto složeniji. Korisnicima kojima je neophodna funkcionalnost izlaznog filtriranja preporučuje se nadogradnja na 2.x seriju Apache poslužitelja.

Kako bi se omogućilo izlazno filtriranje, prvo ga je potrebno uključiti sljedećom direktivom:

```
SecFilterScanOutput On
```

Da bi se omogućila primjena izlaznog filtriranja potrebno je prilikom definicije sigurnosnog pravila navesti parametar OUTPUT kao što je to opisano u nastavku:

```
SecFilterSelective OUTPUT "Fatal Error"
```

Izlazno filtriranje obavlja se tek nakon što je zahtjev izvršen.

Od inačice 1.7.2, `mod_security` modul nudi i dodatnu direktivu za izlazno filtriranje, koja omogućava optimizaciju filtriranja u ovisnosti o MIME tipu izlaznih podataka:

```
SecFilterOutputMimeTypes "text/html"
```

Korištenje navedene direktive ograničava filtriranje izlaza samo na one tipove podataka nad kojima ima smisla provoditi kontrolu. U gore navedenom primjeru izlazni filtri primjenjivat će se samo na html datoteke.

4.2.2. Akcije

Postoji nekoliko tipova akcija koje je moguće izvršiti nakon detekcije odgovarajuće ključne riječi u HTTP zahtjevu:

- Primarna akcija odlučuje da li će se zahtjev proslijediti poslužitelju na izvršavanje ili ne. Moguće je imati samo jednu primarnu akciju: *deny, pass* ili *redirect*.
- Sekundarne akcije izvršit će se po uspješnom pronalasku ključne riječi bez obzira na primarnu akciju. Moguće je imati više sekundarnih akcija.
- Akcije toka mogu promijeniti slijed pravila, što će natjerati *mod_security* da izvrši na određenu akciju ili da preskoči par narednih akcija.

4.2.3. Postavljanje akcija

Akcije koje će se provoditi nad paketima koji odgovaraju definiranim sigurnosnim pravilima moguće je definirati na tri različita mjesta.

Akcije navedene u sklopu `SecFilterDefaultAction` direktive provodit će se nad svim upitima koji odgovaraju bilo kojem od definiranih pravila. Sintaksa je sljedeća:

```
SecFilterDefaultAction "deny,log,status:500"
```

Definirane akcije međusobno se odvajaju zarezom, a parametri određene akcije (ukoliko isti postoje) odvajaju se dvotočkom. U primjeru su navedene dvije akcije koje se sastoje od jedne riječi (`deny` i `log`), te treća akcija koja zahtjeva parametar (`status:500`).

Osim definiranja inicijalnih akcija koje se odnose na sve pakete koji zadovoljavaju bilo koje pravilo, moguće je akcije definirati i za svaki filtar posebno:

```
SecFilter KLJUČNA_RIJEČ "deny,status:404"
```

Postoji određeni broj akcija koje je moguće definirati u sklopu sigurnosnih pravila *mod_security* programa. Njihov kratak pregled dan je u nastavku:

- *pass*

Akcija *pass* dozvoljava prolaz zahtjeva iako isti odgovara jednom od definiranih sigurnosnih pravila. Akcija *pass* praktična je u slučajevima kada je potrebno zabilježiti zahtjev bez njegovog blokiranja:

```
SecFilter KLJUČNA_RIJEČ "pass,log"
```

- *allow*

Stroža verzija prethodno opisane akcije *pass*. Nakon provođenja akcije zahtjev će biti propušten bez provjere ostalih pravila:

```
#Automatski propusti korisnika sa korisničkim imenom "silic"
SecFilterSelective REMOTE_USER "silic" "allow"
```

- *deny*

Nakon detekcije upita koji odgovara pravilu sa definiranom akcijom *deny*, upit se blokira te se ne prosljeđuje poslužitelju na izvršavanje. Ukoliko se ne koristi akcija *status, mod_security* će automatski vratiti HTTP 500 kod pogreške.

- *status*

Akcija *status* omogućuje definiranje koda greške koji se klijentu vraća u slučaju kada je zahtjev blokiran.

```
#U slučaju ne prolaska kroz filtar, ova akcija vraća "Page
not found" stranicu
SecFilter KLJUČNA_RIJEČ "status:404"
```

- *redirect*

Preusmjerava korisnika na zadanu URL adresu:

```
SecFilter KLJUČNA_RIJEČ "redirect:hhtp://www.fer.hr"
```

- *exec*

Pokreće izvršavanje zadane izvršne datoteke. Potrebno je navesti punu stazu datoteke:

```
SecFilter KLJUČNA_RIJEČ "exec:/home/silic/skripta.php"
```

- *log*

Bilježenje zahtjeva na uspješan pronalazak ključne riječi filtara u Apache log datoteku.

- *nolog*

Ne bilježi se zahtjev na uspješan pronalazak ključne riječi filtara u Apache log datoteku.

- *skipnext*

Akcija *skipnext* omogućava zaobilazak jednog ili više pravila koje slijedi. Akcija se koristi u slučajevima kada se za pojedine zahtjeve ustanovi da nije potrebno provoditi filtriranje. Ukoliko se kao parametar ne navede broj pravila koje je potrebno zaobići, podrazumijeva se da se radi o jednom pravilu.

```
SecFilterSelective ARG_varijablaX vrijednost1 "skipnext:2"
SecFilterSelective ARG_varijablaX vrijednost2
SecFilterSelective ARG_varijablaX vrijednost3
```

- *chain*

Ova akcija omogućava ulančavanje pravila kako bi se omogućilo provođenje složenijih testova. Samo posljednji filtar u nizu će utjecati na zahtjev, ali zato svi filtri prije njega moraju zadovoljiti usporedbu.

```
#U zahtjevu mora postojati parametar "username" sa
#vrijednošću "admin", te ako je zadovoljeno to pravilo
#provjerava se IP adresa računala koja mora uvijek biti
#ista
SecFilterSelective ARG_username admin chain
SecFilterSelective REMOTE_ADDR "!IP_adresa"
```

- *pause*

Pauziranje u obradi zahtjeva na određeni broj milisekundi. *Pause* opcija može poslužiti u svrhu usporavanja alata za pregledavanje Web poslužitelja, a ukoliko je pauza dovoljno dugačka neki alati mogu biti u potpunosti zaustavljeni.

