Ausarbeitung für Hauptseminar 'Technischer Datenschutz': Untrusted Hosts and Confidentiality

Josef Spillner

Oktober 2003

Contents

1 Einleitung **3**

2 Beschreibung der Methode **3**

2.1 Verteilung 4

2.2 Statisches Sicherheitsmodell 4

2.3 Dynamisches Sicherheitsmodell 5

3 Implementierung **6**

4 Praxisrelevanz **6**

4.1 Praxisfall P3P 7

4.2 Praxisfall Battleship 7

4.3 Elektronische Wahlen 7

5 Einschränkungen **8**

5.1 Angriffsmöglichkeiten 8

6 Materialien **9**

Ausarbeitung für Hauptseminar 'Technischer Datenschutz': Untrusted Hosts and Confidentiality

Josef Spillner <js177634@inf.tu-dresden.de>

24. Oktober 2003

Basierend auf dem Paper:

Untrusted Hosts and Confidentiality: Secure Program Partitioning

von Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, Andrew C. Myers

1 Einleitung

Das Thema Sicherheit in der Informationstechnologie wandert immer mehr vom administrativen Bereich hin zum Ursprung, der Entwicklung von Software. Erst seit wenigen Jahren gibt es Programmiersprachen, die entweder von sich aus oder durch Erweiterungen entsprechende Möglichkeiten bieten, auf Sprachebene Sicherheitskriterien festzulegen.

Konventionelle Zugriffsschutzmechanismen basieren auf Regeln, die das Subjekt betreffen: Man benötigt eine bestimmte Berechtigung, um auf ein Objekt zugreifen zu können.

Die dargestellte Methode bildet Informationen auf Objekte ab und befaßt sich mit Regeln, denen die Verwendung des Objekts unterliegt: Das Objekt darf nur von diesem und jenem Subjekt zu den betreffenden Verwendungen freigegeben werden.

Dadurch wird laut den Autoren des Papers nicht nur ein starker Zugriffsschutz erzielt, sondern es wird auch die formale Prüfung des Sicherheitsmodells ermöglicht, die eine sichere verteilte Verarbeitung von Daten erlaubt [SPP01].

Rechner (Hosts), denen man nicht vollständig vertraut, bekommen nur die Programmteile und Daten, die sie für eine spezifische Berechnung benötigen. Die Verteilungsstrategie belastet dabei nicht den Software-Entwickler, sondern kann automatisiert angewendet werden.

2 Beschreibung der Methode

Jeder Ausdruck in einem Programm unterliegt der Kontrolle eines Eigentümers ("principal"), wobei die dort verarbeiteten Daten durchaus andere Eigentümer haben können. Die Eigentümer werden in den sicherheits-getypten Daten mitgeführt, ebenso wie die Parteien, denen der Zugriff darauf erlaubt ist, und solche, die der Integrität der Daten vertrauen.

a) `{o1:r1 ; o2:r1,r2}`

b) `{o1:r1 ; o2:r1}`

a): Der Eigentümer o1 gewährt r1 Zugriff auf die Daten, während o2 auch r2 Zugriff erlaubt.

b): Dieses Label ist restriktiver, ausgedrückt als a <= b. Zuweisungen von a nach b sind möglich, nicht jedoch von b nach a, da gemäß b die Partei r2 keinen Zugriff bekommen darf.

Jeder Host hat zwei für die Sicherheit relevante Eigenschaften: Vertraulichkeit

(C) - bis zu welcher Höhe kann die Sicherheit der Daten garantiert werden - und Integrität (I) - welche sicheren Daten werden wieder entgegengenommen.

Nicht nur Variablen, sondern auch Funktionen und Anweisungen können getypt sein. In einem solchen Fall wird der jeweilige untergeordnete Block nur dann ausgeführt, wenn das Label des Programmzählers die verlangten Eigenschaften besitzt.

Sämtliche Vorgaben setzen eine sichere Umgebung voraus; Fehler im Netzwerk oder Einfluß von anderen Hosts als den für die Programmaufteilung zur Verfügung stehenden müssen separat durch geeignete (kryptographische) Maßnahmen verhindert werden. Fehlfunktionen können dazu führen, daß sämtliche Daten, die auf einem bestimmten Host ausgeführt werden, nach außen gelangen. Dies hat aber auf die Gesamtintegrität nur dann eine Auswirkung, wenn diesen Daten auf dem betroffenen Rechner vertraut wird.

