In: Bauknecht, K.; Teufel, S. (Hrsg.): *Sicherheit in Informationssystemen*. Proceedings der Fachtagung SIS '94, 10.-11.3.1994 in Zürich, vdf Zürich 1994, S. 161-173.

SECUBOOT
Authentisches *remote boot* für Client-Server-Netzwerke

Dirk Fox
Universität Siegen, Fachbereich 12
Institut für Nachrichtenübermittlung, D-57068 Siegen
fox@nue.et-inf.uni-siegen.de

Manfred Böttger
Schneider & Koch & Co. Datensysteme GmbH
Siemensstraße 23, D-76275 Ettlingen
mboettger@skd.de

Zusammenfassung

Alle existierenden Realiserungen eines *remote boot*-Prozesses in Client-Server-Netzwerken enthalten eine erhebliche Sicherheitslücke: Da das Booten der Arbeitsstation vor dem Benutzer-Login im Netzwerk-Betriebssystem stattfindet, greifen die dort vorgesehenen Sicherheitsmechanismen noch nicht. Ein Angreifer kann unbemerkt den Boot-Vorgang kontrollieren, indem er sich als Boot-Server ausgibt. So können sämtliche Sicherheitsmechanismen des Betriebssystems unterlaufen werden.

Einen Ausweg aus diesem Sicherheitsproblem bietet die Realisierung eines authentischen Boot-Prozesses, der mit Hilfe kryptographischer Verfahren eine Fälschung der Boot-Daten unmöglich macht.

Ein konkretes, prototypisch realisiertes Lösungskonzept (SECUBOOT) für PC-LANs wird vorgestellt, das unter Verwendung der kryptographischen Hash-Funktion MD5 und des Digitalen Signatursystems RSA ein authentisches *remote boot* unter NetWare (Novell) ermöglicht.

1 Einleitung

Ein *remote boot*, d.h. das Booten der Arbeitsstationen in Client-Server-Netzwerken von einem zentralen Boot-Server, hat viele Vorzüge: Es ermöglicht eine zentrale Installation, Konfiguration und Pflege von Treibern, Betriebssystemen und Anwendungsprogrammen, die in vernetzten Arbeitsstationen während des Bootvorgangs zu laden sind. Dadurch sinkt der Administrationsaufwand; zugleich steigen Stabilität und Zuverlässigkeit der Netzwerkumgebung, da die lokal geladenen Treiber und Betriebssysteme denselben Versionsstand besitzen. Hinzu kommen wirtschaftliche Vorteile, denn Netzwerklizenzen für die eingesetzte Betriebssystem- und Anwendungssoftware sind meist günstiger als Einzellizenzen.

Aus Sicherheitsgründen sollten in Netzen zudem bevorzugt *diskless workstations*, d.h. Arbeitsstationen ohne Laufwerke und Festplatte eingesetzt werden: Dadurch kann sowohl der Aufenthalt wichtiger Daten im Netz auf den oder die zentralen, besonders gesicherten File-Server beschränkt als auch der Einsatz unlizenzierter oder virenverseuchter Software weitgehend verhindert werden. In Netzen mit *diskless workstations* ist ein *remote boot* unverzichtbar.

Vom Sicherheitsstandpunkt ist der Boot-Prozeß eines Rechners besonders sensibel: Kann nicht garantiert werden, daß beim Starten eines Rechners das Betriebssystem und alle Programme wie vorgesehen unverfälscht und vollständig geladen werden, können die vorgesehenen Sicherheitsmechanismen natürlich keinen wirkungsvollen Schutz bieten. Ein sicheres, d.h. authentisches Booten ist daher elementare Voraussetzung jeglicher Sicherheitsmechanismen [Groß_91].

2 Sicherheitslücke im *remote boot*-Prozeß

Der Boot-Prozeß einer lokal, d.h. von Festplatte oder Diskette bootenden Arbeitsstation kann nur mit viel Aufwand authentisch realisiert werden. Lösungsansätze sind die Verschlüsselung der Boot-Sektoren auf der Festplatte oder eine kryptographische Authentizitätsprüfung

der Boot-Daten [OsSa_93]. Grundsätzliches Problem dabei: Lokal müssen geheime Schlüsseldaten (z.B. ein Paßwort) gespeichert werden und sind damit einem Angreifer prinzipiell zugänglich. Einziger Ausweg ist die Verwendung eines externen Sicherheitstoken (z.B. einer intelligenten Chipkarte), der diese Authentizitätsprüfung vornimmt. Bisherige Entwicklungen sind allerdings noch von einer befriedigenden Lösung entfernt und sehr teuer.

