Programsituationer

Lars Sommer < *lasg@lasg.dk* >

11. september 2007

Resumé

Noter til bogen Mastering FreeBSD and OpenBSD Security del 2

Indhold

Kapitel 1

Webserver

1.1 Angreb

En webserver står ofte åben for hele verden, så folk kan komme og se på hjemmesiderne. Dette medfører at webservere ofte er udsat for angreb. Årsagerne til angreb mod webservere kan eksempelvis være:

- At vise sig for andre hackere
- At hænge ejeren af serveren ud
- At stjæle information, som måske kun få udvalgte normalt har adgang til
- At bruge webserveren som mellemled til et større angreb inde i netværket

Der er forskellige tekniske mål man kan opnå ved at angribe en webserver, eksempelvis

- Deface af siden, hvor forside eller hele webstedets indhold er udskiftet
- Denial of Service, hvor webserveren gøres utilgængelig for almindelig trafik
- Shell/CMD exploit, hvor man via webserveren får adgang til andre dele af serveren

Der er mange forskellige midler til at opnå disse mål. Hvordan de enkelte fungerer, er ikke så relevant i denne sammenhæng.

1.2 Særlige trusler

1.2.1 Arbitrær programkørsel

Denne truselstype er meget udbredt og meget skadelig. Angriberen kan få lov at køre programmer eller kode efter eget valg, via huller i, typisk, cgi-programmer eller php-kode. Angriberen kan enten køre de programmer der allerede ligger på serveren (og give sig selv ssh-, eller telnet-adgang), eller selv uploade rootkits og lignende. Selv hvis angriberen ikke kan køre en shell umiddelbart, kan alle exploits der normalt kun gælder for lokalt brug, pludselig være lige så skadelige som de til fjernbrug.

1.2.2 Programmisbrug

SQL-, HTML-, Java-script-kode og XSS kan bruges helt uden operativsystemet eller netværket er klar over det. Derfor er det ekstremt vigtigt, at man stoler på de webapplikationer man benytter sig af, og spærrer disse i chroot eller jails, hvis de ikke er 100% til at stole på.

Måske kan angriberen ikke direkte få adgang til serveren gennem sådanne programmer, men i stedet fiske andre brugere cookies med login-informationer, sende spam via f.eks. en formmail.pl og lignende.

1.2.3 Fillækage

Nogle webservere eller webapplikationer er ikke opmærksomme på alle typer af encoding, og kan måske narres til at lade angriberen downloade bruger-databasen, når den bare encodes korrekt.

Mulighed for upload af data er lige så slemt. Der kan eksempelvis uploades ondsindede programmer, som kan ramme enten lokale, eller fjernsystemer hvor trafikken så kommer fra din server. Der kan overskrives vigtige filer, og sidst kan der lægges ulovligt materiale op, som andre så kan hente fra dig.

1.3 Apache

Hvis du kan nøjes med en lille simpel webserver, uden SSL, store dynamiske sider osv, er thttpd måske mere attraktiv. Men ellers apache.

Der er to store versioner; 1.3 og 2. Hvis du allerede kører et stort og stabilt site med 1.3, er det ok at blive, men starter du noget nyt, så brug endelig version 2. Et par fordele ved 2, er at threading og generel performance er stort forbedret. Den understøtter multitrådede moduler, og har en del moduler inkluderet som standard (SSL f.eks.) Konfigurationen af 2 er også nemmere. Moduler kan slåes til og fra med een enkelt linje pr modul, og behøver ikke stå i korrekt rækkefølge som i version 1.

1.3.1 Installation

I OpenBSD er det stadig en chrooted apache 1.3 der følger med. Version 2 findes ikke endgang i ports. I FreeBSD er der ingen med som standard, men adskillige forskellige versioner i ports. Et godt valg i FreeBSD, vil være www/apache2.

"make show-options"viser en hel del installationsparametre. For at bygge apache med disse, køres f.eks. "make WITH_SUEXEC SUEXEC_UIDMIN=500 SUEXEC_DOCROOT=/var/www"

Efter installationen, er det smart at sætte "WITH_APACHE2=YES"i /etc/make.conf, således diverse webapplikationer ved de bør bygge til version 2, og ikke 1, som de ellers defaulter til, hvis denne option ikke er sat.

