Schutzkonzepte in Multics

Benedict Herzog
Universität Erlangen-Nürnberg
Martensstraße 1
D-91058 Erlangen
benedict.b.herzog@fau.de

Kurzfassung

Multics war von Anfang darauf angelegt ein Betriebssystem zu werden, dass Maßnahmen für die Sicherheit vor Fehler und die Eindämmung selbiger bietet, als auch Sicherheit vor bösartigen Angreifern. Kurz, Multics sollte ein sicheres Betriebssystem werden. Um einen effektiven Schutz zu gewährleisten, mussten diese Maßnahmen von Anfang an im Entwurf eingeplant werden. Außerdem wurde der Entwurf an die technischen und organisatorischen Gegebenheiten angepasst.

Die Arbeit an Multics hat viele Einblicke in den den Aufbau und Betrieb sicherer Rechenanlagen gebracht. Noch heute sind viele der entwickelten Schutzkonzepte in Betriebssystemen auf der ganzen Welt zu finden: Virtuelle Adressräume, Zugriffskontrolllisten, Passwortschutz für Benutzer, usw. Diese Arbeit soll einige der entwickelten Schutzkonzepte und deren Verwendung vorstellen und ein Gefühl dafür vermitteln, was bei der Entwicklung und Benutzung eines sicheren Betriebssystems wichtig ist.

Stichwörter

Betriebsysteme, Schutzkonzepte, Zugriffsschutz, Multics

1. EINLEITUNG

Von Anfang an stand in der Entwicklung von Multics[1] der Mehrbenutzerbetrieb mit vielen Benutzern mit sehr unterschiedlichen Anforderungen im Vordergrund. Um einen sicheren Betrieb mit mehreren Benutzern gewährleisten zu können, war es nötig, eine Vielzahl von verschiedenen Schutzkonzepten zu entwickeln. Dazu gehört die Isolation von Daten und Prozeduren verschiedener Benutzern voreinander und ein höchstmögliches Maß an Vertraulichkeit, aber auch das gezielte Freigeben und Teilen von Daten durch die Benutzer, um einen möglichst produktiven Betrieb zu ermöglichen [3, 16].

Um die Isolation der Benutzer zu gewährleisten, besitzt jeder Multics-Prozess einen virtuellen Adressraum, der durch Segmentierung und einer Seitenverwaltung realisiert wird. Er besteht aus einer Menge von Segmenten, die zugleich die Einheit sind, auf der Zugriffsrechte vergeben werden können. Multics besitzt Methoden, um anderen Benutzern Zugriff auf bestimmte Daten, Prozeduren oder ganze Subsysteme zu gewähren. Dabei ist sehr feingranular kontrollierbar, wer auf welche Daten in welcher Weise zugreifen darf.

Zur dauerhaften Speicherung von Daten, besitzt Multics ein hierarchisches Dateisystem, das von der Struktur her unseren heutigen Dateisystemen ähnelt. Für einen adäquaten Schutz dieser Daten, muss auch das Dateisystem eine Rechteverwaltung besitzen, die den Zugriff auf Ordner und Dateien regelt [4].

Sicherheit war während der gesamten Entwicklung von Multics ein wichtiger Aspekt, da bei allen vorherigen Systemen die Sicherheitskonzepte umgangen werden konnten [13]. Insbesondere für militärische Zwecke stellte das ein großes Problem dar und das Militär entschied einen Anforderungskatalog für sichere Computersysteme zu erstellen: das *Orange Book* [5]. Die Entwicklung einer einheitlichen und systemantischen Evaluation der Sicherheit von Computersystemen wurde durch mehrere Projekte in den 1970er Jahre initiiert und lief parallel zur Entwicklung von Multics. Tatsächlich haben sich die einzelnen Projekte und Multics immer wieder gegenseitig beeinflusst und 1983 wurde eine erste Version des Orange Book vorgestellt, dass die Ergebnisse der Projekte in ein Dokument zusammengefasst hat. Darin werden unterschiedliche Sicherheitsstufen (A1, B3, B2, ...) und Methoden für die Bewertung von Computersystemen definiert. Multics erreichte 1985, nach drei Jahren formaler Evaluation, die Stufe B2, die bis dahin kein anderes System erreicht hatte und auch lange kein anderes erreichen sollte [10].

