

Sicheres Schlüsselmanagement für verteilte Intrusion-Detection-Systeme

Michael Meier · Thomas Holz

Brandenburgische Technische Universität Cottbus
Institut für Informatik / Lehrstuhl Rechnernetze
Postfach 10 13 44
03013 Cottbus
{mm, thh}@informatik.tu-cottbus.de

Zusammenfassung

Verteilte Intrusion-Detection-Systeme (IDS) stellen ein zunehmend an Bedeutung gewinnendes Instrument für den Schutz informationstechnischer Ressourcen dar. Durch die permanente Verarbeitung von Audit-Datensätzen mit (partiell) benutzeridentifizierenden Informationen entsteht allerdings ein datenschutzrechtliches Problem. Eine prinzipielle Lösung dieses Problems ist möglich, indem Audit-Daten in pseudonymer Repräsentation ausgewertet werden. Dabei erfordert der Einsatz der für die Pseudonymisierung und ggf. Depseudonymisierung notwendigen kryptographischen Verfahren ein leistungsfähiges Schlüsselmanagement. Dieses muß nicht nur den konventionellen Anforderungen an ein Schlüsselmanagement genügen, sondern auch der speziellen Arbeitsweise eines verteilten, pseudonymisierte Audit-Datensätze verarbeitenden IDS gerecht werden. Es besteht in erster Linie die Notwendigkeit, den in der Regel hohen Datendurchsatz nicht wesentlich zu beeinflussen. Außerdem ist ein besonders guter Schutz archivierter Schlüssel erforderlich. Das im vorliegenden Beitrag beschriebene Konzept ist auf die Erfüllung der genannten Anforderungen ausgerichtet. Es sieht die Verwendung allgemeiner Prinzipien, wie z.B. mehrere Schlüsselhierarchien, konsequente Schlüsselseparierung und die Nutzung von Kryptoperioden, vor. Ausgehend vom Ziel des Konzepts, der Verwaltung der Pseudonymisierungs- und Depseudonymisierungsschlüssel, sind außerdem eine Reihe von speziellen Maßnahmen notwendig, um den spezifischen Gegebenheiten der für die prototypische Realisierung genutzten Systemumgebung zu entsprechen. Diese Umgebung besteht aus den Betriebssystemen Sun Solaris und Microsoft Windows NT sowie dem an der BTU Cottbus entwickelten Intrusion-Detection-System AID, das unter den genannten Betriebssystemen läuft und in das die Komponenten des Schlüsselmanagements integriert wurden. Der Beitrag schließt mit einer Bewertung der erzielten Testresultate.

1 Einleitung

Mit der wachsenden Bedeutung informationstechnischer Systeme in vielen Bereichen des gesellschaftlichen Lebens geht ein stetig zunehmendes Bedrohungspotential für diese Systeme einher [CSI+99]. Ein wirksamer Schutz der damit verbundenen Werte erfordert leistungsfähige, rechnerunterstützte Schutzmechanismen. In diesem Zusammenhang spielen Verfahren und Systeme der automatischen Erkennung von IT-Sicherheitsverletzungen eine wichtige Rolle. Derartige Systeme werden als *Intrusion-Detection-Systeme* (IDS) bezeichnet [Sund96].

Die meisten IDS führen eine automatische Analyse der *Audit-Daten* durch, die von den zu überwachenden IT-Systemen generiert werden. Audit-Daten sind möglichst detaillierte Auf-

zeichnungen aller sicherheitsrelevanten Aktivitäten. Die Auswertung derartiger Daten stellt häufig die einzige Möglichkeit dar, IT-Sicherheitsverletzungen zu erkennen, deren Ausmaß abzuschätzen, potentielle Angreifer zurückzuverfolgen sowie geeignete Gegenmaßnahmen einzuleiten [MuHL94].

Aufgrund des verteilten Charakters und der hohen zeitlichen Dynamik ernstzunehmender Sicherheitsverletzungen stehen Intrusion-Detection-Systeme vor großen Herausforderungen. In vielen Einsatzfällen müssen die Audit-Datensätze eines gesamten lokalen Netzwerkes ohne signifikante Zeitverzögerung zentral gesammelt und analysiert werden. IDS, die diesen Anforderungen gerecht werden sollen, müssen somit über eine verteilte Architektur leistungsstarker Systemkomponenten verfügen. Häufig ist ein zentralistisch-verteilter Ansatz anzutreffen, der aus einer zentralen Auswertungseinheit und Monitoring-Agenten auf jedem überwachten Rechner besteht [MuHL94]. Während die Monitoring-Agenten für die Erfassung, die Vorverarbeitung und den Transfer der anfallenden Audit-Daten zuständig sind, besteht die Aufgabe der Auswertungseinheit in der schnellstmöglichen Analyse des eintreffenden Datenstroms. Dies kann beispielsweise mit Hilfe eines Echtzeit-Expertensystems erfolgen.

Neben ihrer Bedeutung für den Schutz informationstechnischer Ressourcen weist die Verarbeitung von Audit-Daten jedoch auch einen eher negativ zu bewertenden Aspekt auf: Zur Gewährleistung der Zurechenbarkeit enthalten die entsprechenden Datensätze benutzeridentifizierende Informationen, wodurch ein datenschutzrechtliches Problem entsteht. Eine sozial und rechtlich akzeptable Lösung dieses Problems bietet *Pseudonymes Audit* [SoFR97]. Hierbei werden benutzeridentifizierende Informationen durch Pseudonyme ersetzt, deren Generierung mit Hilfe kryptographischer Verfahren erfolgt. Eine Depseudonymisierung der Daten findet nur bei Vorliegen berechtigter Verdachtsmomente statt, d.h., wenn die Ergebnisse der Audit-Analyse auf IT-Sicherheitsverletzungen hindeuten. Die Bereitstellung sicherer Mechanismen für die Verwaltung der im Verlauf der Pseudonymisierung und Depseudonymisierung verwendeten Schlüssel ist Aufgabe eines geeigneten *Schlüsselmanagements*.