1.3.2 Konfiguration

Generelt gælder det om at tillade mindst muligt globalt, og så tillade de enkelte ting brugerne har behov for, i deres respektive visual hosts.

Brugertilladelser

Mange globale options kan ændres af den enkelte bruger, gennem .htaccess. Hvis du har sat en række globale regler i httpd.conf, kan brugerne ændre disse via .htaccess, hvilket næppe er ønskeligt. Derfor er det smart at nægte

brugen af .htaccess, med AllowOverride None.

Hvis du ønsker at php kun skal være tilgængeligt for de brugere, der har behov for det, kan du slå php fra globalt med "php_flag engine off", og så tilsvarende "php_flag engine on"i de enkelte virtual hosts hvor det er krævet.

Beskyt kritiske filer

Nogle filer bør beskyttes ekstra godt. Dette gælder bl.a. SSL-filer, httpd.conf, .ht*.

SSL-filerne server.key (Den private nøgle) og server.crt (Det offentlige certifikat) bør kun være læsbare af root. Det kan med fordel sættes med ACL, og dernæst et filesystem flag som system immutable. Men husk at server.crt måske skal fornyes om et år eller to, alt efter ens CA.

Med hensyn til httpd.conf, kan man f.eks. have alle de statiske linjer i httpd.conf, og låse denne fil ned til det nødvendige, også med schg-flaget. Og så have alt det der ofte ændres, vhosts f.eks., i en fil for sig, som så er inkluderet med Include direktivet i httpd.conf

.ht*-filerne (.htaccess, .htpasswd, .htgroup og lign.) bør beskyttes mod download. F.eks. med dette i httpd.conf:

```
<Files ~ "^\.ht">
  Order allow,deny
  Deny from all
  Satisfy All
</Files>
```

Derudover kan det være en lille (obskuritets)forbedring, at give de filer man benytter til .ht*-formål andre navne, og opbevare dem udenfor DocumentRoot.

DoS-beskyttelse

Mange (mail)serverprogrammer har indbyggede funktioner til tar-pitting, som beskyttelse mod DoS-angreb. Det har apache ikke. Som lille hjælp, kan man justere optionen MaxClients i httpd.conf

Ved at trække en række tunge sider fra apache, og dermed give den lidt arbejde, kan man se et relativt realistisk ram-forbrug med top. Start top, og se på RES (resident). Se hvad de gennemsnitsligt bruger, og vurder hvor meget ram du højst vil have apache må bruge. Del denne mængde med gennemsnitsmængden for en httpd-tråd, og sæt da MaxClients til noget nær dette resultat. Det kan self. afvige en del, hvis du kører sider med specielle krav for CPU eller ram.

1.3.3 Moduler

Der er mange moduler, og mange meget usikre og meget anvendte. Nogle af de primære beskrives kort her. I stedet for at tage en standardkonfiguration, og slå de moduler fra, du ikke lige mener du har behov for, så start i stedet med ingenting, og sæt kun de til, som du ved du har behov for nu og her. Filen highperformance.conf følger med apache, og indeholder en minimalkonfiguration, for høj ydeevne. Man kan f.eks. tage udgangspunkt i denne.

mod_cgi

Normalt har /cgi-bin/ en rimelig konfiguration i httpd.conf, men vær sikker på at AllowOverride None og Options None er sat, således man ikke kan tilgå listevisning for denne mappe.

Generelt er det smart at køre cgi-programmerne som almindelige brugere. Det kan mod_suexec og cgiwrap hjælpe med. Disse omtales senere.

mod_php

Der er enormt mange webapplikationer i php, der er meget anvendte og meget dårligt og usikkert skrevet.

PHP køres normalt som brugeren www, og dårlig kode kan resultere i at brugeren kan det samme som www. Normalt kan www kun skrive i /tmp, men det kan også være nok til at indsætte diverse exploits og lignende. Derfor kan det være smart udelukkende at benytte databaser som mysql til php's skrivebehov.