Die Einhaltung der mit den Labels vorgegebenen Sicherheitsmerkmale wird sowohl auf Quelltextebene (statisch) wie auch zur Laufzeit (dynamisch) erzwungen.

2.1 Verteilung

Aufgrund der durch die Labels definierten Informationsflüsse wird eine Aufteilung des Programms auf die zur Verfügung stehenden Hosts vorgenommen. Die Aufteilung erfolgt primär anhand der Integritäts- und Vertraulichkeitsinformationen, und sekundär nach technischen Gesichtspunkten zur Gewinnung eines performanten verteilten Programmes.

Ein Rechner ist dann für die Berechnung eines Datums geeignet, wenn der Eigentümer des Datums der Weitergabe vertraut, und wenn dem zurückgelieferten Ergebnis vertraut wird.

Wichtig ist dabei, das unberechtigte Auslesen von Daten während der Übertragung zu verhindern. Aus diesem Grund werden zusätzlich virtuelle Kanäle (read channels) bereitgehalten, beispielsweise mit Hilfe von Zusatzvariablen, so daß die Informationsweitergabe über vertraulichere dritte Hosts geleitet werden kann.

2.2 Statisches Sicherheitsmodell

Der Zugriff auf Variablen kann mit den Methoden `declassify` und `endorse` gesteuert werden. Deklassifikation führt zu einer Relaxation der Zugriffsbeschränkungen, während Endorsement (Befürwortung) eine Einschränkung des Zugriffs darstellt.

Die Eigentümerschaft ganzer Code-Abschnitte kann mit der `authority`-Klausel festgelegt werden.

Sicherheits-getypte Daten dürfen nur von deren Eigentümern sowie den als leseberechtigt angegebenen Parteien gelesen werden. Dabei sind implizite Lesezugriffe zu vermeiden, wie durch das folgende Beispiel illustriert:

```
if x == true then
y=true;
else
y=false;
```

Aus diesem Grund gibt es zusätzlich zu den Labels noch einen Programmablaufzähler (program counter, pc), welcher als spezielles Label implementiert ist, das in jedem anderen Label implizit enthalten ist, insbesondere in denen, die automatisch in jedem Verzweigungspunkt des Programms enthalten sind.

Die oben angesprochene Deklassifikation ist hierbei sehr wichtig: Möchte man diese impliziten Prüfungen kontrolliert umgehen, beispielsweise bei der Ausgabe von verschlüsselten und als vertraulich eingestuften Daten, so kann man dies über die `declassify`-Anweisung tun.

2.3 Dynamisches Sicherheitsmodell

Zusätzlich zur statischen Modellierung ist es erforderlich, Überprüfungen zur Laufzeit mit einzubinden. Diese sollen allerdings vielmehr die Integrität des statischen Sicherheitsmodells sicherstellen, welches beispielsweise durch fehlerhafte Übertragungen von Informationen in einem Netzwerk beeinträchtigt werden könnte.

Überprüft werden unter anderem die Berechtigungen von Hosts bei Zugriffen auf Variablen sowie Weiterleitungen von Daten zwischen zwei Hosts ohne Zugriffsmöglichkeite für dazwischenliegende Hosts.

Für die Berechtigungsmodelle innerhalb einer Hostgruppe ist der Integritäts-Stack verantwortlich. Ein Integritäts-Stack (`ICS`) ist ein virtueller Speicherbereich, bestehend aus einzelnen Stacks auf den beteiligten Hostrechnern, deren Speicherfelder Informationen über das Vertrauen der Hosts ineinander haben. Damit lassen sich kontrolliert Daten verteilt bearbeiten.

Die Anweisungen `lgoto`, `rgoto`, `sync` und `forward` stehen dafür zur Verfügung. Die Übergabe der Kontrolle von einem Programmteil an einen anderen (mit niedriger Integrität) wird durch `rgoto` (regular goto) erledigt, während `lgoto` (LIFO goto) für die Gegenrichtung zu verwenden ist. Die mit `lgoto` vom Integritäts-Stack genommenen Elemente werden dort mit `sync` hinzugefügt, während `rgoto` den Stack nicht beeinflußt. Oben angesprochene Daten-

weiterleitungen werden durch `forward` realisiert.