Verteilte Intrusion-Detection-Systeme stellen aufgrund ihrer Arbeitsweise einen besonderen Anwendungsfall für Mechanismen und Verfahren des Schlüsselmanagements dar. Bisher sind keine Arbeiten bekannt, die sich diesem Anwendungsfall widmen. Wesentliche Schwerpunkte der eigenen Untersuchungen waren daher:

- die Analyse der Anforderungen an ein Schlüsselmanagement für verteilte IDS,
- die Herausarbeitung notwendiger Aufgaben des Schlüsselmanagements,
- die Auswahl geeigneter Mechanismen zur Umsetzung der Aufgaben sowie
- die Erstellung eines Gesamtkonzepts, das den Anforderungen des hier vorliegenden Anwendungsfalls entspricht.

Das entwickelte Konzept wurde in eine prototypische Implementation überführt. Als Entwicklungs- und Testplattform diente das an der BTU Cottbus entwickelte Intrusion-Detection-System AID (Adaptive Intrusion Detection system [SoRK96]). Als besonders bedeutsam erwies sich dabei eine Analyse der Realisierungsmöglichkeiten von Teillösungen des Schlüsselmanagements auf Basis der Betriebssysteme Sun Solaris und Microsoft Windows NT. Diese beiden Betriebssysteme werden gegenwärtig von AID unterstützt.

In den folgenden Abschnitten werden die Ergebnisse der genannten Arbeitsschwerpunkte ausführlicher dargestellt.

2 Verteilte IDS und Pseudonymes Audit

In diesem Abschnitt wird zunächst die grundlegende Architektur eines verteilten IDS erläutert. Diese Beschreibung erfolgt exemplarisch für das der prototypischen Realisierung des Schlüsselmanagements zugrundeliegende Intrusion-Detection-System AID. Außerdem wird die Funktionsweise der zur Realisierung des Pseudonymen Audit in AID integrierten kryptographischen Funktionseinheiten skizziert.

2.1 Architektur und Funktionsweise von AID

AID ist ein zentralistisch verteiltes IDS. Es weist eine Client-Server-Architektur auf, die im wesentlichen aus einer zentralen Auswertungseinheit (Client) und Agenten (Server) auf den zu überwachenden Rechnern besteht (vgl. Abbildung 1). Die Agenten erfassen die auf den Rechnern anfallenden Audit-Daten, führen eine Vorverarbeitung durch und transferieren die Daten zur zentralen Auswertungseinheit. Der Datentransfer erfolgt über ein RPC-Interface (Remote Procedure Call).

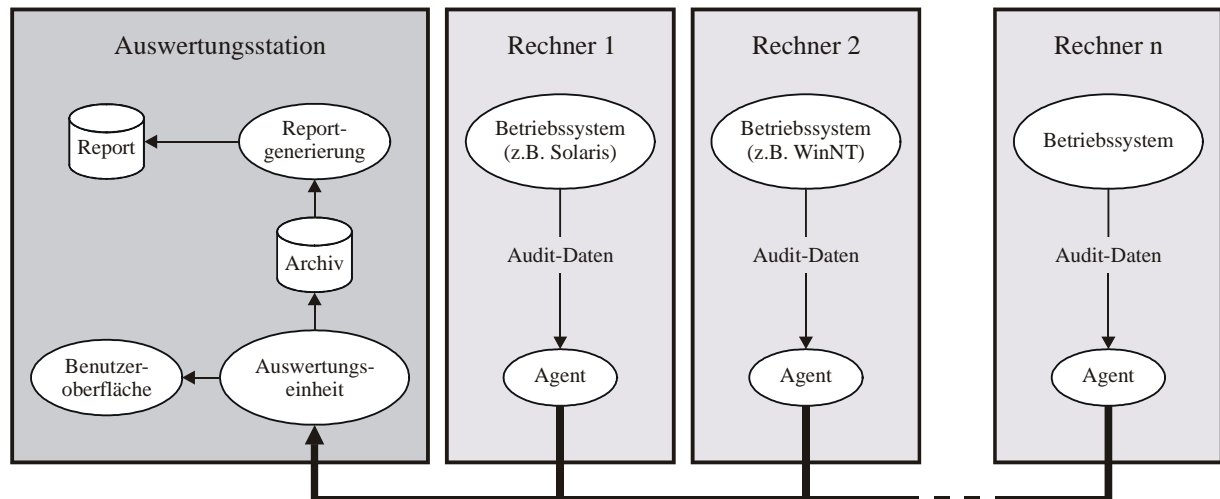


Abbildung 1: Architektur von AID

Die Analyse der Audit-Daten erfolgt mit Hilfe eines echtzeitfähigen Expertensystems, das auf der Basis von RTworks entwickelt wurde. Es verwendet eine Wissensbasis mit zustandsorientiert modellierten Signaturen, die als Regeln implementiert sind.

Durch das System erkannte Sicherheitsverletzungen werden neben anderen sicherheitsrelevanten Informationen sowohl auf einer graphischen Benutzeroberfläche dargestellt als auch protokollarisch archiviert. Mit diesen Archiven stehen wertvolle Informationen für weitergehende Untersuchungen zur Verfügung. Unter Verwendung eines speziell für diesen Zweck entwickelten Programms besteht die Möglichkeit, aus den archivierten Daten Sicherheitsreports zu generieren.

