7.5. Paketsignierung in Debian

Dieser Abschnitt könnte auch mit »Wie man sein Debian GNU/Linux-System sicher upgraded/aktualisiert« überschrieben werden. Es verdient hauptsächlich deshalb einen eigenen Abschnitt, weil es einen wichtigen Teil der Infrastruktur der Sicherheit darstellt. Die Signierung von Paketen ist ein wichtiges Thema, da es die Manipulation von Paketen in Spiegel und von heruntergeladenen Dateien durch Man-in-the-Middle-Angriffen verhindert. Die automatische Aktualisierung von Software ist eine wichtige Fähigkeit, aber es ist auch wichtig, Gefahren für die Sicherheit zu entfernen, die die Verbreitung von Trojanern und den Einbruch ins System während der Aktualisierung fördern können. [55]

FIXME: probably the Internet Explorer vulnerability handling. certificate chains has an impact on security updates on Microsoft Windows.

Debian stellt keine signierten Pakete zur Verfügung. Es gibt aber seit Debian 4.0 (Codename *Etch*) eine Verfahrensweise, mit der die Integrität von heruntergeladenen Paketen überprüft werden kann. [56] Weiterführende Hinweise können Sie unter Abschnitt 7.5.2, „Secure Apt“ finden.

Dieses Problem wird besser im http://www.cryptnet.net/fdp/crypto/strong_distro.html von V. Alex Brennen beschrieben.

7.5.1. Die aktuelle Methode zur Prüfung von Paketsignaturen

Die aktuelle Methode zur Prüfung von Paketsignaturen mit **apt** ist:

- Die **Release**-Datei enthält die MD5-Summe von **Packages.gz** (welche die MD5-Summen der Pakete enthält) und wird signiert. Die Signatur stammt aus einer vertrauenswürdigen Quelle.
- Diese signierte **Release**-Datei wird beim »apt-get update« herunter geladen und zusammen mit **Packages.gz** gespeichert.
- Wenn ein Paket installiert werden soll, wird es zuerst herunter geladen, und dann wird die MD5-Summe erstellt.
- Die signierte **Release**-Datei wird überprüft (ob die Signatur in Ordnung ist) und die MD5-Summe der **Packages.gz**-Datei extrahiert. Die MD5-Summe der **Packages.gz**-Datei wird erstellt und geprüft, und – wenn sie übereinstimmt – wird die MD5-Summe des heruntergeladenen Paketes aus ihr extrahiert.
- Wenn die MD5-Summe des heruntergeladenen Paketes die gleiche ist wie in der **Packages.gz**-Datei, wird das Paket installiert. Andernfalls wird der Administrator alarmiert und das Paket wird im Zwischenspeicher gehalten (so dass der Administrator entscheiden kann, ob es installiert werden soll oder nicht). Wenn das Paket nicht in **Packages.gz** enthalten ist und der Administrator das System so konfiguriert hat, dass nur geprüfte Pakete installiert werden können, wird das Paket ebenfalls nicht installiert.

Durch diese Kette von MD5-Summen ist **apt** in der Lage, zu verifizieren, dass ein Paket aus einer bestimmten Veröffentlichung stammt. Dies ist zwar unflexibler als jedes Paket einzeln zu signieren, kann aber auch mit den unten aufgeführten Plänen kombiniert werden.

Diese Vorgehensweise ist seit der Veröffentlichung von Debian 4.0 verfügbar und vollständig in apt 0.6 http://lists.debian.org/debian-devel/2003/debian-devel-200312/msg01986.html; weitere Informationen finden Sie unter Abschnitt 7.5.2, „Secure Apt“. Pakete, die ein Frontend für apt anbieten, müssen verändert werden, um an diese neue Fähigkeit angepasst zu werden. Das gilt für **aptitude**, das http://lists.debian.org/debian-devel/2005/03/msg02641.html wurde, um zu dieser Vorgehensweise zu passen. Frontends, die bekanntermaßen zurzeit mit dieser Fähigkeit umgehen können, sind **aptitude** und **synaptic**.

Die Signierung von Paketen wurde innerhalb des Debian-Projekts ausführlich diskutiert. Mehr Informationen hierzu finden Sie unter http://www.debian.org/News/weekly/2001/8/ und http://www.debian.org/News/weekly/2000/11/.

7.5.2. Secure Apt

Die Veröffentlichung von apt 0.6, das seit Debian 4.0 (*Etch*) verfügbar ist, enthält *apt-secure* (auch als *Secure Apt* bekannt), das ein Werkzeug ist, mit dem ein Systemadministrator die Integrität von heruntergeladenen Paketen mit dem oben dargestellten Verfahren überprüfen kann. Diese Veröffentlichung enthält das Werkzeug **apt-key**, um neue Schlüssel zum Schlüsselbund von apt hinzuzufügen, welcher standardmäßig nur den aktuellen Signierungsschlüssel des Debian-Archivs enthält.

Diese Veränderungen basieren auf dem Patch für **apt** (verfügbar in http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=203741), der diese Erweiterung zur Verfügung stellt.

Secure Apt überprüft die Distribution mit der **Release**-Datei. Dies wurde schon unter Abschnitt 7.5.3, „Überprüfung der Distribution mit der **Release**-Datei“ dargestellt. Typischerweise erfordert dieser Vorgang kein Mitwirken des Administrators. Aber jedes Jahr müssen Sie eingreifen [57], um den neuen Schlüssel des Archivs hinzuzufügen, wenn dieser ausgewechselt wurde. Weitere Informationen zu den dazu notwendigen Schritten finden Sie unter Abschnitt 7.5.3.7, „Auf sichere Weise einen Schlüssel hinzufügen“.