In Client-Server-Netzwerken scheint ein authentischer Boot-Vorgang via *remote boot* auf den ersten Blick leichter möglich: Die Boot-Daten liegen sicher auf einem zentralen, physisch schützbaren Boot-Server, auf dem auch geheime Schlüsseldaten unzugänglich aufbewahrt werden können.

Aber auch dieser *remote boot*-Prozeß ist problematisch: Die Sicherheitsmechanismen des Netzwerkbetriebssystems greifen zu diesem Zeitpunkt noch nicht, da der Benutzer nicht angemeldet ist. Der erste Kontakt zwischen Arbeitsstation und Boot-Server erfolgt daher bei allen *remote boot*-Realisierungen ungeschützt, d.h. ohne gegenseitige Authentisierung und ohne Integritätsschutz. Er kann in lokalen Netzwerken wegen der Broadcast-Eigenschaft der gängigen Technologien (Ethernet, Token Ring, FDDI) von jeder Stelle des Netzes gestört werden: Ein Eindringling kann sich unerkannt als Boot-Server ausgeben und Boot-Daten seiner Wahl an die anfragende Station senden.

Ein solcher aktiver Angriff kann vollkommen automatisiert, beispielsweise mit einem an eine freie Netz-Anschlußdose angeklemmten Notebook-Computer erfolgen. Auf diese Weise lassen sich trojanische Pferde in der Arbeitsstation installieren. Alle Sicherheitsmechanismen von Betriebssystem und Anwendungen können auf diese Weise ausgehebelt werden (siehe Bild 2-1).[1]

---

1 Leicht zu realisieren ist beispielsweise das Ausspähen des Benutzerpaßwortes: Dazu genügt ein residentes Programm, das die beiden ersten Eingaben des Benutzers an den Angreifer zurückschickt.

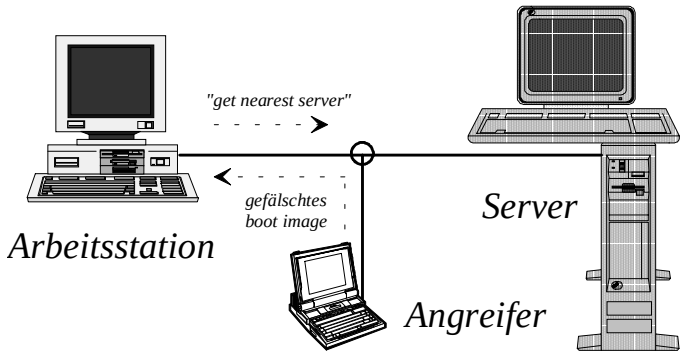

Bild 2-1: Fälschung des *remote boot*-Vorgangs am Beispiel von NetWare

In Token-Ring- und FDDI-Netzen fällt ein Eindringling immerhin dem Ring-Management auf, da er dort als neue Station registriert wird. In Ethernet-LANs bleibt er gänzlich unbemerkt. Bei *thick wire*-Verkabelung ist nicht einmal eine freie Anschlußdose erforderlich: es genügt eine Vampirklemme.

Sternverkabelung und der Einsatz filternder Brücken, die die Ausbreitung von Nachrichten auf die notwendig zu durchlaufenden Netzsegmente beschränken, können die Angriffspunkte im Netz zwar verringern, aber nicht vollständig ausschließen. An dieser Stelle klafft also eine erhebliche Sicherheitslücke.

3 Authentisches *remote boot*

Ein sicherer *remote boot*-Prozeß setzt also voraus, daß Unverfälschtheit und Originalität der vom Server an die bootende Arbeitsstation übertragenen Daten sichergestellt werden [ISO_89]. Die Prüfung der empfangenen Boot-Daten muß dabei durch den Boot-Code erfolgen. Dazu ist die Ergänzung der Boot-Daten um unfälschbare, individuelle und überprüfbare Merkmale erforderlich. Grundsätzlich kommen dafür verschiedene Verfahren in Frage.