2.2 Pseudonymes Audit mit AID

Das Konzept des Pseudonymen Audit sieht vor, Audit-Daten direkt nach ihrer Generierung zu pseudonymisieren und in dieser Form zu analysieren. Eine Depseudonymisierung der Daten erfolgt nur bei Vorliegen berechtigter Verdachtsmomente, d.h., wenn die Ergebnisse der Analyse der Audit-Daten auf Sicherheitsverletzungen hindeuten. Abbildung 2 veranschaulicht, in welchen Repräsentationen Audit- und Ergebnisdaten in den einzelnen Modulen von AID vorliegen und durch welche kryptographischen Funktionseinheiten die Daten verarbeitet werden.

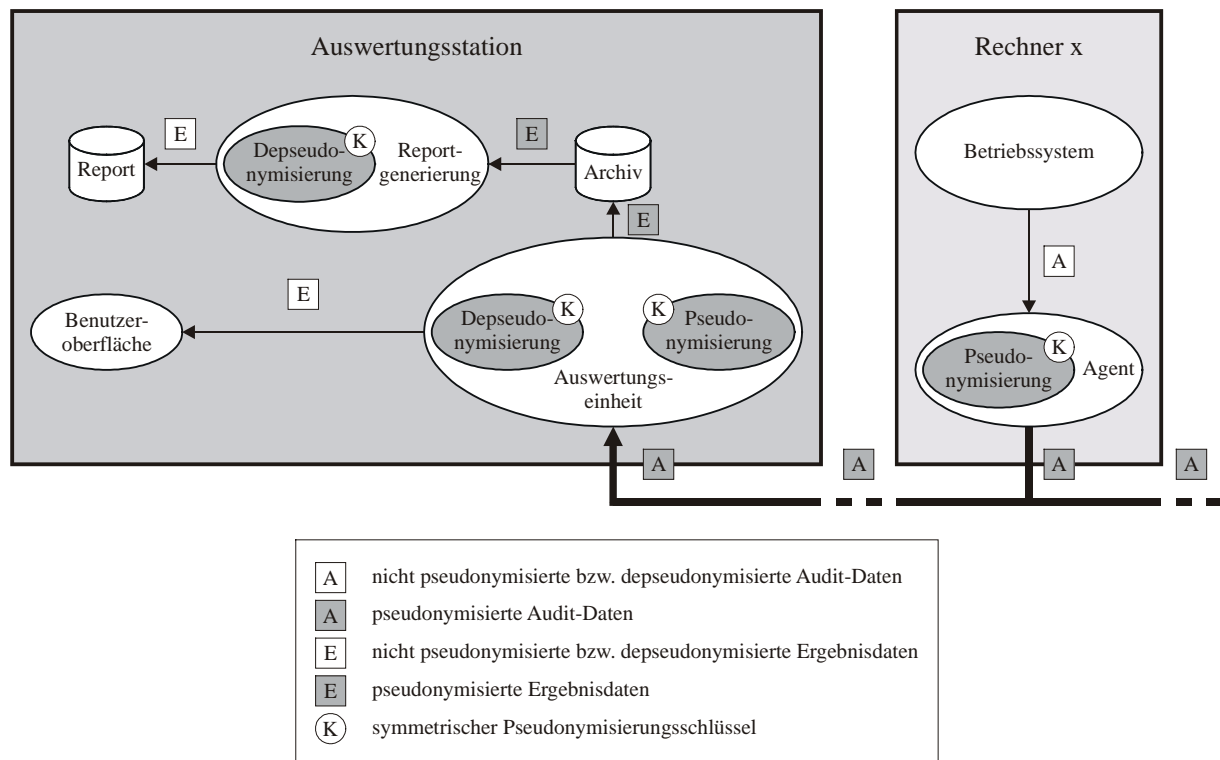


Abbildung 2: Kryptographische Funktionseinheiten in AID

Die Pseudonymisierung der Audit-Daten in AID erfolgt durch die Monitoring-Agenten auf den überwachten Rechnern. Diese ersetzen in Audit-Daten enthaltene, benutzeridentifizierende Informationen durch Pseudonyme, die unter Verwendung symmetrischer Verschlüsselungsverfahren generiert werden.

Audit-Daten werden in pseudonymer Repräsentation analysiert. Deuten die Ergebnisse der Analyse auf Sicherheitsverletzungen hin, werden entsprechende Meldungen auf der graphischen Oberfläche ausgegeben. Darin enthaltene benutzeridentifizierende Informationen werden automatisch depseudonymisiert.

Die Mehrzahl der Signaturen, die der Analyse des Expertensystems zugrunde liegen, kann unabhängig von den als Initiator infrage kommenden Benutzern modelliert werden. Einzelne Angriffssignaturen nehmen jedoch Bezug auf die Identität eines Benutzers, um bspw. Zugriffe des Systemadministrators zu modellieren. Um diese Zugriffe erkennen zu können, muß Bezug auf das aktuelle Pseudonym des Systemadministrators genommen werden. Zu diesem Zweck

verfügt das Expertensystem neben den Funktionen zur Depseudonymisierung auch über Pseudonymisierungsfunktionalität, mit der es in der Lage ist, das aktuelle Pseudonym eines Benutzers zu ermitteln.