Diese Fähigkeit befindet sich noch im Entwicklungsstadium. Wenn Sie glauben, dass Sie Fehler gefunden haben, stellen Sie zuerst sicher, dass Sie die neuste Version verwenden (da dieses Paket vor seiner endgültigen Veröffentlichung noch ziemlich verändern werden kann). Falls Sie die aktuelle Version benutzen, schicken Sie einen Fehlerbericht für das Paket **apt**.

Weiterführende Informationen finden Sie im http://wiki.debian.org/SecureApt und in der offiziellen Dokumentation unter http://www.enyo.de/fw/software/apt-secure/ und http://www.syntaxpolice.org/apt-secure/.

7.5.3. Überprüfung der Distribution mit der Release-Datei

Dieser Abschnitt beschreibt, wie die Überprüfung der Distribution mit Hilfe der **Release**-Datei funktioniert. Dies wurde von Joey Hess geschrieben und ist auch im http://wiki.debian.org/SecureApt abrufbar.

7.5.3.1. Grundlegende Konzepte

Es gibt ein paar grundlegende Konzepte, die Sie brauchen, um den Rest dieses Abschnitts verstehen zu können.

Eine Prüfsumme ist eine Methode, bei der eine Datei auf eine relativ kurze Zahl heruntergekocht wird, mit welcher der Inhalt der Datei eindeutig identifiziert werden kann. Dies ist wesentlich schwieriger, als es zunächst erscheinen mag. Der am weitesten verbreiteteste Typ von Prüfsummen, MD5, wird gerade unbrauchbar.

Verschlüsselung mit öffentlichen Schlüsseln fußt auf einem Schlüsselpaar: einem öffentlichen Schlüssel und einem privaten Schlüssel. Der öffentliche Schlüssel wird an die Allgemeinheit verteilt. Der private muss geheim bleiben. Jeder, der den öffentlichen Schlüssel hat, kann eine Nachricht verschlüsseln, so dass sie nur noch der Besitzer des privaten Schlüssels lesen kann. Es besteht daneben die Möglichkeit, mit einem privaten Schlüssel eine Datei zu

signieren. Wenn eine Datei mit einer digitalen Unterschrift versehen wurde, kann jeder, der den öffentlichen Schlüssel hat, überprüfen, ob die Datei mit diesem Schlüssel unterschrieben wurde. Ohne den privaten Schlüssel lässt sich eine solche Signatur nicht nachmachen.

Diese Schlüssel bestehen aus ziemlich langen Zahlen (1024 oder 2048 Ziffern oder sogar länger). Damit sie leichter zu verwenden sind, haben sie eine kürzere Schlüssel-ID (eine Zahl mit nur acht oder 16 Stellen), mit der sie bezeichnet werden können.

Secure Apt verwendet **gpg**, um Dateien zu unterschreiben und ihre Signaturen zu überprüfen.

Mit dem Programm **apt-key** wird der Schlüsselbund von GPG für Secure Apt verwaltet. Der Schlüsselbund befindet sich in der Datei **/etc/apt/trusted.gpg** (nicht zu verwechseln mit der verwandten, aber nicht sehr interessanten Datei **/etc/apt/trustdb.gpg**). **apt-key** kann dazu verwendet werden, die Schlüssel im Schlüsselbund anzuzeigen oder um Schlüssel hinzuzufügen oder zu entfernen.

7.5.3.2. Prüfsummen der Release-Datei

Jedes Archiv von Debian enthält eine **Release**-Datei, die jedes Mal aktualisiert wird, wenn ein Paket im Archiv geändert wird. Unter anderem enthält die **Release**-Datei MD5-Summen von anderen Dateien, die sich im Archiv befinden. Ein Auszug einer **Release**-Datei:

```
MD5Sum:
 6b05b392f792ba5a436d590c129de21f            3453 Packages
 1356479a23edda7a69f24eb8d6f4a14b            1131 Packages.gz
 2a5167881adc9ad1a8864f281b1eb959            1715 Sources
 88de3533bf6e054d1799f8e49b6aed8b             658 Sources.gz
```

Die **Release**-Datei enthält auch SHA1-Prüfsummen, was nützlich wird, wenn MD5-Summen vollständig unbrauchbar sind. Allerdings unterstützt apt SHA1 noch nicht.

Werfen wir einen Blick in eine **Packages**-Datei: Wir sehen weitere MD5-Summen, eine für jedes darin aufgeführte Paket. Beispiel:

```
    Package: uqm
    Priority: optional
    ...
    Filename: unstable/uqm_0.4.0-1_i386.deb
    Size: 580558
    MD5sum: 864ec6157c1eea88acfef44d0f34d219
```

Mit diesen beiden Prüfsummen kann überprüft werden, ob Sie eine getreue Kopie der **Packages**-Datei heruntergeladen haben, also mit einer MD5-Summe, die mit der in der **Release**-Datei übereinstimmt. Und wenn ein einzelnes Paket heruntergeladen wird, kann auch die MD5-Summe mit dem Inhalt der **Packages**-Datei verglichen werden. Wenn bei einem dieser Schritte ein Fehler auftauchen sollte, bricht Apt den Vorgang ab.