5. Dodatne opcije `mod_security` modula

Osim upravo navedenih mogućnosti `mod_security` IDS modula, postoje i dodatne opcije, koje osim što olakšavaju rad s modulom ujedno i dodatno povećavaju sigurnost Web poslužitelja. U ovom poglavlju biti će opisane mogućnosti `mod_security` modula, uz samo filtriranje zahtjeva.

5.1. Dodavanje zaglavlja zahtjevima

Kad god je to moguće, `mod_security` će dodati odgovarajuće informacije zaglavljima zahtjeva. Sve te dodatne informacije biti će dostupne za korištenje u postojećim skriptama na Web poslužitelju, što u određenim primjenama može biti vrlo korisno.
Postoje tri tipa zaglavlja koja `mod_security` dodaje procesiranim upitima:

- `mod_security-executed`: zaglavlje sa punom stazom do izvršne datoteke koja je upravo bila pokrenuta;
- `mod_security-action`: zaglavlje sa vraćenim status kodom;
- `mod_security-message`: zaglavlje sa porukom o problemu koje je detektiran. Ova poruka je jednaka onoj koja će biti zabilježena u log datoteku u koju `mod_security` bilježi log zapise.

5.2. Komunikacija sa vatrozidom

Kao što je već ranije spomenuto, `mod_security` osim detekcije neovlaštenih aktivnosti omogućuje i njihovu prevenciju, što je danas sve češća funkcionalnost sigurnosnih alata ovog tipa.
Prevencija napada redovito se bazira na mogućnosti blokiranja prometa sa adrese (ili više njih) s kojih je detektiran maliciozni promet. Blokiranje prometa moguće je realizirati lokalno pomoću alata koji omogućuju filtriranje mrežnog prometa (kao što je npr. iptables program) ili na nivou centraliziranog vatrozida koji štiti računalnu mrežu na kojoj se nalazi poslužitelj. Naravno, u tom slučaju IDS sustav mora imati podršku za komunikaciju s određenim tipom vatrozida. *Mod_security* u trenutnoj inačici podržava prevenciju neovlaštenih aktivnosti korištenjem zasebno razvijenih skripti koje će prema potrebi pozivati iptables/ipchains program s odgovarajućim parametrima.
Iako je u određenim situacijama vrlo korisna, ovakva zaštita ponekad može biti vrlo rizična. Ukoliko neovlašteni korisnik lažira izvornu adresu upita sa adresom legitimnog klijenta, kompletan promet sa te adrese može biti blokiran, što može biti uzrok brojnih problema. Na sličan način, ukoliko su napadi izvršeni preko *proxy* poslužitelja, svi korisnici koji vezu na Internet ostvaruju preko tog *proxy* poslužitelja bit će odbijeni, čak i oni legitimni. Ovakav scenarij može se zaobići pokušajem pronalaska prave adrese napadača preko dodatnih informacija u obliku izvorne adrese klijenta koje *proxy* poslužitelji ponekad pružaju. Međutim, čak i u ovakvim slučajevima mogući su određeni problemi. Napadač se može predstavljati kao *proxy* poslužitelj, koristeći slučajno izabrane, ili čak valjane, adrese kao svoju pravu IP adresu. Ukoliko se počnu odbijati zahtjevi na osnovu takve informacije, napadač će samo nastaviti mijenjati IP adresu i nastaviti sa napadima. Rezultat ovakve situacije bit će zabrana korištenja Web poslužitelja legitimnih korisnika, dok napadač još uvijek slobodno odašilje napade na aplikaciju.
Iz upravo navedenih razloga, ovakva metoda prikladna je ukoliko se Web aplikacijama na štićenom poslužitelju ne dozvoljava pristup putem *proxy* poslužitelja, odnosno ukoliko se dozvoljava pristup sa dobro poznatih, i još važnije, povjerljivih *proxy* poslužitelja.
Ukoliko se, bez obzira na moguće probleme, i dalje želi implementirati sustav za prevenciju od neovlaštenih aktivnosti, potrebno je implementirati odgovarajuću skriptu koja će se pokretati prilikom detekcije malicioznih upita. Skripta bi trebala iz varijabli okoline doći do podataka o IP adresi malicioznog korisnika, izvršiti poziv `iptables` ili `ipchains` programa te na taj način zabraniti promet s dotične IP adrese. Ovakva skripta u trenutnoj verziji *mod_security* modula ne postoji, te ukoliko korisnik želi koristiti ovu metodu zaštite, morati će sam razviti specijalno prilagođenu skriptu. Skriptu koja bi omogućila automatsko blokiranje prometa putem `iptables` programa se planira u budućim inačicama *mod_security* modula uključiti kao sastavni dio paketa.