In den von der Auswertungseinheit erstellten Archiven werden die Ergebnisdaten in pseudonymer Repräsentation abgespeichert. Autorisierte Benutzer haben die Möglichkeit, unter Verwendung des Reportgenerators Sicherheitsreports aus den archivierten Daten zu erstellen. Hierbei werden die Ergebnisdaten einer Depseudonymisierung unterzogen.

3 Anforderungen an das Schlüsselmanagement

Für ein Schlüsselmanagement, das als Bestandteil des Konzepts des Pseudonymen Audit innerhalb eines verteilten IDS eingesetzt werden soll, bestehen zunächst einmal Anforderungen, die auch in den meisten anderen Anwendungsgebieten von Schlüsselmanagement-Technologien anzutreffen sind. Hierzu zählen neben dem Schutz der Vertraulichkeit geheimer und der Gewährleistung der Integrität geheimer und öffentlicher Schlüssel in allen Phasen der Verarbeitung insbesondere die notwendigen Eigenschaften der Protokolle zur Schlüsselvereinbarung und -verteilung. Derartige Eigenschaften sind u.a. die Gewährleistung von Perfect Forward Secrecy und die Resistenz gegenüber "Known Key"-Attacken. Ein Protokoll garantiert Perfect Forward Secrecy, wenn durch die Kompromittierung von Langzeitschlüsseln ehemalige Sitzungsschlüssel nicht kompromittiert werden (vgl. [Günt90, MeOV97]). Resistenz hinsichtlich "Known Key"-Attacken bedeutet, daß die Kompromittierung ehemaliger Sitzungsschlüssel die Vertraulichkeit gegenwärtiger oder zukünftiger Sitzungsschlüssel nicht gefährdet (vgl. [YaSh90, MeOV97]).

Die hier vorliegenden anwendungsspezifischen Anforderungen an das Schlüsselmanagement sind ursächlich im Charakter der Datenverarbeitung innerhalb eines verteilten IDS begründet. Da derartige Systeme auf eine kontinuierliche Verarbeitung eines hohen Datenaufkommens ausgelegt sein müssen, unterliegt die Umsetzung der Aufgaben des Schlüsselmanagements hohen zeitlichen Restriktionen. Der Einsatz von für die Realisierung des Schlüsselmanagements notwendigen Hard- oder Softwarekomponenten darf keinesfalls zu einer signifikanten zeitlichen Verzögerung der eigentlichen Tätigkeit eines IDS, d.h., der automatischen Einbruchserkennung und ggf. Initiierung von Gegenmaßnahmen, führen.

Ein weiteres spezifisches Problem besteht in der Notwendigkeit des besonders wirksamen Schutzes der Integrität archivierter Schlüssel. Damit Intrusion-Detection-Systeme unwiderlegbare Nachweise zielgerichteter IT-Sicherheitsverletzungen liefern können, müssen sie in der Lage sein, zeitlich weit auseinanderliegende Einzelereignisse komplexeren Sicherheitsverletzungen zuzuordnen. Dies kann z.B. auch im Verlauf nachträglicher Auswertungen unter Berücksichtigung neuer Angriffsmuster notwendig sein. Da benutzeridentifizierende Informationen in den entsprechenden Audit-Datensätzen in pseudonymer Form vorliegen, genügt es zur Aufrechterhaltung der Zurechenbarkeit nicht, Manipulationen an den verwendeten Schlüsseln lediglich zu erkennen. Derartige Manipulationen müssen vielmehr so wirksam wie möglich verhindert werden, da andernfalls in der Regel keine Depseudonymisierungen mehr möglich sind. Die Veränderung der Pseudonymisierungs- und Depseudonymisierungsschlüssel stellt für einen Angreifer einen effektiven Weg dar, seine Identität zu verbergen. Die konkrete Realisierung der Archivierung dieser Schlüssel erweist sich somit als eine äußerst wichtige und anspruchsvolle Aufgabe im Kontext der Verarbeitung pseudonymisierter Audit-Daten.

4 Übersicht des erstellten Konzepts

Um den genannten Anforderungen im Kontext eines verteilten, partiell pseudonymisierte Audit-Datensätze verarbeitenden Intrusion-Detection-Systems gerecht werden zu können, sind für das gewünschte Schlüsselmanagement eine Reihe von Aufgaben zu erfüllen, die sich zu einem Gesamtkonzept zusammenfügen.

Bevor nun eine Darstellung der wichtigsten Bestandteile der vorgesehenen Schlüsselhierarchien erfolgt, sollen zwei allgemeine Prinzipien genannt werden, die für jeden der verwendeten Schlüssel wirksam sind. Das sind einerseits das Prinzip der Schlüsselseparierung und andererseits die Nutzung von Kryptoperioden. Schlüsselseparierung bedeutet, daß jeder Schlüssel nur für einen Zweck verwendet wird. Kryptoperioden beschreiben Zeitintervalle, in denen Schlüssel als gültig angesehen werden und nach deren Ablauf eine Schlüsselaktualisierung notwendig ist. Beide Mechanismen sollen die Auswirkungen möglicher Schlüsselkompromittierungen minimieren.

Im folgenden werden zunächst im Kontext des Pseudonymen Audit notwendige Schlüsselmanagementaufgaben aufgezeigt, bevor Mechanismen zu ihrer Umsetzung diskutiert werden.

4.1 Notwendige Schlüsselmanagementaufgaben

Aus der Architektur eines verteilten IDS, den verwendeten Konzepten und dem Lebenszyklus der verwendeten Schlüssel ergeben sich folgende Schlüsselmanagementaufgaben:

- Schlüsselgenerierung,
- Schlüsselverteilung,
- Schlüsselverifikation,
- Überwachung von Kryptoperioden,
- Schlüsselaktualisierung,
- Schlüsselarchivierung,
- sichere Schlüsselspeicherung in Dateien,
- sichere Aufbewahrung der Schlüssel im Speicher und
- sicheres Löschen von Schlüsseln.

