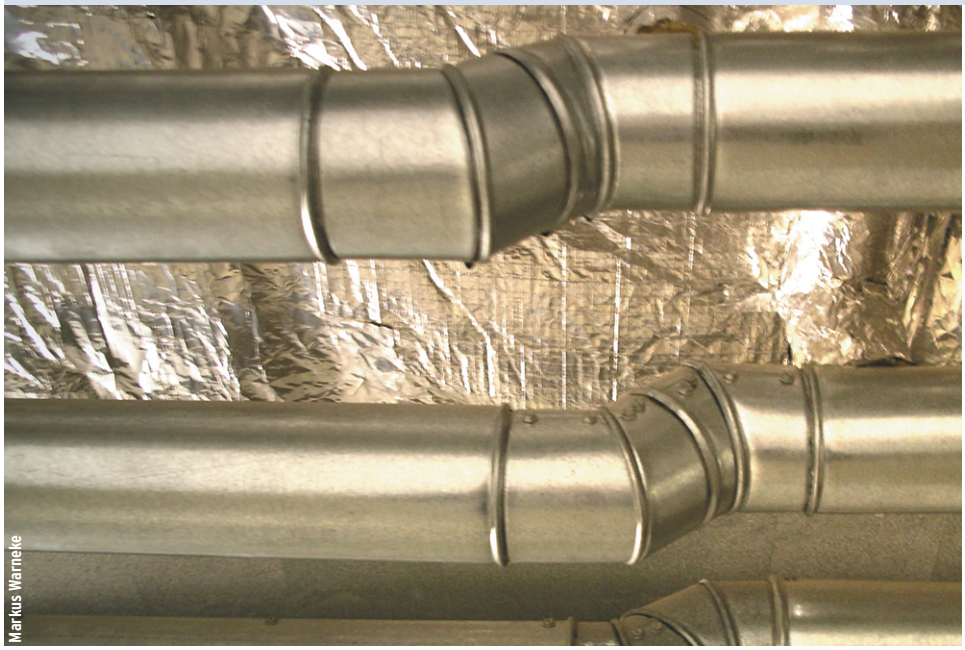
SSH-Zugriff einschränken mit SCPonly

Sichere Leitung

Die meisten Administratoren würden das Uraltprotokoll FTP gerne durch die sichere Alternative SSH mit SCP oder SFTP ersetzen. Diese Programmsuite erlaubt jedoch ohne weitere Maßnahmen komplette Shellzugriffe auf das System. SCPonly schafft Abhilfe und sperrt Benutzer auf Wunsch in eine Chroot-Umgebung. Martin Werthmöller


Markus Warmeke

Für den Up- und Download von Dateien auf einen Server ist das mittlerweile dreißig Jahre alte FTP-Protokoll immer noch De-facto-Standard – obwohl es sehr unsicher ist: Die gesamte Datenübertragung erfolgt unverschlüsselt, Angreifer müssen sich kaum anstrengen, um alle Passwörter mitzulesen.

Außerdem macht die Art, wie FTP arbeitet, die Konfiguration von Firewalls kompliziert [1]. Das Protokoll benötigt bei der Kommunikation nämlich zwei TCP-Verbindungen. Je nach Modus baut entweder der Server oder der Client eine Verbindung zu einem beliebigen, ausgehandelten Zielport bei der Gegenseite auf. Kommt eine derartige Anfrage zum Verbindungsaufbau bei der Firewall an, muss sie wissen, dass es sich um einen FTP-Transfer handelt.

Als eine sichere und kompatible Alternative zu FTP gibt es FTPS, das im RFC 2228 beschrieben ist [2]. FTPS arbeitet genau wie FTP mit zwei unterschiedlichen TCP-Verbindungen und verschlüsselt alle Daten ähnlich wie HTTPS. FTPS ist allerdings nie über das Stadium eines RFC-Entwurfs hinausgekommen und nur sehr wenige Clients und Server implementieren dieses Protokoll.

Alternativen nicht in Sicht

Die HTTP-Protokollerweiterung WebDAV (RFC 2518, [3]) bietet die Möglichkeit, Dateien auf den Server zu laden und dort direkt zu bearbeiten. Wenn die Kommunikation per TLS/SSL erfolgt, ist auch die Verschlüsselung der Daten gesichert. Der Einsatz von WebDAV hat allerdings einige Nachteile: Beim Apache Webserver unterstützt das Modul keinen Wechsel der User-ID. Daher greifen alle WebDAV-Benutzer mit derselben User-ID auf das Dateisystem zu. Mit Apache und WebDAV ist eine Rechteverwaltung also nicht möglich. Zudem ist für die meisten eingesetzten Webserver in der Regel kein WebDAV-Modul installiert oder passend konfiguriert.