5.3. Maskiranje identiteta poslužitelja

Uobičajeno je da Web poslužitelji u zaglavlju otkrivaju svoj identitet kao dio svakog HTTP odgovora. Vrlo često korištena tehnika za zbunjivanje i usporavanje napadača je lažiranje identiteta Web poslužitelja. Lažiranjem identiteta Web poslužitelja može se znatno otežati provođenje malicioznih aktivnosti, a manje iskusni napadači mogu se vrlo jednostavno usmjeriti u potpuno pogrešnom smjeru. Apache poslužitelj je na ovom području vrlo pogodan za napade, jer ukoliko se u konfiguracijskoj datoteci ne isključe direktive zadužene za povrat informacija (`ServerTokens` i `ServerSignature`), Apache će ne samo vratiti svoju punu inačicu, već će i svim modulima koji su instalirani uz njega dozvoliti da nadodaju ispis svojih inačica:

```
[root@linux conf]# telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 23 Mar 2004 14:39:25 GMT
Apache/1.3.29 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7b PHP/4.3.3
…
```

Kako bi se umanjila količina informacija koju poslužitelj otkriva potrebno je promijeniti upravo spomenute direktive, što ne predstavlja preveliki problem. Međutim, ukoliko se u potpunosti želi prikriti identitet Apache Web poslužitelja, javljaju se određeni problemi, budući da je spomenutim direktivama moguće utjecati samo na podatke koji se ispisuju nakon riječi Apache (to znači da nije moguće Apache poslužitelj maskirati kao npr. Microsoftov IIS poslužitelj). Za potpuno uklanjanje svih informacija moguće je promijeniti izvorni kod programa te ga ponovno prevesti ili učiniti odgovarajuće promjene na samoj binarnoj inačici programa (korištenjem specijaliziranih alata).

Od inačice 1.7 `mod_security` modul nudi opciju koja omogućuje vrlo jednostavnu promjenu identiteta poslužitelja:

```
SecServerSignature "Microsoft-IIS/5.0"
```

Iako ova opcija radi savršeno, ona može zbuniti samo manje iskusne napadače, budući da je naprednijim tehnikama moguće dodatno provjeriti o kojem se točno poslužitelju radi. Najavljuje se nastavak rada na ovoj opciji `mod_security` modula sa ciljem poboljšanja maskiranja pravog identiteta poslužitelja.

Bitno je napomenuti da će ova direktiva raditi samo u slučaju kada je konfiguracijska direktiva Apache poslužitelja `ServerTokens` ostavljena, odnosno postavljena na "Full".

5.4. *Chroot* funkcionalnost

Nakon što se obavi *chroot* poziv, aplikacija više nije u mogućnosti pristupati datotekama koje se nalaze iznad njoj dodijeljenog direktorija. Ideja je da, ukoliko napadač i uspije ostvariti pristup Web poslužitelju, ne može učiniti mnogo štete jer se nalazi u samo jednom direktoriju iz kojeg ne može van. Moguće je pokrenuti Apache poslužitelja uz *chroot* ograničenje, međutim, problemi mogu nastati ako aplikacije za svoj ispravan rad moraju pristupati raznim dijeljenim bibliotekama i izvršnim datotekama koje se nalaze van *chroot* direktorija na koji je Apache ograničen.

Od 1.5.1 inačice, *mod_security* ima ugrađenu funkcionalnost *chroot* poziva za Apache poslužitelj. Sve što je potrebno učiniti je dodati sljedeću direktivu u konfiguracijsku datoteku `mod_security` programa:

```
SecChrootDir /chroot/apache
```

Osim što je vrlo jednostavna, ova mogućnost za razliku od standardnog načina implementacije *chroot* funkcionalnosti ne zahtjeva postojanje dodatnih datoteka unutar određenog direktorija. *Chroot* poziv se obavlja prije inicijalizacije samog Web poslužitelja, ali poslije `fork` poziva sustava, te će iz tog razloga sve dijeljenje datoteke već biti učitane, svi Web moduli će biti inicijalizirani i log datoteke otvorene. Sve što je potrebno u dodijeljenom direktoriju su datoteke samih Web aplikacija.

Ukoliko je potrebno izvršavati CGI skripte ili datoteke sustava, tada je potrebno obaviti *chroot* na standardan način, jer ovdje `mod_security` ne može pomoći.

Kako bi sve radilo ispravno, *chroot* poziv je potrebno obaviti u točno određenom trenutku pokretanja Apache poslužitelja, odnosno tek nakon što su svi moduli inicijalizirani. Iz tog razloga, `mod_security` mora biti prvi na listi modula. Kako bi se to osiguralo, potrebno je načiniti promjene u poretku modula tako da se `mod_security` stavi na prvo mjesto. Za detaljnije upute o tome kako obaviti sve potrebne preinake preporučuje se proučavanje dokumentacije za određenu inačicu `mod_security` modula.

5.5. Bilježenje zahtjeva

Standardno bilježenje koje pruža Apache Web poslužitelj ne pomaže mnogo ukoliko je potrebno detaljnije analizirati izvore i tehnike koji su korišteni prilikom provođenja malicioznih aktivnosti. Razlog tome je relativno mala količina informacije o zahtjevu koja biva zapisana u log datoteku. `Mod_security` pruža mogućnost detaljnijeg bilježenja zahtjeva upotrebom dvije direktive:

```
SecAuditEngine On
SecAuditLog logs/audit_log
```

Prva direktiva određuje što će biti zabilježeno kroz sljedeće parametre:

- `On` – bilježi se svaki zahtjev;
- `Off` – zahtjevi se ne bilježe;
- `RelevantOnly` – bilježe se samo važni zahtjevi, odnosno oni zahtjevi kod kojih je pronađena odgovarajuća ključna riječ;
- `DynamicOrRelevant` – bilježe se samo bitni zahtjevi ili dinamički generirane stranice.