PHP kan også bruges som interpreter gennem CGI, og altså ikke med modulet mod_php. Dette er meget muligt langsommere, men mere sikkert. På denne måde kan PHP afvikles gennem mod_suexec eller cgiwrap, som andre cgi-programmer.

PHP-konfigurationen ligger i php.ini, men der kan laves overrides af denne i httpd.conf, og igen i .htaccess, hvis det tillades. Derfor er det vigtigt at sætte AllowOverride til None, så brugerne ikke bare kan sætte om på ens valgte PHP-options.

Et par options, der kan være smarte at sætte i PHP-konfigurationen, er: register_globals=off, så variable i URL'en, ikke direkte indvirker i PHP-scriptet. display_errors=off, uddybende fejlmeddelelser er guld værd for angribere. Selv med denne slået fra, kan administratorerne se fejllog i apaches error log. variables_order="GPCS", Get Post Cookie Server. Hvis du f.eks. aldrig skal bruge Get, kan denne sikres lidt med kun at sætte til "PCS"

mod_perl

Perl er meget fornuftigt anvendt til meget cgi. Mange moduler er skrevet i C(++), og kan derfor nemt byde på diverse buffer overflows og lign., hvis disse er skrevet dårligt. Perl kan køres som cgi, men det giver en noget dårlig performance, når interpreteren skal loades hver gang et script skal køres. Derfor er mod_perl smart. Det loader perl ind, når webserveren startes, og giver dermed ret store performanceforbedringer.

Som ekstra sikkerhed bør PerlTaintCheck On sættes i httpd.conf, for at sikre at ting der går gennem mod_perl, overholder perls "taint-regler (perl -t). Disse kontrollerer at scriptet behandler alle input-data, før de bruges. Det sikrer imod en del input-baserede angreb. Se "man perlsec"for mere info om perls tainting.

Andre moduler

mod_dav giver mulighed for HTTP options SEARCH, PROPFIND og lign. Der har været mange sikkerhedsmæssige problemer med disse, så tag det endelig væk, med mindre du har behov for det.

mod_include giver SSI-mulighed. F.eks. #include og #exec. Sidstnævnte kan hurtigt resultere i skadelige kommandoer. Slå det fra hvis du ikke har behov for det.

mod_info og mod_status giver en del brugbar debugging-information, men også en masse god information for en angriber. Slå dem fra, sålænge du ikke debugger systemet. Når du bruger dem, så tilføj IP-baseret tilgang, og gerne login.

mod_autoindex giver mulighed for listevisning, når filer som index.html og index.php ikke findes. Det kan give angribere adgang til vigtig information, hvis brugerne ikke er opmærksomme på listevisningsmulighederne. Derfor bør dette være slået fra globalt, og så blot tilladt de enkelte steder der er behov for det.

mod_userdir giver brugerne mulighed for at oprette sider i deres hjemmemapper, i public_html, som får adresserne domæne.tld/ brugernavn. Dette bør ligeledes være slået fra globalt, og så kun sat på de enkelte vhosts hvor brugerne har behov for det.

1.3.4 Ændr informationsbannere

For at minimere informationslækager, er det smart at ændre de bannere apache kommer med. Som standard sendes banner med apache-version, samt aktive moduler og deres versioner, ud i alle HTTP-responses, og apache-versionen i autogenererede dokumenter, som listevisning og fejlmeldinger. I Sendmail og Bind er det nemt at sætte et banner selv, men det kræver kildekodeændring i apache. Derimod kan ServerTokens sættes til ProductOnly, og ServerSignature sættes til Off i httpd.conf.

1.3.5 Kør CGI-programmer som normale brugere

Normalt kører alle cgi-programmer, som den samme bruger som apache, nemlig www. Det får mange begynderbrugere til at oprette mapper og filer med helt gale permissions, så deres cgi-programmer kan både læse og skrive. Det kan nemt resultere i at een brugers cgi-program, kan skade andre brugeres.