Sicherer Ersatz

Als Ersatz zu FTP bieten sich daher bei Unix-Systemen vor allem die im SSH-Paket mitgelieferten Programme SCP (Secure Copy) und SFTP an. Mit SCP lassen sich Dateien verschlüsselt durch einen SSH-Tunnel transportieren. Die zu kopierenden Dateien gibt der Benutzer über die Kommandozeile an. Das Protokoll SFTP ist nicht zu verwechseln mit FTPS, mit dem es nichts zu tun hat. SFTP benötigt nur eine TCP-Verbindung, da es wie SCP über SSH arbeitet. Es lässt sich allerdings im Gegensatz zu SCP wie ein FTP-Client interaktiv bedienen.

Diese beiden Varianten erfordern auf Server- und auf Client-Seite keine umfangreiche Konfiguration. Ein SSH-Server ist auf vielen Rechnern bereits standardmäßig eingerichtet und die meisten Firewalls lassen SSH-Verbindungen zu. Leider gibt es aber auch bei diesen sicheren Lösungen einige Nachteile:

- Nicht auf allen Betriebssystemen sind grafische SCP-Clientprogramme installiert.
- Die SCP- und SFTP-Programme in den (Open)SSH-Paketen sind für viele Anwender zu unkomfortabel.
- SCP setzt einen interaktiven SSH-Zugang voraus. Die SCP-Benutzer sind daher auch in der Lage, sämtliche Shellbefehle auszuführen.
- Die Möglichkeit einer Chroot-Konfiguration, wie sie einige FTP-Server bieten, beherrscht der SSH-Daemon nicht nativ.

Die Entwicklung von OpenSSH (siehe Artikelserie in [4]) findet primär nur für OpenBSD statt, doch es gibt Portierungen auf viele Systeme, darunter Linux, Cygwin und Mac OS X. Dank des Cygwin-Ports lässt sich OpenSSH auch unter Windows installieren. Für Windows-Clients eignet sich jedoch Putty besser [5]. Neben den OpenSSH-Client-Programmen »scp« und »sftp« gibt es für Linux, Windows und Mac OS X auch grafische Frontends (siehe Kasten „Grafische SCP-Clients“). Die Lizenzen dieser Clients erlauben oftmals eine kostenlose Nutzung.

Es gibt viele Umgebungen, in denen Nutzer keine Login-Shell bekommen sollten. Bei großen Webspace-Anbietern liegen zum Beispiel Tausende von Webseiten auf einem einzigen Rechner. Voller Shellzugriff könnte fatale Auswirkungen haben, ein solches System ist zudem komplex und aufwändig zu warten.

Zugang einschränken

Da SCP und SFTP eine interaktive Shell voraussetzen, muss die gesuchte Lösung den Shellzugriff einschränken, aber die Funktionalität von SCP und SFTP gewährleisten. Dazu gibt es zwei mögliche Ansätze: Eine angepasste Login-Shell erlaubt es dem Benutzer lediglich, Befehle auszuführen, die für SCP und SFTP nö-

Grafische SCP-Clients

Für SSH unter Linux, Windows und Mac OS X gibt es mehrere grafische Clients. Einige dieser Programme wie Gftp [6] (Abbildung 1) sind im Grunde FTP-Clients, die auch den Zugriff mittels SCP und SFTP beherrschen. Andere wiederum sind reine SCP-Programme, etwa WinSCP [7] für Microsoft Windows. Die Oberflächen unterscheiden sich kaum voneinander. Alle Applikationen zeigen ein zweigeteiltes Fenster (eine Seite für den lokalen und eine für den entfernten Verzeichnisbaum), in dem sich Dateien und Ordner per Drag & Drop hin und her kopieren lassen.

Die Datei-Attribute stellt der Anwender in den meisten Fällen mit einem Klick auf die rechte Maustaste ein. Zusätzlich ist oft noch eine Bookmark- und Passwortverwaltung integriert. Benutzer von WinSCP sollten die aktuelle Version 3 nutzen, da dort etliche Fehler der Vorversion behoben sind. Fehlermeldungen aufgrund eines nicht funktionierenden »groups«-Kommandos auf dem Server lassen sich vermeiden, wenn die Option »lookup user groups« in WinSCP deaktiviert ist. Als Alternative zu WinSCP ist noch der Windows-Client Filezilla [8] zu empfehlen.