Nichts davon ist neu in Secure Apt, sondern bietet nur die Grundlage für Secure Apt. Beachten Sie, dass es bis jetzt eine Datei gibt, die Apt nicht überprüfen kann: die **Release**-Datei. Bei Secure Apt dreht sich alles darum, dass Apt die **Release**-Datei überprüft, bevor es irgendetwas anderes damit macht. Wenn man das schafft, besteht eine lückenlose Authentifizierungskette von dem Paket, das Sie installieren möchten, bis zum Anbieter des Pakets.

7.5.3.3. Überprüfung der Release-Datei

Damit die **Release**-Datei überprüft werden kann, wird sie mit GPG signiert. Diese Unterschrift kommt in die Datei **Release.gpg**, die mit der **Release**-Datei abgerufen werden kann. Sie sieht in etwa so [58] aus, obwohl sich für gewöhnlich nur GPG ihren Inhalt ansieht:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.1 (GNU/Linux)

iD8DBQBCqKO1nukh8wJbxY8RAsfHAJ9hu8oGNRAl2MSmP5+z2RZb6FJ8kACfWvEx
UBGPVc7jbHHsg78EhMBlV/U=
=x6og
-----END PGP SIGNATURE-----
```

7.5.3.4. Release.gpg mit Apt überprüfen

Wenn Secure Apt eine **Release**-Datei herunterlädt, lädt es immer auch die **Release.gpg**-Datei herunter. Falls dies misslingen sollte oder die Signatur nicht stimmt, wird es eine Rückmeldung machen und hinweisen, dass die **Packages**-Dateien, auf welche die **Release**-Datei verweist, und alle darin enthaltenen Pakete von einer nicht vertrauenswürdigen Quelle stammen. So würde dies während **apt-get update** aussehen:

```
W: GPG error: http://ftp.us.debian.org testing Release: The following signatures
 couldn't be verified because the public key is not available: NO_PUBKEY 010908312D230C5F
```

Beachten Sie, dass die zweite Hälfte der langen Nummer die Schlüssel-ID des Schlüssels ist, von dem Apt nichts weiß. Im Beispiel ist sie 2D230C5F.

Falls Sie diese Warnung ignorieren und später versuchen, ein Paket zu installieren, wird Sie Apt nochmals warnen:

```
WARNUNG: Die folgenden Pakete können nicht authentifiziert werden!
  libglib-perl libgtk2-perl
Diese Pakete ohne Überprüfung installieren [j/N]?
```

Wenn Sie nun »J« drücken, haben Sie keine Möglichkeit festzustellen, ob die Datei, die Sie bekommen, wirklich diejenige ist, die Sie auch installieren möchten, oder ob sie eine ganz andere ist, die Ihnen jemand, der die Verbindung mit dem Server abgefangen hat [59], mit einer gemeinen Überraschung unterschieben will.

Hinweis: Sie können diese Abfragen abschalten, indem Sie **apt** mit **--allow-unauthenticated** laufen lassen.

Es lohnt sich auch noch darauf hinzuweisen, dass der Installer von Debian während des Debootstraps des Basissystems, solange Apt noch nicht verfügbar ist, denselben Mechanismus mit signierten **Release**-Dateien verwendet. Der Installer benutzt sogar dieses Verfahren, um Teile von sich selbst zu überprüfen, die er aus dem Netz gezogen hat. Debian signiert im Moment nicht die **Release**-Dateien auf den CDs. Apt kann aber so eingerichtet werden, dass es immer den Paketen von CDs vertraut, so dass dies nicht ein so großes Problem darstellt.

7.5.3.5. Wie man Apt sagt, wem es vertrauen soll

Die ganze Sicherheit des Verfahrens beruht also darauf, dass es eine **Release.gpg**-Datei gibt, die eine **Release**-Datei signiert, und dass diese Signatur von **apt** mit Hilfe von GPG überprüft wird. Dazu muss es den öffentlichen Schlüssel der Person kennen, welche die Datei unterschrieben hat. Diese Schlüssel werden in Apts eigenem Schlüsselbund (**/etc/apt/trusted.gpg**) gespeichert. Bei der Verwaltung dieser Schlüssel kommt Secure Apt ins Spiel.

Standardmäßig befindet sich bei Debian-Systemen der Schlüssel des Debian-Archivs im Schlüsselbund.

```
# apt-key list
/etc/apt/trusted.gpg
--------------------
pub   1024D/4F368D5D 2005-01-31 [expires: 2006-01-31]
uid                  Debian Archive Automatic Signing Key (2005) <ftpmaster@debian.org>
```

Im Beispiel ist 4F368D5D die Schlüssel-ID. Beachten Sie, dass dieser Schlüssel nur für ein Jahr gültig ist. Debian tauscht die Schlüssel als letzte Verteidigungslinie gegen Sicherheitsrisiken, die das Knacken eines Schlüssels umfassen, regelmäßig aus.

Mit dem Schlüssel des Archivs wird **apt** dem offiziellen Archiv von Debian vertrauen. Wenn Sie aber weitere Paketdepots zu **/etc/apt/sources.list** hinzufügen wollen, müssen Sie Apt Ihre Schlüssel mitteilen, wenn Sie wollen, dass Apt ihnen vertraut. Sobald Sie den Schlüssel haben und ihn überprüft haben, müssen Sie nur **apt-key add** *Datei* laufen lassen, um den Schlüssel hinzuzufügen. Der schwierigste Teil dabei ist, den Schlüssel zu bekommen und ihn zu überprüfen.

7.5.3.6. Den Schlüssel eines Paketdepots finden

Mit dem Paket **debian-archive-keyring** werden Schlüssel für **apt** bereitgestellt. Aktualisierungen dieses Pakets führen dazu, dass GPG-Schlüssel für das Debian-Hauptarchiv hinzugefügt (oder gelöscht) werden.

Für andere Archive gibt noch keinen standardisierten Ort, wo sich der Schlüssel für ein Paketdepot befinden soll. Es besteht die grobe Übereinkunft, dass der Schlüssel auf der Webseite des Paketdepots oder im Depot selbst zu finden sein sollte. Wie gesagt ist dies kein echter Standard, so dass Sie den Schlüssel unter Umständen suchen müssen.

Der Schlüssel des Debian-Archivs ist unter http://ftp-master.debian.org/ziyi_key_2006.asc (ersetzen Sie 2006 mit dem aktuellen Jahr) erhältlich. [60]

gpg besitzt mit den Schlüsselservern eine standardisierte Möglichkeit, Schlüssel zu verbreiten. Damit kann GPG einen Schlüssel herunterladen und ihn zum Schlüsselbund hinzufügen. Beispiel:

```
$ gpg --keyserver pgpkeys.mit.edu --recv-key 2D230C5F
gpg: requesting key 2D230C5F from hkp server pgpkeys.mit.edu
gpg: key 2D230C5F: public key "Debian Archive Automatic Signing Key (2006) <ftpm
aster@debian.org>" imported
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:               importiert: 1
```

Sie können dann den Schlüssel aus Ihrem Schlüsselbund exportieren und ihn an **apt-key** weiterreichen:

```
$ gpg -a --export 2D230C5F | sudo apt-key add -
gpg: kein uneingeschränkt vertrauenswürdiger Schlüssel 080F67F4 gefunden
OK
```

Die Warnung »gpg: kein uneingeschränkt vertrauenswürdiger Schlüssel 080F67F4 gefunden« bedeutet, dass GPG nicht so konfiguriert wurde, um einem Schlüssel vollständig zu vertrauen. Das Zuweisen von Vertrauensstufen ist Teil des Web-of-Trust von OpenPGP, was hier nicht Gegenstand ist. Daher ist die Warnung unproblematisch. Für gewöhnlich wird dem eignen Schlüssel eines Benutzers vollständig vertraut.

7.5.3.7. Auf sichere Weise einen Schlüssel hinzufügen

Indem Sie einen Schlüssel zu Apts Schlüsselbund hinzufügen, lassen Sie Apt wissen, dass es allem vertrauen soll, was mit diesem Schlüssel signiert wurde. Dadurch stellen Sie sicher, dass Apt nichts installiert, was nicht vom Inhaber des privaten Schlüssels signiert wurde. Mit ausreichender Paranoia erkennen Sie aber, dass dies das Problem nur um eine Stufe verlagert: Anstatt sich nun darum Sorgen zu machen, ob ein Paket oder eine **Release**-Datei korrekt ist, müssen Sie überprüfen, ob Sie tatsächlich den richtigen Schlüssel haben. Ist die Datei http://ftp-master.debian.org/ziyi_key_2006.asc, die oben erwähnt wird, wirklich der Signierungsschlüssel des Debian-Archivs oder wurde sie verändert (oder wird gar in diesem Dokument gelogen)?

Es ist gut, in Sicherheitsfragen Vorsicht walten zu lassen. Aber ab hier wird es schwieriger, Dinge zu überprüfen. **gpg** arbeitet mit dem Konzept der Kette des Vertrauens (chain of trust), die bei jemandem beginnt, dem Sie vertrauen und der einen anderen Schlüssel unterschreibt usw., bis Sie beim Schlüssel des Archivs sind. Wenn Sie vorsichtig sind, wollen Sie nachprüfen, dass Ihr Archivschlüssel von einem Schlüssel unterschrieben wurde, dem Sie vertrauen können, weil seine Kette des Vertrauens zu jemandem zurückgeht, den Sie persönlich kennen. Dazu sollten Sie eine Debian-Konferenz oder eine lokale LUG zum Unterschreiben der Schlüssel besuchen [61].

Wenn Sie diese Sicherheitsbedenken nicht teilen (können), unternehmen Sie, was auch immer Sie passend finden, wenn Sie eine neue Apt-Quelle oder einen neuen Schlüssel verwenden. Sie könnten demjenigen, der den Schlüssel anbietet, eine Mail schreiben, um den Schlüssel zu überprüfen. Oder Sie vertrauen auf Ihr Glück und gehen davon aus, dass Sie den richten heruntergeladen haben. Das wichtige ist, dass Secure Apt, indem es das Problem darauf reduziert, welchen Archivschlüsseln Sie vertrauen, Sie so vorsichtig und sicher vorgehen lässt, wie es Ihnen passend und notwendig erscheint.

7.5.3.8. Die Integrität eines Schlüssels überprüfen

Sie können dazu sowohl den Fingerabdruck als auch die Unterschriften des Schlüssels überprüfen. Den Fingerabdruck kann man aus verschiedenen Quellen erhalten. Sie können im Buch http://debiansystem.info/readers/changes/547-ziyi-key-2006 nachsehen, im IRC mit Debian-Entwicklern reden oder Mailinglisten lesen, wo ein Wechsel des Schlüssel angekündigt werden wird, oder jede andere erdenkliche Methode verwenden, um den Fingerabdruck zu überprüfen. Zum Beispiel können Sie auch Folgendes machen:

```
$ GET http://ftp-master.debian.org/ziyi_key_2006.asc | gpg --import
gpg: key 2D230C5F: public key "Debian Archive Automatic Signing Key (2006)
  <ftpmaster&debian.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
$ gpg --check-sigs --fingerprint 2D230C5F
pub   1024D/2D230C5F 2006-01-03 [expires: 2007-02-07]
      Key fingerprint = 0847 50FC 01A6 D388 A643  D869 0109 0831 2D23 0C5F
uid   Debian Archive Automatic Signing Key (2006) <ftpmaster@debian.org>
sig!3        2D230C5F 2006-01-03  Debian Archive Automatic Signing Key
                                  (2006) <ftpmaster@debian.org>
sig!         2A4E3EAA 2006-01-03  Anthony Towns <aj@azure.humbug.org.au>
sig!         4F368D5D 2006-01-03  Debian Archive Automatic Signing Key
                                  (2005) <ftpmaster@debian.org>
sig!         29982E5A 2006-01-04  Steve Langasek <vorlon@dodds.net>
sig!         FD6645AB 2006-01-04  Ryan Murray <rmurray@cyberhqz.com>
sig!         AB2A91F5 2006-01-04  James Troup <james@nocrew.org>
```

und dann von Ihrem Schlüssel (oder einem Schlüssel, dem Sie vertrauen) den http://www.de.debian.org/doc/manuals/securing-debian-howto/ch7#s-deb-pack-sign zu wenigstens einem der Schlüssel, der verwendet wurde, um den Archivschlüssel zu unterschreiben, überprüfen. Wenn Sie vorsichtig sein wollen, sollten Sie Apt mitteilen, dass es dem Schlüssel nur vertrauen darf, wenn es einen passenden Pfad gefunden hat:

```
$ gpg --export -a 2D230C5F | sudo apt-key add -
Ok
```

Hinweis: Der aktuelle Schlüssel ist mit dem vorhergehenden Archivschlüssel unterschrieben, so dass Sie theoretisch auf Ihrem alten Vertrauen aufbauen können.

7.5.3.9. Der jährliche Austausch des Archivschlüssels von Debian

Wie schon erwähnt wird der Schlüssel, mit dem das Debian-Archiv signiert wird, jedes Jahr im Januar ausgetauscht. Da Secure Apt noch jung ist, haben wir noch nicht sehr viel Erfahrung damit und es gibt noch ein paar haarige Stellen.

Im Januar 2006 wurde ein neuer Schlüssel für 2006 erstellt und die **Release**-Datei wurde damit unterschrieben. Um aber zu vermeiden, dass Systeme, die noch den alten Schlüssel von 2005 verwenden, nicht mehr korrekt arbeiten, wurde die **Release**-Datei auch mit dem alten Schlüssel unterschrieben. Es war geplant, dass Apt je nach dem verfügbaren Schlüssel eine der beiden Unterschriften akzeptieren würde. Aber es zeigte sich ein Fehler in Apt, da es sich weigerte, der Datei zu vertrauen, wenn es nicht beide Schlüssel hatte und somit beide Unterschriften überprüfen konnte. Dies wurde in der Version 0.6.43.1 ausgebessert. Es gab auch Verwirrung darüber, wie der Schlüssel an Benutzer verteilt wird, die bereits Secure Apt auf ihrem System laufen lassen. Am Anfang wurde er auf die Webseite hochgeladen, ohne Ankündigung und ohne eine echte Möglichkeit, ihn zu überprüfen, und die Benutzer mussten ihn per Hand herunterladen.

7.5.3.10. Bekannte Probleme bei der Prüfung der Release-Datei

Ein nicht offensichtliches Problem ist, dass Secure Apt nicht funktioniert, wenn Ihre Uhr sehr verstellt ist. Wenn sie auf ein Datum in der Vergangenheit wie 1999 eingestellt ist, wird Apt mit einer nichts sagenden Ausgabe wie dieser abbrechen:

```
W: GPG error: http://archive.progeny.com sid Release: Unknown error executing gpg
```

Dagegen macht **apt-key** das Problem deutlich:

```
gpg: key 2D230C5F was created 192324901 seconds in the future (time warp or clock problem)
gpg: key 2D230C5F was created 192324901 seconds in the future (time warp or clock problem)
pub   1024D/2D230C5F 2006-01-03
uid                  Debian Archive Automatic Signing Key (2006) <ftpmaster@debian.org>
```

Falls die Uhr nicht zu weit vorgeht, behandelt Apt die Schlüssel als abgelaufen.

Wenn Sie Testing oder Unstable verwenden, gibt es ein Problem, wenn Sie in letzter Zeit nicht **apt-get update** ausgeführt haben und mit **apt-get** ein Paket installieren möchten. Apt könnte sich darüber beschweren, dass es nicht authentifiziert werden konnte (Warum passiert das bloß?). **apt-get update** löst das Problem.

7.5.3.11. Prüfung von Hand

Für den Fall, dass Sie nun zusätzliche Sicherheitsprüfungen einführen wollen, aber nicht die neuste Version von apt einsetzen wollen oder können [62], können Sie das folgende Skript von Anthony Towns benutzen. Dieses Skript führt automatisch neue Sicherheitsüberprüfungen durch, damit ein Benutzer sicher gehen kann, dass die Software, die er herunterlädt, die gleiche ist wie die, die von Debian bereitgestellt wird. Das verhindert, dass sich Debian-Entwickler in ein fremdes System einhacken können, ohne dass eine Zurechnung und Rückverfolgung möglich wäre, die durch das Hochladen eines Pakets auf das Hauptarchiv gewährleistet werden. Es kann auch verhindern, dass ein Spiegel etwas fast genau abbildet, das aber eben doch nicht ganz wie in Debian, oder dass veraltete Versionen von instabilen Paketen mit bekannten Sicherheitslücken zur Verfügung gestellt werden.

Dieser Beispielscode, umbenannt nach **apt-check sigs**, sollte auf die folgende Art benutzt werden:

```
# apt-get update
# apt-check-sigs
(... Ergebnisse ...)
# apt-get dist-upgrade
```

Zuerst müssen Sie jedoch Folgendes tun:

- Holen Sie sich den Schlüssel, den die Archiv-Software verwendet, um **Release**-Dateien zu signieren, also http://ftp-master.debian.org/ziyi_key_2006.asc, und fügen Sie ihn **~/.gnupg/trustedkeys.gpg** hinzu (was standardmäßig von **gpgv** benutzt wird).

```
gpg --no-default-keyring --keyring trustedkeys.gpg --import ziyi_key_2006.asc
```

- Entfernen Sie alle Zeilen aus **/etc/apt/sources.list**, die nicht die normale »dists«-Struktur benutzen, oder ändern Sie das Skript, so dass es auch mit denen funktioniert.
- Ignorieren Sie die Tatsache, dass Sicherheitsaktualisierungen von Debian keine signierten **Release**-Dateien haben, und das **Sources**-Dateien (noch) keine richtigen Prüfsummen in der **Release**-Datei haben.
- Bereiten Sie sich darauf vor, zu prüfen, dass die richtigen Quellen durch den richtigen Schlüssel signiert wurden.

Dies ist der Beispielscode für **apt-check-sigs**. Die neuste Fassung ist unter http://people.debian.org/~ajt/apt-check-sigs erhältlich. Dieser Code befindet sich im Moment noch im Beta-Stadium. Für weitere Informationen sollten Sie http://lists.debian.org/debian-devel/2002/debian-devel-200207/msg00421.html lesen.

```
#!/bin/bash

#!/bin/bash

# Copyright (c) 2001 Anthony Towns <ajt@debian.org>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.

rm -rf /tmp/apt-release-check
mkdir /tmp/apt-release-check || exit 1
cd /tmp/apt-release-check

>OK
>MISSING
>NOCHECK
>BAD

arch=`dpkg --print-installation-architecture`

am_root () {
        [ `id -u` -eq 0 ]
}

get_md5sumsize () {
        cat "$1" | awk '/^MD5Sum:/,/^SHA1:/' |
          MYARG="$2" perl -ne '@f = split /\s+/; if ($f[3] eq $ENV{"MYARG"}) {
print "$f[1] $f[2]\n"; exit(0); }'
}

checkit () {
        local FILE="$1"
        local LOOKUP="$2"
```

```
        Y="`get_md5sumsize Release "$LOOKUP"`"
        Y="`echo "$Y" | sed 's/^ *//;s/  */ /g'`"

        if [ ! -e "/var/lib/apt/lists/$FILE" ]; then
                if [ "$Y" = "" ]; then
                        # No file, but not needed anyway
                        echo "OK"
                        return
                fi
                echo "$FILE" >>MISSING
                echo "MISSING $Y"
                return
        fi
        if [ "$Y" = "" ]; then
                echo "$FILE" >>NOCHECK
                echo "NOCHECK"
                return
        fi
        X="`md5sum < /var/lib/apt/lists/$FILE | cut -d\  -f1` `wc -c < /var/lib
/apt/lists/$FILE`"
        X="`echo "$X" | sed 's/^ *//;s/  */ /g'`"
        if [ "$X" != "$Y" ]; then
                echo "$FILE" >>BAD
                echo "BAD"
                return
        fi
        echo "$FILE" >>OK
        echo "OK"
}

echo
echo "Checking sources in /etc/apt/sources.list:"
echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
echo
(echo "You should take care to ensure that the distributions you're downloading
"
echo "are the ones you think you are downloading, and that they are as up to"
echo "date as you would expect (testing and unstable should be no more than"
echo "two or three days out of date, stable-updates no more than a few weeks"
echo "or a month)."
) | fmt
echo

cat /etc/apt/sources.list |
  sed 's/^ *//' | grep '^[^#]' |
  while read ty url dist comps; do
        if [ "${url%%:*}" = "http" -o "${url%%:*}" = "ftp" ]; then
                baseurl="${url#*://}"
        else
                continue
        fi

        echo "Source: ${ty} ${url} ${dist} ${comps}"

        rm -f Release Release.gpg
        lynx -reload -dump "${url}/dists/${dist}/Release" >/dev/null 2>&1
        wget -q -O Release "${url}/dists/${dist}/Release"

        if ! grep -q '^' Release; then
```

```
                echo "  * NO TOP-LEVEL Release FILE"
                >Release
        else
                origline=`sed -n 's/^Origin: *//p' Release | head -1`
                lablline=`sed -n 's/^Label: *//p' Release | head -1`
                suitline=`sed -n 's/^Suite: *//p' Release | head -1`
                codeline=`sed -n 's/^Codename: *//p' Release | head -1`
                dateline=`grep "^Date:" Release | head -1`
                dscrline=`grep "^Description:" Release | head -1`
                echo "  o Origin: $origline/$lablline"
                echo "  o Suite: $suitline/$codeline"
                echo "  o $dateline"
                echo "  o $dscrline"

                if [ "${dist%%/*}" != "$suitline" -a "${dist%%/*}" != "$codeline" ]; then
                        echo "  * WARNING: asked for $dist, got $suitline/$codeline"
                fi