3 Implementierung

Als Referenzimplementierung kommt das Java-Programm Jif/split zum Einsatz, welches die Java-Sicherheitserweiterung Jif erweitert. Jif (`jifc`) erlaubt die Programmierung mit sicherheits-getypten Daten, das heißt, Daten haben nicht nur einen lexikalischen Gültigkeitsbereich, sondern auch eine Sicherheits-Domäne, innerhalb derer auf sie zugegriffen werden darf (lesend oder schreibend), während von außerhalb kein Zugriff möglich ist.

Das Programm Split (`splitc`) kann nun anhand der Domänen eine Aufteilung des Programmes auf verschiedene Hosts vornehmen. Die Hosts werden in einer Konfigurationsdatei angegeben.

Kritischer Punkt hierbei ist der Gegensatz zwischen Sicherheit und Ablaufgeschwindigkeit. Je nach Anteil der Kommunikation am Programm sind Verlangsamungen um den Faktor 2 und mehr hinzunehmen, wenn alle Prüfroutinen für viele kleine Nachrichten verwendet werden. Empfohlen (und für die Implementierung in Zukunft angedacht) sind deshalb Optimierungen wie das Zusammenfassen von mehreren Nachrichten in einer sowie eine besser vom Entwickler steuerbare Auswahl der Hosts.

Aktuell ist eine Implementierung nur für die Sprache Java verfügbar, durch die Vorverarbeitung könnten aber auch andere Sprachen mit einbezogen werden, solange eine Jif-Laufzeitbibliothek dafür vorhanden ist.

Interessant an der aktuellen Implementierung ist die Tatsache, daß Labels, sofern nicht angegeben, aus der Umgebung inferiert werden können. Dies kann pro Klasse oder sogar pro Anwendung geschehen.

4 Praxisrelevanz

Die durchgängige Verwendung von sicherheits-getypten Daten ist in der Praxis nicht durchsetzbar, da der Entwicklungsaufwand immens steigen würde. Stattdessen ist es aber denkbar, daß in sicherheitskritischen Bereichen die Kernfunktionalität von Anwendungen durch solche Verfahren abgesichert werden.

Durch die Implementierung bekommt man dann nicht nur einen Quelltext, dessen Korrektheit leichter zu beweisen ist, sondern auch die Möglichkeit, Daten in verteilten Systemen sicher zu berechnen. Im Zeitalter des Internet bedeutet dies, daß Rechner, die normalerweise nicht notwendigerweise als vertrauenswürdig eingestuft werden können, durch die automatische Partitionierung der Berech-

nung eine Teilberechnung durchführen, mit deren Ergebnis allein der Nutzer des Rechners nichts anfangen kann, das Resultat jedoch den Sicherheitsansprüchen genügt.

4.1 Praxisfall P3P

Die Autoren des Papers haben als eine mögliche Nutzung des Verfahrens die W3C-Empfehlung für P3P (Platform for Privacy Preferences) angegeben. Diese beschreibt einen Standard, mit dem Inhaber von Webseiten angeben können, wie mit eingegebenen Nutzerdaten verfahren wird [W3C02].

Die Weiterverarbeitung der Daten kann über Jif/split erzwungenermaßen in dem in der P3P-Sektion der Webseite festgelegten Rahmen erfolgen. In der Realität konkurriert dieser Ansatz mit existierenden (konventionellen) Lösungen wie PKI-Verfahren, hebt aber die Vertraulichkeit der Daten hervor.

Generell ist der Einsatz von sicherheits-getypten Sprachen in verteilten Systemen für den Datenschutz interessant, reicht allein aber nicht aus, um den Schutz der Daten zu gewährleisten, da nicht nur Verarbeitungsvorgänge, sondern auch die Weitergabe von Daten geregelt werden muß.

4.2 Praxisfall Battleship

Eine eher überschaubare Referenzimplementierung wird seit der Version 1.0 von Jif mitgeliefert. Das für zwei Spieler ausgelegte Brettspiel Battleship wurde so entwickelt, daß jeder Spieler als Principal seines eigenen Spielbretts auftritt. Sollen nun Informationen über das Spielbrett freigegeben werden, beispielsweise um einen Treffer zu erkennen, so muß diese Information erst vom Spieler deklassifiziert werden.