Druga direktiva određuje u koju datoteku će biti zabilježeni zahtjevi.
Ovo je primjer zapisa jednog zahtjeva:

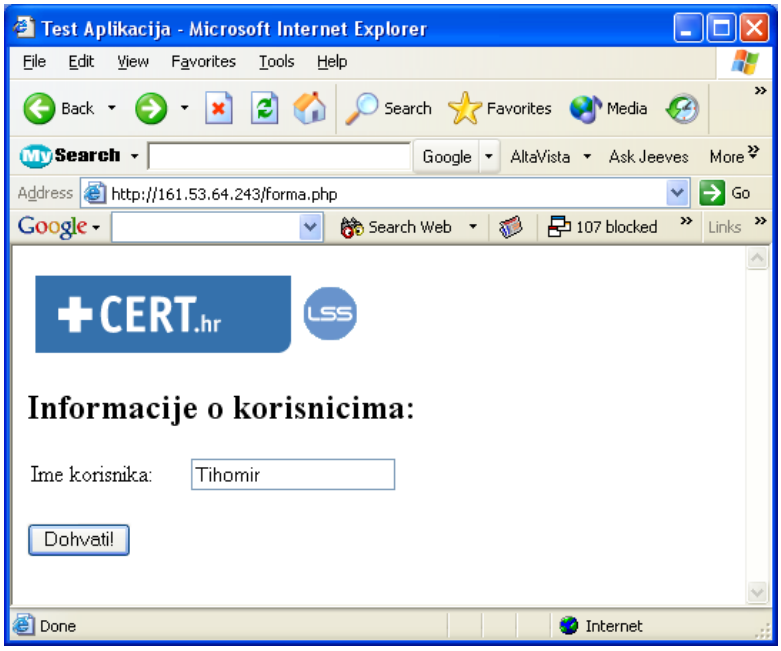
```
============================================
Request: 66.143.227.204 - - [Fri Apr  2 06:24:37 2004] "GET /
HTTP/1.1" 200 65
Handler: (null)
--------------------------------------------
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
*/*
Connection: Keep-Alive
Host: 161.53.64.243
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)

HTTP/1.1 200 OK
Last-Modified: Sat, 27 Mar 2004 10:36:44 GMT
ETag: "3f67c-41-4065593c"
Accept-Ranges: bytes
Content-Length: 65
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
============================================
```

Prva linija jednaka je onome što Apache poslužitelj zapisuje u svoju log datoteku. Druga linija sadrži naziv programa koji je zadužen za obradu zahtjeva. Nakon odjeljivanja slijedi cijeli zahtjev uključujući zaglavlja koje je nadodao `mod_security`, dok je odgovor poslužitelja dan nakon jednog praznog reda.
Ukoliko je uključeno POST filtriranje, POST sadržaj će uvijek biti uključen u zapisu.

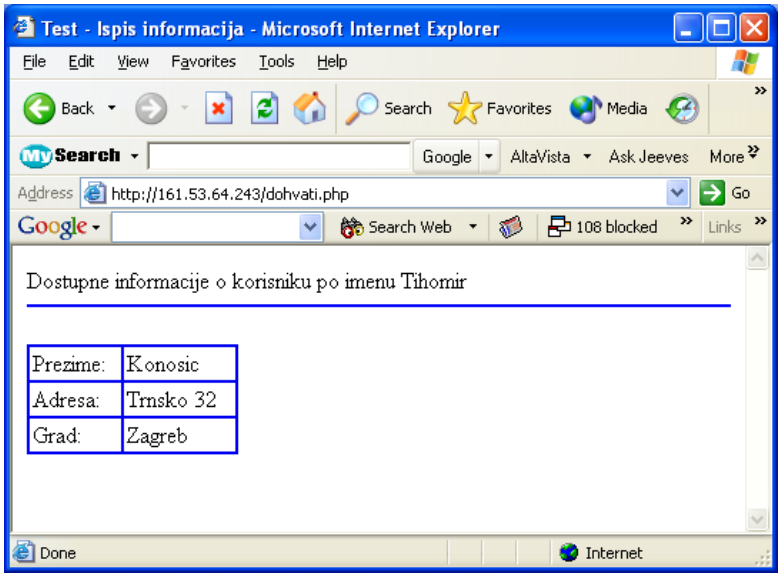
6. Testiranje

U svrhu testiranja *mod_security* programa razvijena je jednostavna aplikacija čije je sučelje prikazano na slici 1. Aplikacija nudi jednostavnu interakciju s korisnikom putem Web formi, te se za uneseni izraz, u ovom slučaju ime korisnika, ispisuju svi dostupni podaci o traženom korisniku.



Slika 1: *Glavno sučelje testne aplikacije*

Aplikacija je realizirana u PHP programskom jeziku, dok su podaci pohranjeni u dvije relacije MySQL baze podataka. Aplikacija posjeduje dvije inačice, pri čemu je razlika jedino u načinu prijenosa korisničkog unosa: jedna forma koristi POST metodu, dok druga koristi GET metodu. Dvije inačice baze razvijene su kako bi se što detaljnije ispitale pojedine funkcionalnosti modula.