3.1 Authentizität mit asymmetrischer Kryptographie

Einen besonders hohes Maß an Sicherheit erreicht man durch die Verwendung einer Digitalen Signatur als Authentizitätsmerkmal. Digitale Signatursysteme basieren auf asymmetrischen Kryptoverfahren, die mit Schlüsselpaaren (g, ö) aus einem geheimen Schlüssel g und einem allgemein bekannten, öffentlichen Schlüssel ö arbeiten [DiHe_76]. Der Signierer berechnet mit dem nur ihm bekannten geheimen Schlüssel und dem Signieralgorithmus S zu einer Dateneinheit D die Signatur:

$$Sig = S(D, g)$$

und hängt diese an die zu schützende Dateneinheit an. Jeder Empfänger der Dateneinheit D' mit zugehöriger Signatur Sig' kann dann deren Unverfälschtheit und Originalität mit dem passenden öffentlichen Schlüssel ö und dem Prüfalgorithmus P überprüfen:

$$P(D', Sig', ö) = true\ ?$$

Üblicherweise wird die Dateneinheit nicht unmittelbar signiert, sondern erst ein kryptographischer Hashwert H(D) bestimmt,[2] zu dem dann die Signatur berechnet wird: Sig = S(H(D), g). Dadurch werden der Signiervorgang beschleunigt, der Speicherbedarf der Signatur reduziert und jede arithmetische Struktur des Signieralgorithmus' zerstört.[3]

Unverfälschtheit und Originalität der Boot-Daten können während eines *remote boot*-Vorgangs mit Hilfe eines Digitalen Signatursystems wie folgt sichergestellt werden: Mit einer kryptographischen Hashfunktion H wird zuvor der "Fingerabdruck" H(B) der Boot-Daten B berechnet und mit dem geheimen, nur dem Sicherheitsadministrator

---

2 In der Literatur wird dieser Wert häufig auch als "Fingerabdruck" oder *message digest* bezeichnet.

3 So erleichtert beispielsweise die Multiplikativität des Digitalen Signatursystems RSA prinzipiell eine Fälschung. Es gilt: $S(D_1 \bullet D_2, g) = S(D_1, g) \bullet S(D_2, g)$. Eine kryptographische Hashfunktion zerstört diese Struktur: $S(H(D_1) \bullet H(D_2), g) \neq S(H(D_1), g) \bullet S(H(D_2), g)$.

bekannten Schlüssel g die Digitale Signatur dieses "Fingerabdrucks" berechnet:

$$Sig = S(H(B), g)$$

berechnet. Diese wird an die Boot-Daten angehängt.

Die Digitale Signatur selbst muß praktisch unfälschbar sein, d.h. es darf auch mit erheblicher Rechenleistung nicht in vernünftiger Zeit gelingen, unbefugt, also ohne Kenntnis des geheimen Signierschlüssels g, eine gültige Digitale Signatur zu irgendwelchen (unsignierten) Boot-Daten zu bestimmen.

In der bootenden Station berechnet der Boot-Code aus empfangenen Boot-Daten B' zunächst den kryptographischen "Fingerabdruck" H(B') und prüft die empfangene Digitale Signatur Sig' mit dem im Boot-Code eingetragenen, zum Signierschlüssel passenden öffentlichen Schlüssel ö:

$$P(H(B'), Sig', ö) = true\ ?$$

Der Prüfschlüssel darf allgemein bekannt sein, denn der geheime Schlüssel g kann aus ö nicht gewonnen werden. Allerdings muß die Integrität des öffentlichen Schlüssels sichergestellt werden, z.B. indem der Boot-Code von einem schreibgeschützten Datenträger oder aus einem ROM gestartet wird.[4]

Vorteil dieser asymmetrischen Lösung ist, daß in den Arbeitsstationen keine geheimen Schlüssel oder Merkmale aufbewahrt werden müssen. Der Boot-Code muß also nicht aufwendig "personalisiert" werden: Netzwerkweit können alle Boot-Daten mit demselben geheimen Schlüssel g signiert und mit einem einheitlichen Testschlüssel ö geprüft werden, ohne daß eine Signaturfälschung möglich wäre – vorausgesetzt natürlich, der geheimzuhaltende Signierschlüssel wird vom Administrator sorgfältig verwahrt. Je Netzwerk (bzw.