Im Gegensatz zum verteilten Rechnen sind hier also mehrere Parteien Teil des Systems [SS03].

4.3 Elektronische Wahlen

Elektronische Wahlen werden allerorts eingeführt, und mit ihnen Abläufe, die etlichen Ansprüchen genügen müssen: Jeder Wahlberechtigte soll genau eine Stimme abgeben, die Wahl soll geheim ablaufen, und der Wähler sollte sich sicher sein können, daß seine Stimme angenommen wurde und in die Statistik mit eingeht [OT99].

Für derartige Anwendungen hat man "vergessliche Übertragungen" (oblivious

transfers) eingeführt, welche in verschiedenen Varianten existieren und beispielsweise mit dem Diffie-Hellman-Austausch oder RSA-Verfahren realisiert werden.

Mit Hilfe sicherheits-getypter Daten kann man diese Prozedur aus Sicht des Entwicklers bei gleicher Funktionalität wesentlich vereinfachen. Zwar ist ein dritter Host erforderlich, um die Vertraulichkeit der übergebenen Daten gewährleisten zu können, dafür sind aber alle Risiken bezüglich fehlerhafter Anwendung von kryptographischen Routinen oder Fehler in ihnen minimiert.

5 Einschränkungen

Weitere Papers zu diesem Thema, wie [SIF98], haben gezeigt, daß ein für eine bestimmte Gruppe von Programmen ausgelegter Schutzmechanismus nicht notwendigerweise für andere Gruppen gilt. So ist beispielsweise die Sicherheits-Typisierung von Variablen mit den Typen `HIGH` und `LOW` bei paralleler Ausführung (Multi-Threading) nur dann sinnvoll, wenn While-Schleifen und teilweise auch If-Bedingungen bestimmten Anforderungen genügen.

Die Wahl der Programmiersprache ist ebenfalls entscheidend. So sind imperative Sprachen noch nicht von Haus aus mit Methoden wie Typ-Inferenz ausgestattet, was die Kontrolle und Auswertung des Informationsflusses erschwert.

5.1 Angriffsmöglichkeiten

Das Sicherheitsmodell ist mit etlichen Schutzmechanismen ausgestattet. Dennoch folgt eine kurze Zusammenstellung möglicher Angriffspunkte, die eventuell noch nicht ausreichend beleuchtet worden sind.

1. Ausnutzung von `endorse`-Anweisungen durch unbefugte Rechner macht es ihnen möglich, an nicht für sie bestimmte Daten heranzukommen. Dieses Problem wird durch kryptographische Methoden aufgefangen, ein Exploit wäre aber denkbar.

2. Kollaboration mehrerer Hosts zur ungewollten Datengewinnung. Dieses Risiko ist abschätzbar: die Vertraulichkeit von Daten ist beispielsweise nur dann gefährdet, wenn die Vereinigung des Vertrauens in alle beteiligten Rechner mächtiger ist als das Vertrauen im betroffenen Programmstand.

3. Senden gefälschter Nachrichten unter dem Namen eines anderen Hosts. Auch hier sind kryptographische Prüfverfahren zur Abwehr hilfreich.

6 Materialien

[OT99]
Oblivious Transfer
Bruce Litow, 1999
http://www.cs.jcu.edu.au/Subjects/cp5030/1999/lectures/oblivious/oblivious.html

[SIF98]
Secure Information Flow in a Multi-threaded Imperative Language
Geoffrey Smith and Dennis Volpano, 1998
http://www.cs.fiu.edu/~smithg/papers/popl98.ps

[SPP01]
Untrusted Hosts and Confidentiality: Secure Program Partitioning
Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, Andrew C. Myers
http://www.cs.cornell.edu/zdance/zznm01-abs.htm

[SS03]
How Should We Make Software Secure?
http://research.microsoft.com/projects/SWSecInstitute/slides/Myers.pdf

[W3C02]
The Platform for Privacy Preferences 1.0 (P3P1.0) Specification
http://www.w3.org/TR/P3P/

Jif/split-Sourcen
http://www.cs.cornell.edu/jif

Quellen zur Ausarbeitung
http://mindx.dyndns.org/uni/papers/securepartitioning/ausarbeitung/