Slika 2: *Rezultati izvršenja upita*

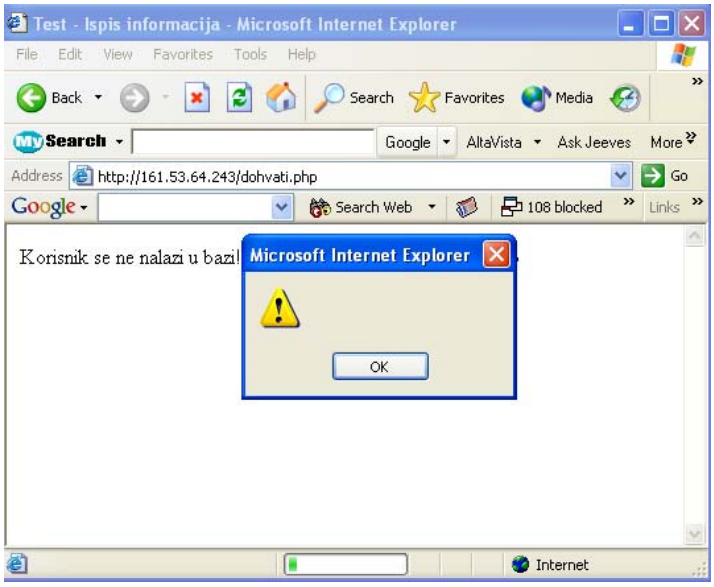
Ovakva aplikacija dovoljna je za pokušaj izvršavanja gotovo svih trenutno popularnih napada (*Cross Site Scripting*, *SQL injection* i sl.). Testiranje *mod_security* modula izvršeno je upotrebom alata za ispitivanje sigurnosti Web aplikacija (Nikto, TrustSight Security Scanner, Paros, itd.)

6.1. *Cross Site Scripting* napad

Ranjivost aplikacije na ovaj tipa napada moguće je provjeriti unosom vrlo jednostavnog JavaScript koda u Web formu:

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

Bez uključenog filtriranja ovaj dio koda će se upisati u tijelo aplikacije i izvesti, pri čemu će se u "*Alert*" prozoru ispisati sadržaj kolačića (*engl. cookie*), ukoliko on postoji (Slika 3.).



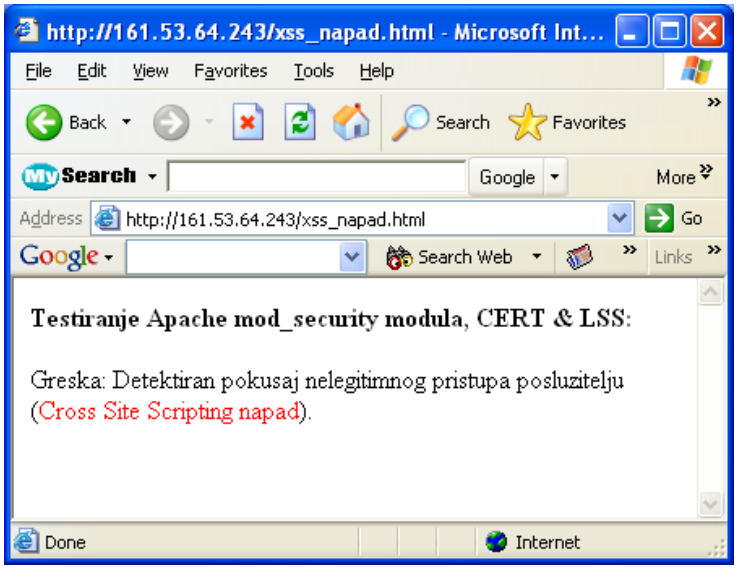
Slika 3: *Rezultat XSS napada bez uključenog filtera*

Ovakav tip napada moguće je zaustaviti prilično jednostavno pomoću `mod_security` filtra. Potrebno je pažljivo definirati sigurnosna pravila koja će detektirati maliciozni programski kod, dok akcije koje se provode nakon detekcije mogu biti različite, ovisno o specifičnim zahtjevima sustava. Moguće je obaviti redirekciju klijenta na određenu stranicu, pozvati izvršenje neke unaprijed pripremljene skripte ili napadaču zabraniti ispis stranice i proslijediti neku od standardnih HTTP grešaka:

```
#Redirekcija na posebno pripremljenu stranicu
SecFilter "<[[:space:]]*script" log,redirect:xss_napad.html
#Poziv za izvršenje skripte
SecFilter     "<[[:space:]]*script"     log,exec:/home/silic/
xss_napad.php
#Vraćanje standardne HTTP greške 403 «Forbidden»
SecFilter "<[[:space:]]*script" "deny,log,status:403"
```

Potrebno je napomenuti da je, ako se koristi POST metoda, nužno uključiti provjeru POST sadržaja kako bi ovakav napad bio uspješno detektiran.

Slika 4 prikazuje rezultat redirekcije prilikom detektiranog napada.



***Slika 4**: Rezultat XSS napada sa uključenim filterom*

Ukoliko je bilježenje uključeno, detektirani zahtjev biti će zapisan u log datoteci na sljedeći način:

```
========================================
Request: 161.53.64.73 - - [Wed Mar 31 08:12:13 2004]
"POST/dohvati.php HTTP/1.1" 302 276
Handler: (null)
Error: mod_security: Access denied with redirect to
[xss_napad.html]. Pattern match "&lt;[[:space:]]*script" at
POST_PAYLOAD.
----------------------------------------
```

Ukoliko se filtriranje želi proširiti kako bi se onemogućilo ubacivanje i svih ostalih HTML oznaka, potrebno je drukčije definirati regularni izraz filtera:

```
SecFilter " <(.|\n)+>" log,redirect:novi_tag.html
```

6.2. *Directory traversal* napad

Ukoliko Web aplikacije na bilo koji način koriste datoteke sustava, posebnu pažnju potrebno je posvetiti detekciji specijalnih znakova unutar HTTP upita. Kombinacija znakova "../" u putanji je zahtjev za prelazak u direktorij jedan nivo iznad trenutnog, što je proteklih godina bila jedna od standardnih tehnika za izlazak iz direktorija u kojem se nalaze Web stranice koje poslužitelj prikazuje. Budući da u normalnim operacijama nema potrebe za ovakvom kombinacijom znakova, lako ju je onemogućiti jednostavnim pravilom:

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

Prilikom provođenja malicioznih aktivnosti specijalni znakovi često se kodiraju *hex* ili *Unicode* tehnikom, ne bi li se zavarali sustavi za detekciju neovlaštenih aktivnosti.