---

4 In PC-Netzwerken erfolgt der *remote boot*-Prozeß üblicherweise mit Hilfe eines sogenannten Boot-ROMs, das auf dem Netzwerkadapter eingesteckt wird.

Sicherheitsadministrator) ist unabhängig von der Stationenzahl daher nur ein einziges Schlüsselpaar (g, ö) erforderlich.

Diese Lösung erfordert auch keine Änderungen des Boot-Vorgangs im Server, da das Signieren der Boot-Daten einmalig und *offline*, d.h. bei der Installation durch den Administrator erfolgen kann.

3.2 Authentizität mit symmetrischer Kryptographie

Eine Alternative zur Verwendung eines Digitalen Signatursystems ist ein symmetrischer Integritätsschutz, der die Unverfälschtheit der übertragenen Boot-Daten mit Hilfe eines kryptographischen Integritätswertes garantiert. Dazu wird zu jeder zu übertragenen Dateneinheit D mit Hilfe eines geheimen Schlüssels k und einem Authentizitätsalgorithmus A ein 2 bis 4 Byte langer Authentikator

$$Auth = A(D, k)$$

berechnet, der an die Daten angehängt wird [ISO_87]. Die Arbeitsstation, die diesen geheimen Schlüssel k kennt, kann den zu der empfangenen Dateneinheit D' gehörigen Authentikator Auth' prüfen:

$$A(D', k) = Auth' ?$$

Mit Hilfe eines sogenannten *challenge response*-Mechanismus kann zugleich eine indirekte gegenseitige Authentisierung von Boot-Server und bootender Arbeitsstation erfolgen (siehe z.B. [Fumy_93]).

Entscheidender Nachteil eines solchen symmetrischen Integritätsverfahrens: Sender und Empfänger, d.h. in diesem Fall Boot-Server und Boot-Code der Arbeitsstation, müssen über einen gemeinsamen geheimen Schlüssel k verfügen, der in der Arbeitsstation geschützt aufzubewahren ist. Trägt man ihn fest in den Boot-Code ein, kann er prinzipiell von einem Eindringling ausgelesen und für die Fälschung von Boot-Daten verwendet werden.

Zudem entstehen organisatorische Schwierigkeiten: Für jede Arbeitsstation (d.h. jeden LAN-Adapter) muß ein individueller Schlüssel er-

zeugt und in den Boot-Code eingetragen werden.[5] Dadurch ist diese Lösung relativ unflexibel und produktionstechnisch aufwendig.

Weiter muß der Server vor der Übertragung der Boot-Daten den richtigen Schlüssel für die Berechnung des Authentikators aus einer Schlüsselliste heraussuchen und verwenden: Dies würde eine aufwendige Sonderbehandlung von Paketen mit Boot-Daten und die Verwaltung von Benutzer-IDs und zugehörigen Schlüsseln erfordern.

3.3 Authentizität ohne Kryptographie

Schließlich gibt es noch die Möglichkeit, in der bootenden Station ausschließlich Boot-Daten von einem bestimmten, zuvor festgelegten Boot-Server zu akzeptieren.

Diese Maßnahme bietet allerdings keinen verläßlichen Schutz. Entscheidender Nachteil: Das unfälschbare Merkmal, hier die Netzwerkadresse des Boot-Servers, hängt nicht von den übertragenen Boot-Daten ab, sondern bezeichnet lediglich den Sender. Damit schützt das Verfahren weder vor Maskerade noch vor einer Modifikation der Daten während der Übertragung.[6]

Die Lösung ist zudem sehr unflexibel, da die Netzwerkadresse des Boot-Servers im Boot-Code fest eingetragen werden muß, und damit produktionstechnisch problematisch.[7] In jedem Fall sind Installation und Konfiguration umständlich.