4.2 Mechanismen zur Umsetzung der Aufgaben

Die Generierung der zur Pseudonymisierung und Depseudonymisierung eingesetzten Schlüssel erfolgt durch die zentrale Auswertungseinheit des IDS. Sie werden an die einzelnen Agenten verteilt und ebenso wie die Audit-Daten in Dateien archiviert, um für nachträglich erfolgende Analysen zur Verfügung zu stehen. Zur Schlüsselgenerierung finden die Funktionen der jeweils verwendeten Kryptobibliotheken Anwendung.

Die Verteilung der Schlüssel erfolgt unter Verwendung eines Sitzungsschlüssels, dessen Etablierung mit Hilfe des Protokolls SKEME (Secure Key Exchange Mechanism) im Basic Mode (vgl. [Kraw96]) vorgenommen wird. SKEME im Basic Mode gewährleistet Perfect Forward Secrecy und ist resistent gegen „Known Key“-Attacken. Eine kurze Beschreibung des Protokolls erfolgt im Anhang. Es wurde hier gewählt, da es aufgrund seines modularen Aufbaus vielseitig einsetzbar ist und verschiedene Kryptoverfahren leicht ausgetauscht werden können.

Zur Verteilung der Pseudonymisierungsschlüssel führt die Auswertungseinheit des IDS nacheinander mit jedem Agenten das Schlüsseletablierungsprotokoll durch, um den jeweils benötigten Sitzungsschlüssel zu etablieren.

Voraussetzung für die Verwendung von SKEME ist, daß jede kommunizierende Systemkomponente (Auswertungseinheit und Agenten) ein asymmetrisches Schlüsselpaar besitzt und über authentische Kopien der öffentlichen Schlüssel der Kommunikationspartner verfügt. Dieser Sachverhalt wird hier realisiert. Der initiale Austausch bzw. die Verifikation öffentlicher Schlüssel erfolgen dabei manuell. Asymmetrische Schlüssel werden, ebenso wie jene für die Pseudonymisierung und Depseudonymisierung, in Dateien gespeichert.

Alle öffentlichen und geheimen Schlüssel müssen während ihrer Verweildauer im Intrusion-Detection-System möglichst zuverlässig vor Manipulationen, geheime Schlüssel zusätzlich vor unbefugter Einsichtnahme geschützt werden. Dies betrifft sowohl den Aufenthalt der Schlüssel im Hauptspeicher als auch den Aufenthalt in den dafür vorgesehenen Dateien. Im Hauptspeicher, d.h. während der Verwendung durch die Systemkomponenten, sind Integrität und Vertraulichkeit der Schlüssel relativ stark gefährdet. Daher ist die Dauer des Aufenthalts im Hauptspeicher zu minimieren. Nach der Verwendung erfolgt ein Überschreiben der zuvor von den Schlüsseln belegten Speicherbereiche. Einzelne Schlüssel, z.B. Pseudonymisierungsschlüssel, werden jedoch kontinuierlich verwendet und liegen daher zur Laufzeit des IDS permanent unverschlüsselt im Speicher. Dieser Umstand stellt einen prinzipiellen Schwachpunkt in der praktischen Realisierung des Schlüsselmanagements dar, denn ein permanentes Aus- und Einlagern der Schlüssel würde das verteilte IDS massiv verlangsamen.

Die Vertraulichkeit von Hauptspeicherseiten ist in modernen Betriebssystemen zwei weiteren Gefährdungen ausgesetzt. Dabei handelt es sich um die Erzeugung von Speicherabbildern (Core bzw. Memory Dumps) in Ausnahmesituationen sowie die Auslagerung von Seiten in eine Swap-Partition bzw. Auslagerungsdatei. Beide Situationen müssen nach Möglichkeit unterbunden werden, indem durch eine entsprechende Konfigurierung des Betriebssystems die Erzeugung von Speicherabbildern nach Laufzeitfehlern und die Auslagerung vertraulicher Speicherseiten verhindert werden. Dies wird im realisierten Konzept umgesetzt, wobei die Systemkomponenten für Solaris die für Core-Dumps mögliche Größe auf 0 setzen und für spezielle Speicherbereiche die Systemfunktionen `mlock()` verwenden [Stev95]. Die Windows-NT-Komponenten verwenden entsprechend die Funktion `VirtualLock()` [Micr99].

Die Speicherung der beschriebenen Schlüsselarten in Dateien erfolgt teilweise verschlüsselt. Zur Verschlüsselung geheimer Schlüssel wird eine Paßphrase verwendet. Des weiteren wird für jede Schlüsseldatei ein MAC (Message Authentication Code) berechnet und gespeichert. Der hierfür verwendete geheime Schlüssel wird ebenfalls in der Datei abgelegt. Die Festlegung von Paßphrasen sowie die Generierung von MAC-Schlüsseln erfolgt bei der Initialisierung des Systems.