KDE-Benutzer haben eine sehr komfortable Möglichkeit, um SCP zu benutzen. Sie geben einfach »fish://*Server*« in die Eingabezeile des Konqueror ein und kopieren Dateien, als lägen sie auf der lokalen Festplatte. Unter Mac OS X gibt es das Programm Fugu [9]. Es ist im Quellcode verfügbar und unterstützt FTP, SFTP sowie SCP.

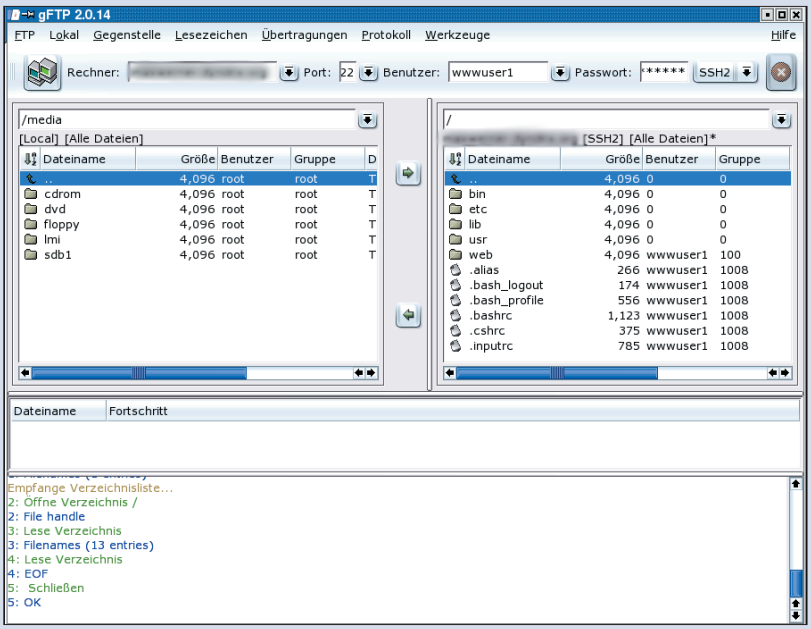


Abbildung 1: Eine SCP-Session mit dem grafischen Programm Gftp. Da auf dem entfernten Rechner SCPonly läuft, befindet sich der Benutzer in einer Chroot-Umgebung.

tig sind. Die zweite Möglichkeit ist, die Anwender beim Einloggen in eine Chroot-Umgebung einzusperren und nur die notwendigen Programme dort bereitzustellen. So hat der Client keine Möglichkeit, auf andere Dateien zuzugreifen als auf seine eigenen.

Ein Wechsel in die Chroot-Umgebung durch die Login-Shell birgt die Gefahr, dass der Benutzer den Vorgang vor dem Chroot unterbricht und Befehle in der ungesicherten Umgebung absetzt. Die Login-Shell muss daher sicherstellen, dass der Chroot-Aufruf erfolgt, bevor der Benutzer in der Lage ist, Eingaben zu tätigen. Daher gibt es unter **[10]** ein Patch, das direkt den SSH-Daemon veranlasst »chroot()« aufzurufen.

Beste Lösung SCPonly

Ein derart gepatchter SSH-Daemon ist aber leider nicht als RPM- oder Deb-Paket vom Distributor verfügbar, sodass sich der Administrator von Sicherheits-Updates der Distributoren abkoppelt. Er muss die OpenSSH-Installation auf dem Server bei jedem Sicherheits-Update manuell nachpflegen.

Die beiden Programme RSSH **[11]** und SCPonly **[12]** vereinen beide Möglichkeiten: Sie sperren den Benutzer in eine Chroot-Umgebung und erlauben ihm nur die nötigsten Befehle aufzurufen. Ein Vorteil von SCPonly gegenüber RSSH ist die Kompatibilität zum weit verbreiteten Client WinSCP. Außerdem lässt sich mit SCPonly Rsync über SSH ausführen und in zukünftigen Versionen soll zusätzlich der Support von CVS über SSH integriert sein.

Gegenwärtig liefert noch keine der gängigen Linux-Distributionen SCPonly-Pakete mit. Die Debian-Entwickler haben aber bereits ein Paket in den Unstable- und Testing-Zweig aufgenommen. Bis dahin müssen Benutzer aller Distributionen das Programm aus dem Quellcode installieren, er steht unter **[8]** zum Download bereit.

Der Autor von SCPonly entwickelt und nutzt das Programmpaket unter FreeBSD, das sich unter anderem bei den Programmen zur Benutzer- und Passwortverwaltung von Linux unterscheidet. In älteren Versionen erforderte daher insbesondere die automatische Konfiguration eines Chroot-Käfigs oft manuelles Nacharbeiten an den Setup-Skripten. In der aktuellen Version 3.11 ist das aber nicht mehr nötig.