Det vil være smart at lade en brugers cgi-program køre som denne bruger. Det kan man via cgiwrap eller mod_suexec.

cgiwrap

Kan installeres fra www/cgiwrap, og giver filerne cgiwrap og cgiwrapd som begge er setuid root. Disse kan lægges i /usr/local/www/cgi-bin/, og bør være eksekverbare, men ikke læsbare, for alle.

For at køre et script gennem cgiwrap, bruges URL'en domain.tld/cgi-bin/cgiwrap/username/scriptname. cgiwrapd er en debugger-udgave, som giver en masse brugbar information. Denne information er guld værd for en angriber, og derfor bør man kun sætte cgiwrapd op når man selv har behov for den.

mod_suexec

Modulet laver 20 tjek, der primært omhandler hvorvidt cgi-programmet har lov til at køre. Idet modulet er tæt integreret i apache, til forskel fra cgiwrap, har det visse fordele. Det giver to nye direktiver, der kan bruges i vhosts i httpd.conf; User og Group. Disse siger hvilken bruger og gruppe cgi-programmer for den enkelte vhost skal køres som.

Ulempen er at apache skal restartes når der er ændret på konfigurationen, men fordelene er den tættere integration, og det at konfigurationsændringer kan laves af flere administratorer, hvor cgiwrap kræver man kan ændre setuid root-filer

1.4 TLS

Det vi normalt kalder SSL, er i dag ofte TLS. Det er en transportlagsprotokol, lige over HTTP. Det sikrer at trafikken mellem klient og vært er krypteret, og det forsikrer klienten om at værten er den denne udgiver sig for at være.

For at bruge TLS, skal du have en privat nøgle (server.key) og et offentligt certifikat. Certifikatet skal signeres. Det kan købes dyrt hos Verisign, billigt hos InstantSSL, gratis med CAcert, eller ved at signere det selv. Selvsignerede certifikater sikrer ikke at andre har godkendt ægtheden af en. Hvis ikke

klienternes browsere har godkendt ens CA, får brugerne en del advarsler når de tilgår ens side, hvilket måske ikke er smart.

Eksempel på brug af TLS(/SSL) i en vhost i apache:

```
<VirtualHost *:443>
  DocumentRoot /usr/local/www/data/
  SSLEngine On
  SSLCertificateFile /usr/local/etc/apache2/ssl.crt/server.crt
  SSLCertificateKeyFile /usr/local/etc/apache2/ssl.key/server.key
</VirtualHost>
```

Hvis du føler at serveren cpu-forbrug er for højt, når man forbindelser med TLS er igang, er det måske relevant at sætte et kryptoakseleratorkort i serveren. Både Open- og FreeBSD understøtter en stor mængde kort, og det er normalt blot at sætte kortet i, og så kører det.

1.4.1 Udeluk svage algoritmer

Du kan sætte direktivet SSLCipherSuite til et mønster, der afspejler hvilke krypteringsalgoritmer du ikke tillader, foretrækker osv. Hvis du er for hård, risikerer du at gamle versioner af IE ikke kan forbinde til dig, og dette vil ikke endgang blive vist i logfilerne.

Denne kan måske bruges: TLSv1:!ADH:!EXP:!NULL:!MD5:!LOW:+HIGH:+MEDIUM, som siger ja til TLS-ciphers, nej til Diffie-Hellman, exportniveau-algoritmer, NULL-algoritmer og de der i "man ciphers"er markederde som svage algoritmer. Dernæst siger den ja til de der er højtmarkerede, og til sidst ja til mellemmarkerede.

Kapitel 2

IDS

Baseret på kapitel 9 i "Mastering FreeBSD and OpenBSD security".

Intrusion Detection Systems forsvarer ikke aktivt netværket. De rapporterer udelukkende, og derfor skal en administrator manuelt handle på IDS'ens logs.

IDS'er kan tage en del resurser og diskplads, og bør derfor ikke køres på alt for gamle maskiner. Eksempelvis kan en maskine på et par hundrede MHz, med 256mb ram og 10gb disk godt klare en middelstor DSL-linje. Hvis der er tale om en NIDS, vil netværkskort der har hardwarehåndtering af pakker være smart. F.eks. Intels kort med fxp-driveren.