Die Arbeit ist wie folgt gegliedert: Kapitel 2 beschreibt, wie der virtuelle Speicher bei Multics aufgebaut ist und wie Daten und Prozeduren im Speicher geschützt werden. Kapitel 3 stellt die Authentifizierung von Benutzer vor und welche Schutzmaßnahmen dort ergriffen wurden. In Kapitel 4 wird die Umsetzung von Schutzkonzepten im Dateisystem von Multics beschrieben. Kapitel 5 beschreibt den Prozess zur Erlangung der B2 Zertifizierung nach dem Orange Book. Außerdem fasst es kurz die Entwicklung von Zertifizierungen bis heute zusammen. Kapitel 6 rekapituliert die Arbeit und gibt einen Ausblick.

2. SCHUTZKONZEPTE IM ADRESSRAUM

Eines der Ziele von Multics war der sichere Mehrbenutzerbetrieb, also dass sich gleichzeitig mehrere Programme unterschiedlicher Benutzer in Ausführung befinden können sollen. Die Benutzer sollen dabei aus ganz unterschiedlichen Domänen kommen können und daraus folgend auch ganz unterschiedliche Schutzbedürfnisse haben dürfen.

Es gibt einerseits Benutzer, deren Daten strikt vor anderen Benutzern geschützt werden müssen, weil es sich um geheime Daten handelt oder die Benutzer in einem (kommerziellen) Wettbewerb zueinander stehen. Auf der andere Seite gibt es Gruppen von Benutzer, die zusammen an einem

[1] **Mult**iplexed **I**nformation and **C**omputing **S**ervice

gemeinsamen Projekt arbeiten und gegenseitigen Zugriff auf Projektdaten haben müssen. Es gibt auch Daten, die in keine der beiden Kategorien passen, zum Beispiel öffentliche Prozedurbibliotheken, die von allen ausgeführt, aber nicht geändert werden dürfen [7].

Aus diesen Anforderungen folgt der Bedarf eines flexiblen Mechanismus zu Rechtevergabe und -verwaltung für Daten und Prozeduren innerhalb des Adressraums von Multics, der im Folgenden beschrieben wird.

2.1 Eigenschaften des Schutzmodells

In früheren Systemen wurde Adressraumschutz meistens dadurch realisiert, dass es zwei Ausführungsmodi, einen priviligierten und einen nicht-priviligierten, gab und durch ein *Memory-Bounds-Register* der Adressraum in zwei Domänen geteilt wurde[2]. Eine der beiden Domänen lies sich aus beiden Modi zugreifen, die andere nur aus dem priviligierten Modus. Diese Art von Schutz gewährte aber immer nur einen Alles-oder-Nichts Schutz. Um den oben beschriebenen Anforderungen gerecht zu werden, bedarf es flexiblerer Schutzmechanismen. Die Eigenschaften für einen solchen Mechanismus wurden in [7] zusammengefasst und werden in diesem Abschnitt vorgestellt.

Ziel ist es für alle logisch zusammengehörenden Informationen eigene Zugriffsrechte vergeben zu können. Der Adressraum muss also in viele kleinere Bereiche (genannt *Segmente*) geteilt werden können und für jedes Segment sollen eigene Zugriffsrechte vergeben werden können. Hierbei wurden die (auch heute noch üblichen) Arten von Zugriff unterschieden:

- (nicht-)schreibbar
- (nicht-)lesbar
- (nicht-)ausführbar

Neben der Art der Zugriffsrechte muss noch geregelt werden, für welchen Benutzer bestimmte Zugriffsrechte gelten sollen. Der Besitzer von Daten braucht zum Beispiel andere Zugriffsrechte, als jemand, der die Daten zwar lesen, aber keine Änderungen daran ausführen können soll.