Eine weitere zu beachtende Falle ist das »groups«-Kommando, das unter Linux häufig als Shellskript implementiert ist. Das Setup-Skript »setup_chroot.sh« kopiert unter anderem die Datei »/usr/bin/groups« in die Chroot-Umgebung. Beim Aufruf von »groups« versucht der Kernel dann, das Skript an die Shell »/bin/sh« zu übergeben.

Wenn der Administrator keinen vollständigen Kommandozeilen-Interpreter im Käfig installieren will, bleibt ihm nur die Möglichkeit, »/usr/bin/groups« in der Chroot-Umgebung durch das von SCPonly mitgelieferte Kommando zu ersetzen. Einige grafische SCP-Clients bereiten dabei Probleme, die sich mit dem im Kasten „Grafische SCP-Clients“ beschriebenen Tipp beheben lassen.

Installation

Nach dem Entpacken der SCPonly-Sourcen sind mit dem »configure«-Skript einige Einstellungen vorzunehmen, etwa Rsync- oder Chroot-Funktionalität. Eine vollständige Liste der Optionen gibt »./configure --help« aus. Die folgenden Beispielaufrufe aktivieren Rsync-Kompatibilität, erstellen ein Chroot-Binary und installieren alles Nötige:

```
./configure --enable-rsync-compat ➥
    --enable-chrooted-binary
make
make install
```

Um eine Chroot-Umgebung einzurichten, müsste der Administrator jetzt alle erforderlichen Bibliotheken sammeln und an die korrekte Position im Chroot-Verzeichnisbaum kopieren. Diese Aufgabe übernimmt bei SCPonly das angesprochene »setup_chroot.sh«. Es ist standardmäßig nicht mit dem Execute-Bit versehen, die Eingabe von »make jail« statt »make install« schafft Abhilfe und kümmert sich um die nötigen Schritte zur Ausführung des Skripts.

Das Setup-Skript muss der Admin mit Root-Rechten ausführen, daher sollte er bereits »configure« als Root starten, da es die Datei »setup_chroot.sh« aus »setup_chroot.sh.in« erzeugt und einige Dateien in privilegierten Pfaden wie »/usr/sbin/« benötigt. Ist »setup_chroot.sh« erzeugt und gestartet, fragt es nach dem Homeverzeichnis und dem Benutzernamen des neuen SCPonly-Benutzers. Nach dem Login per SFTP befindet sich dieser Benutzer nun in einer Chroot-Umgebung. Er hat ausschließlich in »~/incoming« (oder einem anderen vom Admin festgelegten Verzeichnis) Schreibrechte.

Komfortable Handhabung

Um den Benutzern die Handhabung zu erleichtern, ist es möglich, für das Homeverzeichnis der SCPonly-Benutzer direkt »/home/*Benutzer*/incoming« anzugeben. Mit dieser Einstellung wechselt SCPonly nach dem Login in das Verzeichnis »~/incoming«.

Mit SCPonly steht Server-Anbietern ein Werkzeug zur Verfügung, um das veraltete FTP-Protokoll von ihren Rechnern zu verbannen und auf moderne Authentifizierung und Verschlüsselung umzusteigen. *(mwe)* ■

Infos

[1] Frank Bernard, „Lebendes Relikt“: Linux-Magazin 06/02, S. 54
[2] RFC zu FTPS: [http://www.ietf.org/rfc/rfc2228.txt]
[3] RFC zu WebDAV: [http://www.ietf.org/rfc/rfc2518.txt]
[4] Karl-Heinz Haag und Achim Leitner, Artikelserie zu OpenSSH: Linux-Magazin 05/02, 07/02, 09/02
[5] Putty: [http://www.chiark.greenend.org.uk/~sgtatham/putty/]
[6] Gftp: [http://www.gftp.org]
[7] WinSCP: [http://winscp.sourceforge.net]
[8] Filezilla: [http://filezilla.sourceforge.net]
[9] Fugu: [http://rsug.itd.umich.edu/software/fugu]
[10] OpenSSH-Chroot-Patch: [http://chrootssh.sourceforge.net]
[11] RSSH: [http://www.pizzashack.org/rssh/index.shtml]
[12] SCPonly: [http://sublimation.org/scponly/]

Der Autor

Martin Werthmöller lebt mit seiner Frau und den beiden Söhnen in der Nähe von Münster und Osnabrück. Er ist freiberuflich im Bereich Web-Engineering und Netzwerkadministration tätig.