Det er vigtigt at have handleplaner for hvad der skal ske i forskellige situationer. Udarbejd dokumenter, der fortæller hvad der f.eks. skal ske ved portscans, exploitforsøg og lign. Måske er der forskellige regler alt efter om angriberen kommer inde- eller udefra.

Nødvendigheden af IDS er en del lavere end nødvendigheden af alm. sikker konfiguration, firewalls, opdateringer osv. Sørg for at hav de basale ting på plads, og implementer IDS derefter, hvis der er plads til det, både økonomisk og tidsmæssigt administrativt.

2.1 IDS arkitekturer

2.1.1 HIDS

Host based IDS er installeret på den maskine der ønskes IDS på. Den kan f.eks. overvåge systemkald, filer og input fra både netværk og andre inputenheder (terminaler, keyboard osv). Den skal vedligeholdes på hver enkelt maskine, så det kan hurtigt blive en stor administrativ opgave i sig selv, hvis der er tale om et større antal maskiner.

2.1.2 NIDS

Network based IDS er styret fra en enkelt maskine. Denne sniffer så al traffik på netværket, eventuelt ved hjælp af mindre sensorer hvis det er et større netværk. Administrationen er central, men til gengæld holdes der kun øje med netværkstraffik. NIDS-sensorer kan ved hver maskine, lige efter firewallen, eller sammen med en netværks tap (span port, mirror port på switches), der så sender al traffik til NIDS'en. Det er en planlægningsopgave i sig selv.

Sikring af NIDS

En NIDS bør sikres som en firewall, eller bedre. Overfor portskanning bør den være ikkeeksisterende, og man kan sagtens undlade at tildele den en IP-adresse. I små netværk med begrænset hardware, kan den dog måske tillades at indgå på samme maskine som firewall. Et typisk NIDS-angreb, er at lade NIDS'en generere så mange logfiler, at der bliver problemer med diskpladsen. Derfor kan det være en ide at lade NIDS'en have sin egen partition, f.eks. i /var/ids/

2.1.3 Loganalyser og honeypots er ikke IDS

Nogle gange misforståes hvad der er IDS, og hvad der blot er at kigge i logfiler. Programmer som Swatch, der holder øje med logfiler og kan rapportere sikkerhedsproblemer på baggrund af logfiler alene, er ikke en hel IDS. Men derudover er log monitering også vigtigt.

Honeypots er gode til at give en ide om hvordan en angriber forsøger

at trænge ind, men de giver ikke nogen information om angreb på ens reele system.

2.1.4 IPS

Intrusion Prevension System er en lidt udvidet IDS-type, der sidder aktivt i et netværk, ligesom en firewall. Hvis der er mistanke om angreb, kan den lukke af for den mistænkelige trafik. Det kan dog medføre at lovlig trafik hindres ved uheld, og i værste fald kan det misbruges til DOS-angreb.

2.1.5 Fejlrapporteringer

En IDS kan både rapportere forkert positivt og negativt. "False positives" er når IDS'en tror den detekterer et angreb, selv om det ikke sker. Det kan skabe en del forvirring og unødvendigt arbejde. "False negatives" er dog værre, da det vil sige angreb uden IDS'en har detekteret det. Begge dele bør man prøve at undgå, ved at finindstille sin IDS til netværket. Det kan være en god ide selv at lave ægte angreb og angrebslignende traffik, for at teste IDS'en.

2.2 NIDS på BSD

NIDS virker fint på både Open- og FreeBSD. Det er vigtigt at holde sin IDS sikker, da en angriber der opdager en IDS, naturligt vil prøve at sætte denne ude af funktion. OpenBSD vil være oplagt til de fleste mindre IDS'er, men til store net med mange forskellige krav, kan FreeBSD måske være smartere.

2.2.1 Snort

Installation og konfiguration

Snort ligger i ports, og bør være ligetil at installere. Som standard gemmer den logfiler i ren tekst, men den kan bygges med "–with-mysql" for at bruge database til formålet i stedet.

2.2.2 Acid

2.3 HIDS på BSD

2.3.1 Osiris