Inhalte von Schlüsseldateien werden zweifach vor unautorisierten Zugriffen gesichert. Einerseits erfolgt die Nutzung der Zugriffskontrollmechanismen des jeweiligen Betriebssystems, andererseits werden geeignete Synchronisationskonzepte angewandt. Diese Synchronisationskonzepte sorgen auch im Fall der Überwindung der Zugriffskontrolle, z.B. durch einen erfolgreichen Angreifer, für den Schutz der Dateiinhalte. Beispiele sind obligatorische Datensatzsperrungen in UNIX (vgl. [Stev95]) oder die Verwendung eines geeigneten File-Sharing-Mechanismus' in Windows NT (vgl. [Micr99]). Beide Maßnahmen sorgen dafür, daß lediglich der

sperrende Prozeß (einer der Prozesse der Auswertungseinheit oder der Agenten) den gewünschten Zugriff auf die entsprechende Datei hat. Die Verwendung dieser Mechanismen erfolgt in Solaris über die Funktion `fcntl()` [Stev95] und in Windows NT mit der Funktion `LockFile()` [Micr99].

Sollte doch einmal der Fall eintreten, daß eine Manipulation gespeicherter Schlüssel festgestellt wird, greift der Mechanismus der verteilten Kopien des Schlüsselarchivs. Das verteilte IDS wird dahingehend erweitert, daß neben der Auswertungseinheit jeder Agent Schlüsselarchive verwalten kann.

Das Löschen von Schlüsseldateien erfolgt, indem die Dateien mehrfach mit den in [Gutm96] für diesen Zweck vorgeschlagenen Mustern überschrieben werden.

6 Prototypische Realisierung

Das hier in seiner wesentlichen Struktur beschriebene Schlüsselmanagementkonzept kann sowohl in eine alleinstehende verteilte Testapplikation als auch in eine Erweiterung bereits existierender Intrusion-Detection-Systeme überführt werden. Da an der BTU Cottbus in Form von AID bereits ein lauffähiges, für heterogene lokale Netze konzipiertes IDS zur Verfügung steht, wurde dieses als Integrationsplattform der prototypischen Bausteine des Schlüsselmanagements verwendet.

Für eine Implementierung von Teilkomponenten des Konzepts kommen verschiedene Verfahren in Betracht. Hier wurde auf die beiden, zum Zeitpunkt der Implementierung verfügbaren Kryptobibliotheken LiSA (Library for Secure Applications, vgl. [BKM+96]) und `crypto++` (vgl. [Wei98]) zurückgegriffen. Im Rahmen des Prototyps kommen dann u.a. die folgenden Kryptoprimitive zum Einsatz:

- RSA mit 2048 Bit Schlüssellänge (zum Schlüsseltransport bei der Schlüsseletablierung),
- das Diffie-Hellman-Verfahren mit 2048 Bit Schlüssellänge (innerhalb des Schlüsseletablierungsprotokolls)
- MD5 (für die Durchführung des Schlüsseletablierungsprotokolls),
- ein CBC-MAC auf der Basis von IDEA im CBC-Mode (zur Durchführung der Schlüsseletablierung und zur Überprüfung der Integrität von Schlüsseldateien) und
- IDEA mit 128 Bit Schlüssellänge (zur Verschlüsselung von Schlüsseln).

Zum Zeitpunkt der Implementierung war AID in der Lage, Rechner zu überwachen, die entweder unter dem Betriebssystem Solaris oder dem Betriebssystem Windows NT laufen. Dementsprechend wurden die genannten Mechanismen auf diese beiden Systeme angepaßt, um eine Integration in AID zu ermöglichen. Eine zuvor durchgeführte, detaillierte Untersuchung der Sicherheitsarchitekturen dieser Betriebssysteme verdeutlichte, daß diverse Schwachstellen den Schutz der verwalteten Schlüssel erheblich beeinträchtigen. So ist z.B. die Vertraulichkeit geheimer Schlüssel in Hauptspeicherbereichen in der Regel nur durch die Zugriffskontrolle der Betriebssysteme geschützt. Diese Zugriffskontrolle ist jedoch, wie zahlreiche Untersuchungen sowie nahezu täglich im Internet veröffentlichte Nachrichten belegen, teilweise überwindbar (z.B. durch Ausnutzen von Debugging-Features).

Allgemein kann festgehalten werden, daß schon aufgrund der Existenz eines Superusers (UNIX) bzw. Administrators (Windows NT) sowie der fast immer eingesetzten, paßwortbasierten Authentifikation durch die Betriebssysteme der ausschließlich durch die Zugriffskon-

trolle erfolgende Schutz von Systemressourcen als unzureichend zu bewerten ist. Daher sind die für Verschlüsselungen definierten Schlüssellängen streng genommen uninteressant, da es für einen Angreifer fast immer relativ einfach ist, das Superuser- bzw. Administrator-Paßwort zu brechen. In Solaris ist z.B. ein Angreifer nach Erlangen der Zugriffsrechte des Superusers in der Lage, unter Verwendung des Systemprogramms `gcore` explizit Speicherabbilder von Prozessen zu erstellen und aus diesen die verwendeten Schlüssel zu extrahieren.

6 Ergebnisse und Bewertung

Wird die auch für etliche andere Operationen notwendige Anlaufphase eines verteilten IDS vernachlässigt, so ist im Zusammenhang mit der nach Möglichkeit in Echtzeit vorzunehmenden Erkennung von IT-Sicherheitsverletzungen insbesondere der durch automatische Schlüsselaktualisierungen verursachte Mehraufwand von Bedeutung. Die hierzu erforderliche Rechenzeit wird einerseits bei der Generierung neuer Schlüssel und andererseits für die Verteilung dieser Schlüssel verbraucht. Zur Abschätzung der damit einhergehenden Leistungseinträchtigung wurden mit Hilfe des um die prototypischen Komponenten des Schlüsselmanagements erweiterten Intrusion-Detection-Systems AID Messungen vorgenommen. Diese Messungen erfolgten sowohl unter Solaris als auch unter Windows NT.