Gore navedeno pravilo biti će dovoljno za detekciju ovakvih napada, ukoliko je u konfiguracijskoj datoteci postavljena direktiva koja daje `mod_security` modulu upute za provjeravanje svih načina kodiranja:

```
SecFilterCheckURLEncoding On
SecFilterCheckUnicodeEncoding On
```

Prilikom testiranja mogućnosti detekcije *directory traversal* napada uočeni su određeni problemi, budući da neki od testnih upita sa kombinacijom "../" znakova nisu bili detektirani.

Za detekciju *directory traversal* napada koristilo se pravilo preporučeno dokumentacijom `mod_security` programa. Iako su definirana i dodatna pravila kako bi se napadi ovog tipa uspješno detektirali, određeni maliciozni upiti uspjeli su proći nezapaženo od strane `mod_security` modula. Može se pretpostaviti da je do problema došlo uslijed kreiranja zahtjeva koji su sadržavali specijalno

oblikovani niz znakova, koji je pri prolasku kroz normalizaciju izgubio svoj originalni smisao. Bez obzira na postupke normalizacije upita i činjenicu da danas ovakvi napadi rijetko prolaze i pored samostalnog Apache poslužitelja, propuštanje zahtjeva koji bi prema sigurnosnim pravilima modula trebali biti blokirani smatra se propustom u implementaciji.

6.3. *SQL Injection* napad

Velika većina današnjih Web aplikacija kao neizostavni element koristi određenu bazu podataka u kojoj se pohranjuju podaci bitni za rad same aplikacije. Ukoliko se prilikom dizajna Web aplikacije (i baze podataka) nije vodilo dovoljno računa o problemima sigurnosti (provjera korisničkog ulaza i sl.), Web aplikacije vrlo lako postaju ranjive upravo zbog svoje veze sa bazom podataka koja postaje meta napada.

Jedan od najpopularnijih napada u ovom smislu je tzv. *SQL injection* koji izlazi van opsega ovog dokumenta. Zainteresirani čitatelji mogu nešto više o *SQL Injection* napadima pronaći u poglavlju s Referencama (Poglavlje 8).

Korištenjem `mod_security` modula ovakav napad je vrlo lako spriječiti korištenjem regularnih izraza. Osim regularnih izraza, potrebno je dobro poznavati i bazu s kojom aplikacije komuniciraju te sve potencijalno slabe točke implementacije koje napadač može iskoristiti.

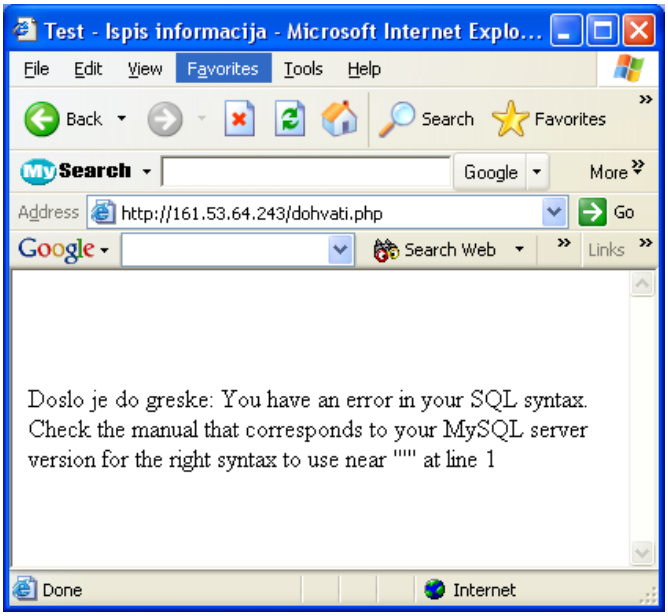
Regularni izraz koji se može upotrijebiti za detekciju *SQL Injection* napada dan je u nastavku:

```
# SQL injection napadi
SecFilter "select.+from"
SecFilter "delete [[:space:]] +from"
SecFilter "insert [[:space:]] +from"
```

Navedeno pravilo uspješno će zaustaviti većinu napada umetanjem SQL naredbi, budući da se njime onemogućuje unošenje SQL naredbi u forme s kojima korisnik pristupa bazi podataka (`select`, `delete` i `insert` SQL naredbe). Pravila filtriranja potrebno je pažljivo osmisliti i prilagoditi strukturi i načinu rada baze podataka.

Prilikom provođenja SQL napada, napadaču je dovoljno da Web poslužitelj javi grešku u SQL upitu, čime dolazi do korisnih informacija o ranjivosti Web aplikacije. Za vješte maliciozne korisnike pitanje je trenutka kada će na temelju dobivenih informacija osmisliti specijalno prilagođeni upit koji će omogućiti dolazak do podataka u bazi.

Kod testne Web aplikacije, u polje za unos bilo je dovoljno unijeti bilo koji specijalni znak koji će proizvesti grešku u SQL sintaksi. Rezultat nakon unosa jednostrukog navodnika prikazan je na slici 5.