---

5 Ein netzwerkweit einheitlicher Boot-Schlüssel würde einem Angreifer die Arbeit deutlich erleichtern; für Netzbenutzer wäre eine Fälschung besonders einfach.

6 Allerdings wird einem Eindringling die Arbeit erschwert: Die Fälschung der Senderadresse bleibt im allgemeinem in einem Broadcast-LAN nur für die Dauer von wenigen Paketen unbemerkt, wenn die maskierte Station selbst aktiv ist.

7 Wird ein Boot-ROM verwendet, müssten hier *remote* ladbare Flash- oder EPROMS eingesetzt werden. Dann wären allerdings auch Vorkehrungen gegen unautorisierte Modifikationen des Boot-Codes zu treffen.

4 Authentisches *remote boot* für NetWare-LANs

In PC-Netzwerken unter dem Betriebssystem NetWare (Novell) ist ein Angriff auf den *remote boot*-Vorgang besonders einfach: Da eine Station nach dem Einschalten zunächst keinen der im Netz aktiven Server kennt, fragt sie zu Beginn mit einer Broadcast-Nachricht ("*get nearest server*") nach einem Boot-Server. Antworten mehrere Boot-Server, fordert sie von der ersten antwortenden Station die in einer Datei, dem sogenannten *boot image* zusammengefaßten Boot-Daten an (siehe auch Bild 2-1).[8]

Ein Eindringling kann der Arbeitsstation unbemerkt ein *boot image* seiner Wahl unterschieben, wenn er schneller als die Server desselben Netzsegmentes reagiert. Das ist sehr leicht möglich, da in größeren Netzwerken Server üblicherweise beschäftigt sind und daher erst mit einer gewissen Verzögerung auf den Rundruf antworten.

Schneider & Koch hat mit SECUBOOT eine Lösung entwickelt, die ein authentisches *remote boot* in PC-LANs unter NetWare ermöglicht. Der Prototyp, ein Boot-ROM für Ethernet-PC-LAN-Adapter, trägt dabei dem speziellen *remote boot*-Prozeß des Netzwerkbetriebssystems NetWare Rechnung.

4.1 Besonderheiten des *remote boot* unter NetWare

Beim remote boot-Prozeß unter NetWare wird der Arbeitsstation durch das Boot-ROM des Netzwerkadapters ein Booten von einem Boot-Laufwerk vorgespielt; die Boot-Daten werden jedoch nicht von den Boot-Sektoren einer Boot-Diskette, sondern aus einer Datei vom Boot-Server eingelesen.

Will man ein authentisches Booten auf die in Abschnitt 3.1 beschriebene Weise für PC-Netze unter NetWare realisieren, bereitet dieser Boot-Vorgang Schwierigkeiten: Das sektorweise Einlesen des *boot image* erfolgt nämlich in nicht (allgemein) vorhersagbarer Reihen-

---

8 Eine recht übersichtliche Erläuterung des *remote boot*-Prozesses unter NetWare (Novell) findet sich in [BrUn_93].

folge und nicht notwendig vollständig. Eine Digitale Signatur des gesamten *boot image* kann daher nicht verwendet werden: Erhält die Arbeitsstation nicht alle Sektoren des *boot image* in der richtigen Reihenfolge, kann die Signatur nicht geprüft werden.

Außerdem dürfte das Boot-ROM den geladenen Boot-Code erst nach dem Laden des gesamten *boot image* und der Prüfung von Hashwert und Signatur zur Ausführung freigeben. Dazu müßte erheblich in den Boot-Prozeß des BIOS eingegriffen werden. Selbst dann wäre diese Lösung für ein mehrere hundert kByte großes *boot image* in einem DOS-PC nicht realisierbar. Das Verfahren zum authentischen Booten muß daher etwas modifiziert werden.

4.2 Erstellen der Boot-Signatur

Eine mögliche Lösung des Problems wäre ein separates Signieren aller Sektoren des *boot image*. Dieser Weg scheidet jedoch aus zwei Gründen aus: Erstens wäre der Rechenaufwand erheblich, da für jeden der 512 Byte großen Sektoren eine Sektor-Signatur vom Boot-ROM getestet werden müßte. Zweitens wäre je 512-Byte-Sektor eine Signatur von 64 bis 128 Byte Länge zu speichern: Das *boot image* würde dadurch um ca. ein Drittel verlängert.