Es zeigte sich, daß die zur Schlüsselgenerierung notwendigen Prozessorzeiten im Vergleich zu den Prozessorzeiten, die für die Schlüsselverteilung erforderlich sind, vernachlässigt werden können. Die durchschnittliche Zeit, die die Auswertungseinheit oder ein Agent in der genutzten Systemplattform zur Schlüsselverteilung benötigt, beträgt ca. zehn Sekunden. In einer überwachten Umgebung mit einer Auswertungseinheit und einem Agenten bedeutet dies, daß die Audit-Analyse durch eine Aktualisierung der Pseudonymisierungsschlüssel um ca. 20 Sekunden unterbrochen wird. In einer Umgebung mit einer Auswertungseinheit und zehn Agenten wird dieser Vorgang zehnmal wiederholt, die entstehende Verzögerung beträgt dann ca. 200 Sekunden.

Im Kontext der automatischen Erkennung von IT-Sicherheitsverletzungen stellen Systembeeinträchtigungen dieser Größenordnung ein signifikantes Problem dar, da sich die Zeitspanne zwischen dem Auftreten einer Sicherheitsverletzung und deren Erkennung erheblich vergrößert. Besonders problematisch ist diese Situation in Umgebungen mit vielen Agenten oder bei der Definition relativ kurzer Kryptoperioden.

Das Ergebnis ist jedoch wenig überraschend, wenn berücksichtigt wird, daß die Funktionen des implementierten Schlüsselmanagements nicht innerhalb separater Threads ausgeführt werden, keine Kryptohardware zum Einsatz kam, und auch keine hinreichend effizient (z.B. in optimiertem Assembler) programmierten Kryptobibliotheken zur Verfügung standen. Die notwendige Verringerung der Beeinträchtigung des laufenden IDS muß daher durch die nach Möglichkeit kombinierte Realisierung der folgenden Ansätze erzielt werden:

- Die verwendeten kryptographischen Algorithmen müssen durch effiziente Implementierungen umgesetzt werden. Noch besser wäre der Einsatz von Spezialhardware.
- In der Auswertungseinheit und den Agenten sollte eine Verteilung der Verarbeitung der Audit-Daten und der Schlüsselverteilung auf verschiedene Prozesse oder Threads erfolgen, so daß mit der Audit-Datenverarbeitung fortgefahren werden kann, während auf die Berechnung der Protokollnachrichten durch den jeweiligen Kommunikationspartner gewartet wird.

- Es sind nach Möglichkeit Mehrprozessorsysteme zu verwenden, damit Audit-Analyse und Schlüsselverteilung parallel erfolgen können.
- Die Initiierung der Schlüsselverteilung kann auch auf einen separaten Rechner ausgelagert werden. In diesem Fall besteht die Aufgabe sowohl der Auswertungseinheit als auch der Agenten lediglich in der Verarbeitung der Protokollnachrichten zum Empfang der Schlüssel.

Nach der Umsetzung dieser Maßnahmen kann mit einer drastischen Verbesserung der Effizienz des laufenden Intrusion-Detection-Systems gerechnet werden.

Die Ergebnisse der durchgeführten Arbeiten haben verdeutlicht, daß die Konzipierung und Realisierung eines Schlüsselmanagements für verteilte Intrusion-Detection-Systeme eine sehr anspruchsvolle Aufgabe darstellt. Es wurde gezeigt, daß der Entwurf der erforderlichen Komponenten prinzipiell möglich ist, die konkrete technische Umsetzung jedoch diffizile Probleme aufwirft. Der Schutz der Schlüssel erweist sich aufgrund bestimmter, aus Sicht des Schlüsselmanagements ungeeigneter Konzepte der betrachteten Betriebssysteme als schwierig. Eine besondere Herausforderung ist außerdem das Finden eines praktikablen Kompromisses zwischen der möglichst schnellen, häufig Echtzeitanforderungen unterliegenden Analyse der Audit-Daten einerseits und dem Rechenaufwand, der zur Gewährleistung einer hinreichend sicheren Schlüsselverteilung erforderlich ist, andererseits.

A SKEME Basic Mode

Im folgenden wird das Schlüsseletablierungsprotokoll SKEME im Basic Mode kurz vorgestellt (ausführlich siehe [Kraw96]). Zunächst werden im Protokoll verwendeten kryptographischen Primitive und die zur Beschreibung benutzte Notation vorgestellt.

SKEME verwendet ein asymm. Verschlüsselungsverfahren. Mit $E_{A_{pub}}(X)$ wird die Verschlüsselung der Nachricht X unter Verwendung des öffentlichen Schlüssels der Partei A bezeichnet. Das Protokoll basiert auf dem Schlüsselvereinbarungsprotokoll von Diffie und Hellman. Der öffentliche Diffie-Hellman-Schlüssel (DH-Schlüssel) einer Partei wird mit $g^x \bmod m$ dargestellt. Außerdem kommt ein MAC zum Einsatz. $MAC_K(X)$ bezeichnet den unter Verwendung von Schlüssel K über die Nachricht X erstellten MAC. Mit $Hash(X,Y)$ wird der Hash-Wert über die Konkatenation der Nachrichten X und Y ausgedrückt.

Das Protokoll besteht aus den drei Phasen SHARE, EXCH und AUTH. Voraussetzung für die Verwendung des Protokolls ist, daß die Kommunikationspartner A und B jeweils über ein asymm. Schlüsselpaar verfügen, und jede Partei eine authentische Kopie des öffentlichen Schlüssels der anderen Partei besitzt.