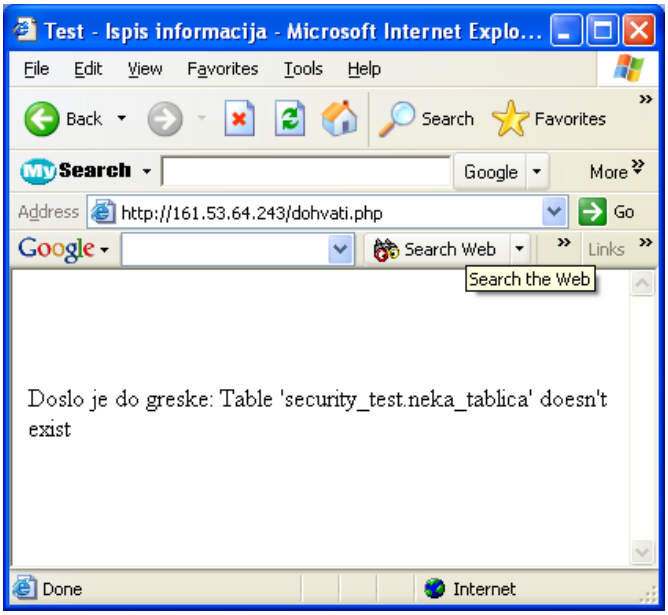


Slika 5: *Rezultat prvog koraka SQL Injection napada*

Napadač na temelju ovog ispisa saznaje da Apache Web poslužitelj radi sa MySQL bazom podataka. Nakon određenog broja pokušaja napadač može naslutiti kakva vrsta upita se provodi nad bazom te nastavlja svoj napad korištenjem sljedećeg niza:

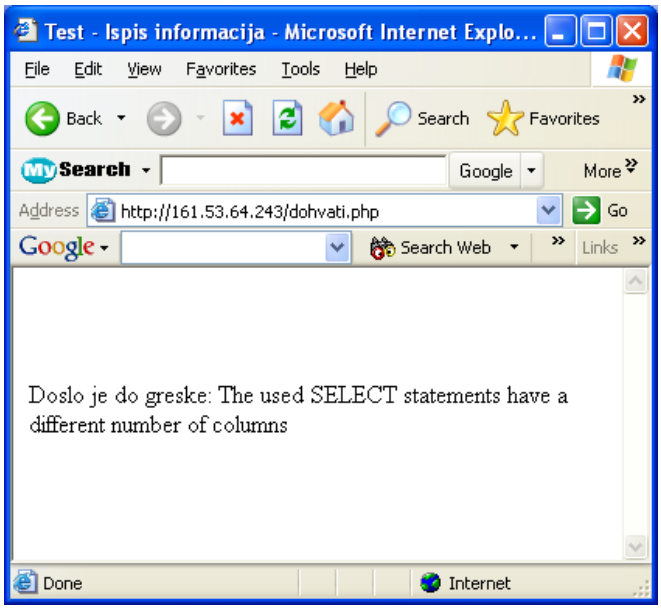
```
' UNION ALL SELECT * FROM neka_tablica WHERE ''='
```

Prvi navodnik zatvara varijablu za upit, nakon čega se u tablici traži prazan niz, pri čemu se ujedno provodi i unija sa još jednim `select` upitom koji je uvijek istinit jer se u `where` dijelu upita kompariraju prazni nizovi. Poslužitelj vraća grešku prikazanu na slici 6.



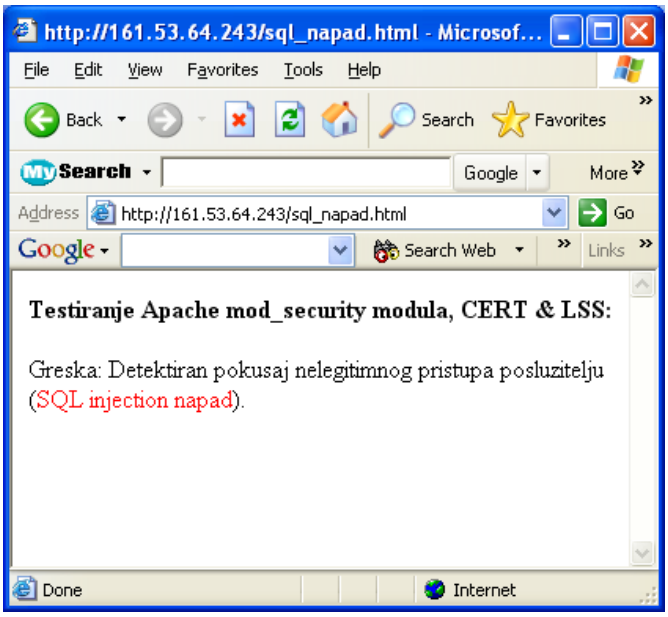
Slika 6: *Rezultat drugog koraka SQL Injection napada*

Napadač doznaje naziv baze, "*security_test*", te da u njoj ne postoji tablica pod nazivom "*neka_tablica*". Nastavkom napada moguće je doći do tablice koja postoji, a rezultat provođenja malicioznog upita prikazan je na slici 7.



Slika 7: *Rezultat trećeg koraka SQL Injection napada*

Napadaču preostaje doznati koliko entiteta se čita iz tablice, da bi svoj `select` upit mogao prilagoditi i uspješno ga izvršiti.
Opisani napad vrlo je jednostavan primjer *SQL Injection* napada. Postoje razne varijacije ovog napada koje neovlaštenim korisnicima omogućavaju ne samo čitanje pohranjenih podataka, već i ubacivanje vlastitih podataka ili brisanje postojećih.
Nakon što je `mod_security` modul uključen i nakon implementacije sigurnosnih pravila za sprječavanje SQL napada, napad izveden na testnoj Web aplikaciji biti će onemogućen. Rezultat pokušaja napada prikazan je slikom 8.