Einen Ausweg bietet die Verwendung von Sektor-Hashwerten statt Sektor-Signaturen. Dazu wird beim Signieren des *boot image* zunächst zu jedem Sektor ein Hashwert berechnet und in der Reihenfolge der Sektornummern in eine Tabelle eingetragen. Je Sektor fallen dabei lediglich 16 bis 20 Byte an. Aus den Hashwerten wird anschliessend ein "Meta-Hashwert" gebildet und zu diesem die Digitale Signatur bestimmt. Auf diese Weise wird die Reihenfolge der Sektoren unfälschbar festgelegt; jede Vertauschung wird vom Boot-ROM bei der Prüfung des Meta-Hashwertes sofort erkannt.

Die Signatur des *boot image* besteht damit aus der Hashwerttabelle, dem Meta-Hashwert und der Digitalen Signatur des Meta-Hashwertes (siehe Bild 4-1).

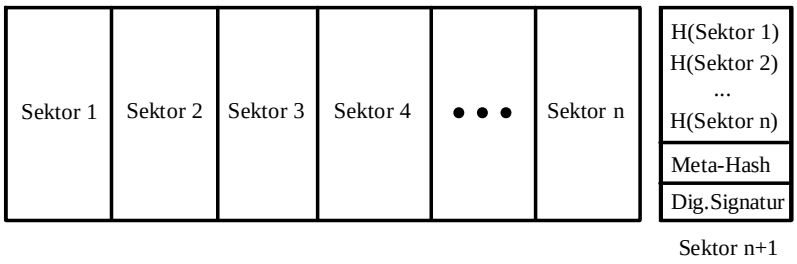

Bild 4-1: Signieren des *boot image*: Berechnung der Sektor-Hashwerte H(Sektor i), des Meta-Hashwertes und der Digitalen Signatur S(Meta-Hashwert, g); Anhängen an die Boot-Daten.

Dieser Signiervorgang, bei dem je nach Größe des *boot image* (oft 800 bis 1000 Sektoren) und Länge der Sektor-Hashwerte (16-20 Byte) bis zu 20 kByte Signaturdaten angehängt werden, muß bei der Einrichtung eines *boot image* im Server einmalig *offline* durchgeführt werden (üblicherweise nur ein einziges Mal). Weitere Änderungen am Server sind nicht erforderlich.

4.3 Prüfung der Authentizität beim Booten

Beim *remote boot* wird das *boot image* unverändert sektorweise eingelesen; begonnen wird jedoch mit der angehängten Signatur (Hashwerttabelle, Meta-Hashwert und Digitale Signatur). Es werden zunächst der "Meta-Hashwert" und, mit Hilfe des im Boot-ROM fest eingetragenen öffentlichen Schlüssels ö, die Digitale Signatur geprüft (siehe Bild 4-2).

Besteht die Signatur die Prüfung, dann sind der Meta-Hashwert und damit auch die vom Boot-ROM empfangene Hashwerttabelle unverfälscht und authentisch. Im weiteren Verlauf des Boot-Prozesses genügt es daher, zu jedem empfangenen Sektor des *boot image* den kryptographischen "Fingerabdruck" zu berechnen und mit dem Eintrag in der Hashwerttabelle zu vergleichen. Eine Fälschung ist allein durch eine Änderung des im Boot-Code eingetragenen öffentlichen Schlüssels und damit nur durch Modifikation oder Austausch des Boot-ROMs möglich.

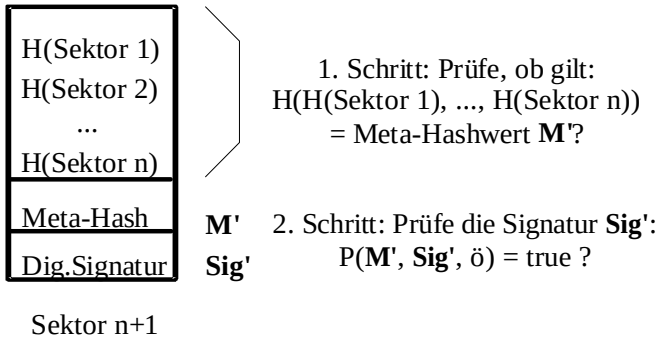

Bild 4-2 Feststellung der Unverfälschtheit der Signaturdaten: Einlesen der Hashwerttabelle, Berechnung des Meta-Hashwertes und Prüfen der Digitalen Signatur.