                lynx -reload -dump "${url}/dists/${dist}/Release.gpg" >/dev/null 2>&1
                wget -q -O Release.gpg "${url}/dists/${dist}/Release.gpg"

                gpgv --status-fd 3 Release.gpg Release 3>&1 >/dev/null 2>&1 | sed -n "s/^\[GNUPG:\]
//p" | (okay=0; err=""; while read gpgcode rest; do
                        if [ "$gpgcode" = "GOODSIG" ]; then
                            if [ "$err" != "" ]; then
                                echo "  * Signed by ${err# } key: ${rest#* }"
                            else
                                echo "  o Signed by: ${rest#* }"
                                okay=1
                            fi
                            err=""
                        elif [ "$gpgcode" = "BADSIG" ]; then
                            echo "  * BAD SIGNATURE BY: ${rest#* }"
                            err=""
                        elif [ "$gpgcode" = "ERRSIG" ]; then
                            echo "  * COULDN'T CHECK SIGNATURE BY KEYID: ${rest %% *}"
                            err=""
                        elif [ "$gpgcode" = "SIGREVOKED" ]; then
                            err="$err REVOKED"
                        elif [ "$gpgcode" = "SIGEXPIRED" ]; then
                            err="$err EXPIRED"
                        fi
                    done
                    if [ "$okay" != 1 ]; then
                        echo "  * NO VALID SIGNATURE"
                        >Release
                    fi)
        fi
        okaycomps=""
        for comp in $comps; do
                if [ "$ty" = "deb" ]; then
                        X=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/binary-${arch}/Release" |
sed 's,//*,_,g'`" "${comp}/binary-${arch}/Release")
                        Y=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/binary-${arch}/Packages" |
sed 's,//*,_,g'`" "${comp}/binary-${arch}/Packages")
                        if [ "$X $Y" = "OK OK" ]; then
                                okaycomps="$okaycomps $comp"
                        else
                                echo "  * PROBLEMS WITH $comp ($X, $Y)"
                        fi
```

```
                elif [ "$ty" = "deb-src" ]; then
                        X=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/source/Release" | sed
's,//*,_,g'`" "${comp}/source/Release")
                        Y=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/source/Sources" | sed
's,//*,_,g'`" "${comp}/source/Sources")
                        if [ "$X $Y" = "OK OK" ]; then
                                okaycomps="$okaycomps $comp"
                        else
                                echo "  * PROBLEMS WITH component $comp ($X, $Y)"
                        fi
                fi
        done
        [ "$okaycomps" = "" ] || echo "  o Okay:$okaycomps"
        echo
  done

echo "Results"
echo "~~~~~~~"
echo

allokay=true

cd /tmp/apt-release-check
diff <(cat BAD MISSING NOCHECK OK | sort) <(cd /var/lib/apt/lists && find . -type f -maxdepth 1 | sed
's,^\./,,g' | grep '_' | sort) | sed -n 's/^> //p' >UNVALIDATED

cd /tmp/apt-release-check
if grep -q ^ UNVALIDATED; then
    allokay=false
    (echo "The following files in /var/lib/apt/lists have not been validated."
    echo "This could turn out to be a harmless indication that this script"
    echo "is buggy or out of date, or it could let trojaned packages get onto"
    echo "your system."
    ) | fmt
    echo
    sed 's/^/    /' < UNVALIDATED
    echo
fi

if grep -q ^ BAD; then
    allokay=false
    (echo "The contents of the following files in /var/lib/apt/lists does not"
    echo "match what was expected. This may mean these sources are out of date,"
    echo "that the archive is having problems, or that someone is actively"
    echo "using your mirror to distribute trojans."
    if am_root; then
        echo "The files have been renamed to have the extension .FAILED and"
        echo "will be ignored by apt."
        cat BAD | while read a; do
            mv /var/lib/apt/lists/$a /var/lib/apt/lists/${a}.FAILED
        done
    fi) | fmt
    echo
    sed 's/^/    /' < BAD
    echo
fi

if grep -q ^ MISSING; then
    allokay=false
    (echo "The following files from /var/lib/apt/lists were missing. This"
```

```
    echo "may cause you to miss out on updates to some vulnerable packages."
    ) | fmt
    echo
    sed 's/^/    /' > MISSING
    echo
fi

if grep -q ^ NOCHECK; then
    allokay=false
    (echo "The contents of the following files in /var/lib/apt/lists could not"
    echo "be validated due to the lack of a signed Release file, or the lack"
    echo "of an appropriate entry in a signed Release file. This probably"
    echo "means that the maintainers of these sources are slack, but may mean"
    echo "these sources are being actively used to distribute trojans."
    if am_root; then
        echo "The files have been renamed to have the extension .FAILED and"
        echo "will be ignored by apt."
        cat NOCHECK | while read a; do
            mv /var/lib/apt/lists/$a /var/lib/apt/lists/${a}.FAILED
        done
    fi) | fmt
    echo
    sed 's/^/    /' > NOCHECK
    echo
fi

if $allokay; then
    echo 'Everything seems okay!'
    echo
fi

rm -rf /tmp/apt-release-check
```

Sie müssen vielleicht bei *Sid* diesen Patch verwenden, da **md5sum** ein »-« an die Summe anfügt, wenn die Ausgabe auf stdin erfolgt:

```
@@ -37,7 +37,7 @@
        local LOOKUP="$2"