Slika 8: *Rezultat SQL Injection napada sa uključenim filterom*

Za ovakav rezultat, u konfiguracijskoj datoteci je prilikom detektiranja regularnih izraza koji sadrže SQL naredbe `select`, `insert` i `delete`, kao akcija postavljena redirekcija na prethodno pripremljenu stranicu `sql_napad.html`. Konačna odluka što će biti učinjeno u slučaju detekcije napada ostaje na administratoru sustava.
Zapis u log datoteci je sljedeći:

```
========================================
Request: 161.53.64.73 - - [Fri Apr 2 04:40:27 2004] "POST
/dohvati.php HTTP/1.1" 302 276
Handler: (null)
Error: mod_security: Access denied with redirect to
[sql_napad.html]. Pattern match "select.+from" at
POST_PAYLOAD.
----------------------------------------
```

6.4. *Mod_security* skripta za testiranje

U sklopu `mod_security` modula distribuira se i skripta za njegovo testiranje. Kako bi se testirala konfiguracija, potrebno je osmisliti i realizirati HTTP zahtjeve koji će se prosljeđivati poslužitelju na izvršavanje. Pokretanje skripte bez argumenata ispisuje način njenog korištenja:

```
[root@linux tests]# ./run-test.pl
Usage: ./run-test.pl host[:port] testfile1, testfile2, ...
```

Prvi parametar je naziv poslužitelja sa opcionalnim dodatnim parametrom port, u slučaju da se zahtjevi ne žele poslati na port 80 koji je inače standardan za Web servis. Potrebno je navesti i datoteke u kojima se nalaze zapisani specijalno formirani, testni, HTTP zahtjevi.

Zaglavlja koja skripta za testiranje sama dodaje zahtjevu su: `Host`, `User-Agent` i `Connection`, te ova zaglavlja nije potrebno samostalno ugrađivati u vlastite zahtjeve.
HTTP zahtjev koji je potrebno smjestiti u datoteku može izgledati na sljedeći način:

```
# Zahtjev za provjeru jednostavnog filtera sa ključnom riječi
"/etc/passwd"
GET /etc/passwd HTTP/1.0
```

Poziv programa i rezultat njegovog izvršavanja jasno otkriva da definirano pravilo uspješno radi i da vraća status 403, kako je i navedeno u `httpd.conf` konfiguracijskoj datoteci.

```
[root@linux tests]# ./run-test.pl 127.0.0.1 test1.test
Test "Zahtjev za provjeru jednostavnog filtera sa kljucnom
rijeci "/etc/passwd"": Failed (status = 403)
```

Moguće je i kreirati složenije zahtjeve na sličan način:

```
# Pokušaj poziva cgi skripte post metodom
POST /cgi-bin/modsec-test.pl HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 16
```

Skripta vraća status 200 kao rezultat uspješnog izvršavanja zahtjeva.
Unutar direktorija u kojem je smještena skripta nalazi se veliki izbor već gotovih HTTP zahtjeva sa specifičnim napadima, čime je korisnicima olakšan posao testiranja sigurnosti vlastitog Web poslužitelja.

7. Zaključak

Web aplikacije danas obavljaju sve složenije i zahtjevnije zadatke, te je i njihovoj zaštiti potrebno pridati veću pažnju. Iz skupine dostupnih alata koji se bave ovom problematikom, `mod_security` se pokazuje kao dobar izbor za zaštitu Web poslužitelja. Instalacija prolazi bez ikakvih problema, te se kao jedini potencijalni nedostatak pokazuje potreba za ponovnom reinstalacijom Apache poslužitelja, što na sustavima sa velikim brojem već instaliranih modula ponekad može prouzročiti probleme.

Dokument ukratko opisuje problematiku sigurnosti Web aplikacija, nakon čega je dan detaljan pregled mogućnosti za detekciju i prevenciju neovlaštenih aktivnosti usmjerenih prema Apache Web poslužiteljima. Uz brojne funkcionalnosti koje alat nudi, opisani su i postupci njegove instalacije i konfiguracije, kao i rezultati testiranja provedenih u svrhu ispitivanja mogućnosti.

Ispravna konfiguracija zahtjeva određeni nivo poznavanja problematike sigurnosti Web aplikacija, međutim, jednom podešene postavke uspješno će štititi Web poslužitelj od različitih tipova napada. Jedna od kvaliteta `mod_security` alata je podrška za regularne izraze, što pruža iznimnu slobodu prilikom definiranja sigurnosnih pravila.

Testiranje je pokazalo da jedini problemi sa kojima se `mod_security` susreće nastaju prilikom pokušaja *directory traversal* napada. Uslijed zaprimanja zahtjeva sa određenom kombinacijom znakova, koju bi modul trebao prepoznati kao nepoželjnu, blokada i nepropuštanje zahtjeva se ne događa, već zahtjev biva proslijeđen Apache Web poslužitelju. Ovo je ujedno i jedini nedostatak koji je zamijećen prilikom testiranja, dok su svi ostali pokušaji napada bili uspješno zastavljeni i prijavljeni.

8. Reference

Introduction to mod_security, http://www.onlamp.com/pub/mod_security.html,
Security Focus article, http://www.securityfocus.com/infocus/1739,
Reference manual, http://www.modsecurity.org/documentation/modsecurity-manual-1.7.4.pdf,
Using regular expressions, http://etext.lib.virginia.edu/helpsheets/regex.html,
Detection of SQL Injection and XSS Attacks, http://www.securityfocus.com/infocus/1768,
XSS FAQ, http://www.cgisecurity.com/articles/xss-faq.shtml,
SQL Injection, http://www.spidynamics.com/whitepapers/WhitepaperSQLInjection.pdf