Das gesamte SECUBOOT-Paket setzt sich zusammen aus einem Programm zum Signieren einer *boot image*-Datei auf dem Server, einer Datei mit einem geheimen Signierschlüssel und den SECUBOOT-Boot-ROMs für die Netzwerkadapter, die den speziellen Boot-ROM-Code für authentisches Booten und den zum geheimen Signierschlüssel passenden öffentlichen Prüfschlüssel enthalten.

Als Hashfunktion wurde der von RSA Inc. entwickelte Algorithmus MD5 gewählt, der 16 Byte Hashwerte erzeugt [Rive_92]. Die Digitalen Signaturen werden mit dem RSA-Verfahren und einem 512 Bit langen Schlüssel in Software berechnet [RSA_78, Fox_93].

4.4 Sicherstellung von Aktualität

Ein wichtiger Teilaspekt des Integritätsschutzes von Daten ist streng genommen auch deren Aktualität, um *replay*-Angriffe[9] auszuschliessen. Beim Integritätsschutz der Datenübertragung wird dies im allgemeinen durch Zeitstempel erreicht, die die Gültigkeit des unfälschbaren Integritätswertes bzw. der Digitalen Signatur einer Dateneinheit zeitlich begrenzen [DaPr_89].

---

9 Das Wiedereinspielen von zu einem früheren Zeitpunkt abgehörten Daten.

Auch für den authentischen *remote boot*-Prozeß sind Zeitstempel zur Sicherstellung von Aktualität geeignet. Das Boot-ROM muß bei der Prüfung des Zeitstempels jedoch auf eine authentische Uhr zugreifen können. Die Uhr der Arbeitsstation kann dafür im allgemeinen nicht verwendet werden, da sie von einem Eindringling modifiziert werden kann.

Einziger Ausweg ist ein Protokoll zwischen Arbeitsstation und Boot-Server zum authentischen Austausch des Standes der Server-Uhrzeit unmittelbar vor der Prüfung der Signatur des *boot image*: Dazu schickt die Arbeitsstation eine Zufallszahl z an den Server, die dieser zusammen mit der aktuellen Uhrzeit t signiert zurücksendet (siehe Bild 3-1).

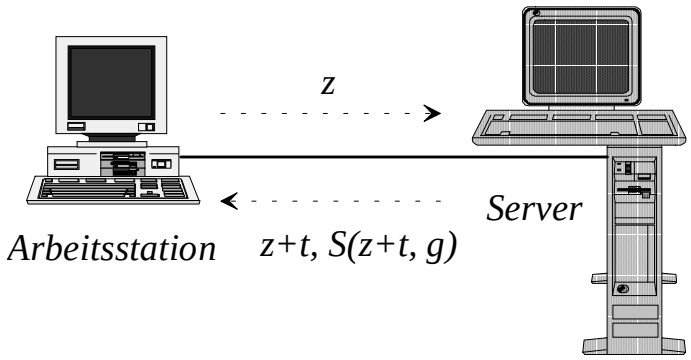

Bild 3-1: Austausch einer authentischen Uhrzeit (*z*: Zufallszahl, *t*: Serverzeit)

Die Aktualität der vom Server geschickten Zeit wird durch die signierte, nur einmalig zu verwendende Zufallszahl z der Arbeitsstation sichergestellt. Die Sicherstellung von Aktualität erfordert auch eine regelmäßige Erneuerung von Zeitstempel und Signatur des *boot image*. Auch dieser Mechanismus ist mit Bedacht einzusetzen: Ersetzt ein Administrator vor Ablauf des Gültigkeitszeitraums die Boot-Daten, besitzt die Signatur der alten Daten natürlich nach wie vor Gültigkeit; ein Eindringling kann sie dann wie oben beschrieben einer Arbeitsstation unbemerkt unterschieben.