Die SHARE-Phase dient der Etablierung eines gemeinsamen geheimen Schlüssels für die Parteien A und B . Partei A erzeugt einen symm. Schlüssel (K_A) und sendet ihn verschlüsselt mit dem öffentlichen Schlüssel B_{pub} an B (siehe Tabelle 1). Nach Empfang der Nachricht erzeugt B einen symm. Schlüssel K_B und sendet ihn verschlüsselt mit A_{pub} an A . Nach Empfang der Nachrichten berechnen beide Parteien den gemeinsamen geheimen Schlüssel K_0 .

Tabelle 1: SHARE-Phase

A	B
$E_{B_{pub}}(K_A)$	→
←	$E_{A_{pub}}(K_B)$
$K_0 = Hash(K_A, K_B)$	

In der EXCH-Phase erfolgt der Austausch der öffentlichen DH-Schlüssel. Dazu erzeugt jede Partei ein DH-Schlüsselpaar und sendet den öffentlichen Schlüssel an den Kommunikationspartner (siehe Tabelle 2).

Tabelle 2: EXCH-Phase

A	B
$g^x \bmod m$	→
←	$g^y \bmod m$

Der etablierte Schlüssel K_0 wird in der AUTH-Phase zur Authentifikation der öffentlichen DH-Schlüssel verwendet. Jede Partei erzeugt einen MAC über ihren eigenen öffentlichen DH-Schlüssel, den des Kommunikationspartners sowie über den eigenen Identifikator und den des Kommunikationspartners (siehe Tabelle 3). Zur Berechnung des MAC wird der Schlüssel K_0 verwendet. Nach Empfang des MAC wird er durch erneute Berechnung überprüft. Durch dieses Vorgehen werden beide Parteien und die ausgetauschten DH-Schlüssel authentifiziert.

Tabelle 3: AUTH-Phase

A	B
$MAC_{K_0}(g^y, g^x, id_A, id_B)$	→
←	$MAC_{K_0}(g^x, g^y, id_B, id_A)$

Die einzelnen Protokollphasen können kombiniert werden, so daß für das Protokoll nur drei Nachrichten benötigt werden (siehe Tabelle 4). Der symm. Sitzungsschlüssel S_K wird im Anschluß an das Protokoll als Hash-Wert des gemeinsamen DH-Schlüssels berechnet.

Tabelle 4: Protokollnachrichten

A	B
$E_{B_{pub}}(id_A, K_A), g^x$	→
←	$E_{B_{pub}}(K_B), g^y, MAC_{K_0}(g^x, g^y, id_B, id_A)$
$MAC_{K_0}(g^y, g^x, id_A, id_B)$	→
$SK = Hash(g^{xy} \bmod m)$	

Literatur

- [BKM+96] Biehl, I.; Kenn, H.; Meyer, B.; Müller, B.; Schwarz, J.; Thiel, Ch.: LiSA - Eine C++-Bibliothek für kryptographische Verfahren. In: Digitale Signaturen, Vieweg, 1996, S. 237-248.
- [CSI+99] Computer Security Institute; Federal Bureau of Investigation: Computer Crime and Security Survey. 1999 - URL: <http://www.gocsi.com/prelea990301.htm>.
- [Günt90] Günther, C. G.: An identity-based key-exchange protocol. In: Advances in Cryptology - EUROCRYPT'89 (LNCS 434), 1990, S. 29-37.
- [Gutm96] Gutmann, P.: Secure Deletion of Data from Magnetic and Solid-State Memory. In: Proc. of the sixth USENIX Security Symposium, 1996.
- [Kraw96] Krawczyk, H.: SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In: Proc. of the 1996 Symposium on Network and Distributed System Security, San Diego, 1996.
- [MeOV97] Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A.: Handbook of applied cryptography. CRC Press, Boca Raton, 1997.
- [Micr99] Microsoft Corporation: MSDN Library - July 1999, 3 CDs.
- [MuHL94] Mukherjee, B.; Heberlein, L. T.; Levitt, K. N.: Network Intrusion Detection. In: IEEE Network 8 (1994), Nr. 3, S. 26-41.
- [SoRK96] Sobirey, M.; Richter, B.; König H.: The Intrusion Detection System AID - Architecture, and experiences in automated audit analysis. In: Horster P. (ed.): Communications and Multimedia Security II, Proc. of the IFIP TC6 / TC11 International Conference on Communications and Multimedia Security (CMS'96), Essen, Germany, 1996, Chapman & Hall, London, S. 278-290.
- [SoFR97] Sobirey, M.; Fischer-Hübner, S.; Rannenber, K.: Pseudonymous Audit for Privacy Enhanced Intrusion Detection. In: Yngström, L.; Carlsen, J. (eds.): Information Security in Research and Business, Proc. of the 13th International Information Security Conference (SEC'97), Copenhagen, Denmark, May 1997, Chapman & Hall, London, S. 151-163.
- [Stev95] Stevens, W. R.: Programmierung in der Unix-Umgebung: Die Referenz für Fortgeschrittene. Addison-Wesley, Bonn, 1995.
- [Sund96] Sundaram, A.: An Introduction to Intrusion Detection. In: ACM Crossroads Issue 2.4, 1996 - URL: <http://www.acm.org/crossroads/xrds2-4/>.
- [Wei98] Wei Dai: crypto++ - A C++ Class Library of Cryptographic Primitives, 1998 - URL: <http://www.eskimo.com/~weidai/cryptlib.html>.
- [YaSh90] Yakobi, Y.; Shmueli, Z.: On key distribution systems. In: Advances of Cryptology - CRYPTO'89 (LNCS 435), 1990, S. 344-355.