Aus dieser Anforderung folgt das Konzept von *logischen Segmenten*. Jeder Benutzer hat Zugriff auf eine Menge von logischen Segmenten. Jedem logischen Segment ist eine Menge von Zugriffsrechten zugeordnet und es zeigt auf ein physisches Segment. Daraus folgt, dass es mehrere logische Segmente (von unterschiedlichen oder sogar des gleichen Benutzern) geben kann, die auf dasselbe physische Segment zeigen, aber unterschiedliche Zugriffsrechte haben. Dadurch lassen sich unterschiedliche Zugriffsrechte für unterschiedliche Benutzer auf ein und dieselben Daten realisieren.

Ziel war es, dass jeder Benutzer nur Zugriffsrechte auf die Segmente bekommt, die er unbedingt braucht, um seine Aufgaben erledigen zu können. Diesem Prinzip folgend lassen sich Segmente in Schutzebenen organisieren, wobei gilt: je höher die Ebene, desto mehr Rechte hat ein Prozess, kann also auf mehr und weitreichender auf Segmente zugreifen und umgekehrt. Gleichzeitig gilt, je höher die Ebene, desto weniger Prozesse sollten auf dieser Ebene laufen. Die Rechte eines Prozesses leiten sich also davon ab, in welcher Schutzebene er läuft. Um auf ein Segment (Daten oder Prozedur) zuzugreifen zu können, muss er mindestens auf der, dem Segment zugeordneten, Schutzebene laufen. Der Prozess kann natürlich auch auf einer Schutzebene laufen, die weitreichendere Rechte gewährt, das sollte jedoch vermieden werden.

Zusammenfassend sollte ein Schutzmechanismus die folgenden drei Eigenschaften haben [7]:

- Es soll möglich sein, einen Prozess komplett von allen anderen Prozessen zu isolieren. Dass heißt, dass kein anderer Prozess auf irgendein Segment des Prozesses zugreifen kann.
- Es soll möglich sein, beliebige Segmente kontrolliert und komfortabel anderen Benutzern zugänglich zu machen und dabei Zugriffsrechte pro Benutzer vergeben zu können.
- Es soll möglich sein, Prozeduren durch Schutzebenen vor Missbrauch zu schützen. Zusätzlich dazu soll der Übergang eines Prozess zwischen verschiedenen Ebenen transparent sein, so dass Prozeduren ohne Aufwand anderen Ebenen zugeordnet werden können.

2.2 Virtueller Speicher

Um die Schutzkonzepte des Adressraums verstehen zu können, ist es nötig den Aufbau des virtuellen Speichers in Multics zu verstehen. Dieses Kapitel gibt einen kurzen Übersicht über den Aufbau des Adressraums von Multics.

Jeder Prozess in Multics besitzt seinen eigenen Adressraum, der aus maximal 2^{14} Segmenten besteht. Dabei ist ein Segment ein Feld von aufeinander folgenden Speicherwörtern. Ein Segment kann aus maximal 2^{18} 36-bit Speicherwörtern bestehen. Ein Segment kann also maximal 1152 KiByte und der gesamte Adressraum (theoretisch) 18 GiByte adressieren [3].

In einem Segment sollen logisch zusammenhängende Informationen zusammengefasst werden. Ein Beispiel für logisch zusammenhängende Informationen ist der Code für eine Prozedur. Tatsächlich sollte in Multics anfangs jede Prozedur in einem eigenen Segment liegen. Das wurde jedoch aus Leistungsgründen wieder aufgegeben. Konsequenterweise stellt das Segment auch die kleinste mögliche Einheit zur Rechteverwaltung dar; für jedes Segment können unterschiedliche Zugriffsrechte vergeben werden.