Idealerweise werden die Boot-Daten daher *online*, d.h. erst unmittelbar vor der Übertragung mit einem Zeitstempel versehen und digital signiert. In diesem Fall kann das Protokoll zum Austausch einer authentischen Uhrzeit entfallen: Es genügt die Übertragung einer Zufallszahl z an den Server und die Verwendung von z+1 als Zeitstempel. Dabei ist allerdings eine Sonderbehandlung von Boot-Datenpaketen durch den Server erforderlich.

Bei dem Prototyp SECUBOOT wurde auf die Sicherstellung von Aktualität verzichtet. Ein *replay*-Angriff durch Wiedereinspielen eines alten, signierten *boot images* ist in der Praxis für einen Eindringling wenig erfolgversprechend. Eine Lösung, die das *boot image online* mit einem Zeitstempel und einer Digitalen Signatur versieht, wäre hingegen nur sehr aufwendig zu realisieren. Sie würde erhebliche Änderungen in der Serversoftware erfordern und könnte daher auch nicht ohne weiteres mit herkömmlichen, nicht authentisch bootenden Boot-ROMs gemeinsam in einem Netz verwendet werden.

5 Bewertung

Das vorgestellte Konzept zur Realisierung eines authentischen Boot-Vorgangs ist prinzipiell für alle (Netz-) Betriebssysteme mit *remote boot*-Fähigkeit geeignet. Die notwendige Integration der kryptographischen Mechanismen in den jeweiligen Boot-Code erfordert allerdings Anpassungen an die speziellen Erfordernisse des Betriebssystems und die Realisierung des *remote boot*-Vorgangs.

Die beschriebene prototypische Realisierung SECUBOOT für Ethernet-LANs unter NetWare zeigt die Praxistauglichkeit des Ansatzes.

6 Dank

Eine Reihe von Ideen zur Realisierung einer praxistauglichen Lösung des authentischen Bootens unter NetWare, viele hilfreiche Diskussionen und Unterstützung bei den erforderlichen Modifikationen des Boot-ROM-Codes verdanken wir Thomas Ruf und Stephan Wolf.

7 Literatur

[BrUn_93] *Braband, Jan; Ungerer, Bert*: Fernweh - Der holprige Weg zum Booten via NetWare-Server. magazin für computer technik, c't 3/1993, S. 186-188.

[DaPr_89] *Davies, Donald W.; Price, Wyn L.*: Security for Computer Networks. 2. Aufl., John Wiley & Sons Ltd., Chichester 1989.

[DiHe_76] *Diffie, Whitfield; Hellman, Martin E.*: New Directions in Cryptography. IEEE Transactions on Information Theory, Vol. IT-22, No. 6, 1976, S. 644-654.

[Fox_93] *Fox, Dirk*: Der Digital Signature Standard. Aufwand, Implementierung und Sicherheit. Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS '93, DuD-Fachberichte 16, Vieweg, Braunschweig 1993, S. 333-352.

[Fumy_93] *Fumy, Walter*: Designprinzipien für Authentifizierungsmechanismen. Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS '93, DuD-Fachberichte 16, Vieweg, Braunschweig 1993, S. 307-319.

[Groß_91] *Groß, Michael*: Vertrauenswürdiges Booten als Grundlage authentischer Basissysteme. Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS '91, Informatik-Fachberichte Nr. 271, Springer-Verlag, Heidelberg 1991, S. 190-207.

[ISO_87] *International Organisation for Standardization (ISO)*: Banking - Requirements for Message authentication (wholesale). International Standard ISO 8730, Genf 1987.

[ISO_89] *International Organisation for Standardization (ISO)*: Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture. International Standard ISO 7498-2 (E), Genf 1989.

[OsSa_93] *Osterlehner, Stefan; Sauerbrey, Jörg*: Authentisches Booten und Software-Integritätstest auf PC-Architekturen. Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS '93, DuD-Fachbeiträge 16, Vieweg, Braunschweig 1993, S. 321-331.

[Rive_92] *Rivest, Ronald L.*: The MD5 Message-Digest Algorithm. Request for Comments (RFC) 1321, Network Working Group, 4/1992, S. 1-21.

[RSA_78] *Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard*: A Method for obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, Vol. 21, No. 2, 1978, S. 120-126.