        Y="`get_md5sumsize Release "$LOOKUP"`"
-       Y="`echo "$Y" | sed 's/^ *//;s/  */ /g'`"
+       Y="`echo "$Y" | sed 's/-//;s/^ *//;s/  */ /g'`"

        if [ ! -e "/var/lib/apt/lists/$FILE" ]; then
                if [ "$Y" = "" ]; then
@@ -55,7 +55,7 @@
                return
        fi
        X="`md5sum < /var/lib/apt/lists/$FILE` `wc -c < /var/lib/apt/lists/$FILE`"
-       X="`echo "$X" | sed 's/^ *//;s/  */ /g'`"
+       X="`echo "$X" | sed 's/-//;s/^ *//;s/  */ /g'`"
        if [ "$X" != "$Y" ]; then
                echo "$FILE" >>BAD
                echo "BAD"
```

7.5.4. Prüfung der Release-Datei von Debian-fremden Quellen

Beachten Sie, dass, wenn Sie die neuste Version von Apt (mit *Secure Apt*) einsetzen, kein zusätzlicher Aufwand auf Ihrer Seite notwendig sein sollte, wenn Sie keine Debian-fremden Quellen verwenden. Anderenfalls erfordert **apt-get** eine zusätzliche Bestätigung. Dies wird verhindert, wenn **Release**- und **Release.gpg**-Dateien in den Debian-fremden Quellen zur Verfügung stehen. Die **Release**-Datei kann mit **apt-ftparchive** (ist in **apt-utils** 0.5.0 und später enthalten) erstellt werden, die **Release.gpg** ist nur die abgetrennte Signatur. Beide können mit folgender einfacher Prozedur erstellt werden:

```
$ rm -f dists/unstable/Release
$ apt-ftparchive release dists/unstable > dists/unstable/Release
$ gpg --sign -ba -o dists/unstable/Release.gpg dists/unstable/Release
```

7.5.5. Alternativer Entwurf zur Einzelsignierung von Paketen

Dieser zusätzliche Entwurf, jedes Paket einzeln zu signieren, erlaubt es, Pakete zu prüfen, selbst wenn sie nicht mehr in irgendeiner **Packages**-Datei erwähnt werden. Und so können auch Pakete von Dritten, für die es nie eine **Packages**-Datei gab, unter Debian installiert werden. Dies wird aber kein Standard werden.

Dieser Entwurf zur Paketsignierung kann mit **debsig-verify** und **debsigs** umgesetzt werden. Diese beiden Pakete können in einer .deb-Datei eingebettete Unterschriften erstellen und prüfen. Debian hat bereits jetzt die Möglichkeiten, dies zu tun. Aber es gibt keine Planung, dieses Regelwerk oder ähnliche Werkzeuge umzusetzen, da nunmehr das Schema mit der Signierung des Archivs bevorzugt wird. Die Werkzeuge werden dennoch für Benutzer und Administratoren von Archiven zur Verfügung gestellt, wenn sie diese Vorgehensweise bevorzugen.

Die aktuellen Versionen von **dpkg** (seit 1.9.21) beinhalten einen http://lists.debian.org/debian-dpkg/2001/debian-dpkg-200103/msg00024.html, der diese Funktionen zur Verfügung stellt, sobald **debsig-verify** installiert ist.

HINWEIS: Derzeit wird **/etc/dpkg/dpkg.cfg** standardmäßig mit der Option »no-debsig« ausgeliefert.

HINWEIS 2: Unterschriften von Entwicklern werden im Moment entfernt, wenn sie in das Paketarchiv gelangen, da die derzeit vorzugswürdige Methode die Überprüfung der Release-Datei ist, wie es oben beschrieben wurde.

---

[55] Einige Betriebssysteme wurden schon von Problemen mit automatischen Aktualisierungen heimgesucht, wie z.B. die http://www.cunap.com/~hardingr/projects/osx/exploit.html.

[56] Ältere Veröffentlichungen wie Debian 3.1 (*Sarge*) können mit zurückportierten Versionen des Paketmanagers auf diese Methode zugreifen.

[57] Bis ein automatischer Mechanismus entwickelt wurde.

[58] Genau genommen handelt es sich um eine ASCII-armored abgetrennte GPG-Signatur.

[59] Oder Ihren DNS vergiftet hat oder den Server spooft oder die Datei auf einem Spiegel platziert hat, den Sie verwenden, oder ...

[60] »Ziyi« ist der Name des Werkzeugs, mit dem die Debian-Server signiert werden, und beruht auf dem Namen einer http://de.wikipedia.org/wiki/Ziyi_Zhang.

[61] Nicht alle Schlüssel der Apt-Depots sind überhaupt mit einem anderen Schlüssel unterschrieben. Vielleicht hat derjenige, der das Depot einrichtet, keinen anderen Schlüssel zur Verfügung, oder vielleicht ist es ihm unangenehm, einen Schlüssel mit einer derartig wichtigen Funktion mit seinem Hauptschlüssel zu unterschreiben. Hinweise, wie man einen Schlüssel für ein Depot einrichtet, finden Sie unter Abschnitt 7.5.4, „Prüfung der Release-Datei von Debian-fremden Quellen“.

[62] Entweder weil Sie Stable (*Sarge*) oder eine ältere Veröffentlichung verwenden, oder weil Sie nicht die neuste Version von Apt einsetzen wollen, obwohl wir das Testen wirklich schätzen würden.

Zurück ben Zum Anfa Weiter
7.4. Die Infrastruktur des Sicherheitsprozesses i... Kapitel 8. Sicherheitswerkzeuge in Debian