Die Adresse für den Zugriff auf ein Speicherwort besteht aus zwei Teilen:

$$\langle\ S\ ,\ W\ \rangle$$

Die *Segmentnummer* S, die das referenzierte Segment bestimmt und die *Wortnummer* W, die das referenzierte Wort innerhalb des Segments bestimmt. Jedes Segment wird durch einen Segmentdeskriptor beschrieben, der Metainformationen über das Segment enthält. Er enthält zum Beispiel die physikalische Startadresse und die Länge eines Segments und die zugeordneten Zugriffsrechte. Jeder Prozess hat ein *Deskriptorsegment*, in dem alle verfügbaren Segmente, für diesen Prozess, beschrieben werden, es enthält also alle Segmentdeskriptoren. Ein Segmentdeskriptor wird durch ein *Segment Descriptor Word* (SDW) realisiert, womit das Deskrip-

[2]Dies war zum Beispiel bei der GE-635 der Fall. Mit der GE-645 wurde dieses Feature durch die Segmentierung und die Seitenverwaltung ersetzt. In der Honeywell 6000er Serie hielt das Feature, parallel zur Segmentierung und Seitenverwaltung, jedoch wieder Einzug, damit das General Electrics eigene Betriebssystem *Gecos* auf den Prozessoren lauffähig war.

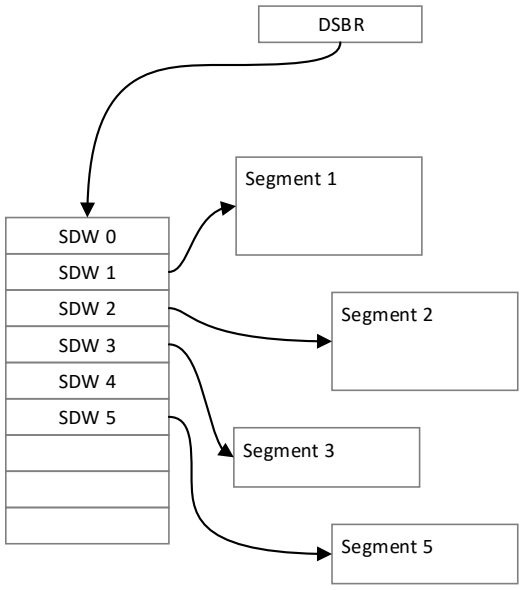

Abbildung 1: Das *Descriptor Segment Base Register* (DSBR) zeigt auf ein Deskriptorsegment mit sechs *Segment Descriptor Words* (SDWs). Ein SDW zeigt jeweils auf ein physisches Segment [11].

torsegment aus einer Folge von SDWs besteht. Die Segmentnummer aus der Adresse, die das referenzierte Segment bestimmt, ist der Index in das Deskriptorsegment. Die physikalische Anfangsadresse des Deskriptorsegments wird durch das *Descriptor Segment Base Register*[3] (DSBR) bestimmt [19]. Abbildung 1 zeigt ein beispielhaftes Deskriptorsegment mit sechs SDWs.

Das erste SDW (Position 0) beschreibt per Konvention immer das Deskriptorsegment selbst. Die darauf folgenden etwa 100 SDWs zeigen für alle Prozesse auf dieselben Segmente[4], die vom Supervisor benötigt werden, blenden also den Adressraum des Supervisors in den Adressraums des Prozesses ein [11].

Wird das DSBR geändert, werden ab diesem Zeitpunkt alle Adressen relativ zu einem anderen Deskriptorsegment aufgelöst, womit der Kontextwechsel zwischen Prozessen implementiert wird.

Multics unterstützt auch eine Seitenverwaltung. Normalerweise wächst ein Segment immer seitenweise und die Seitenverwaltung muss zur Übersetzung von virtuellen in physikalische Adressen hinzugezogen werden. Da die Seitenverwaltung aber keinen Einfluss auf die wesentlichen Schutzmechanismen hat und für Prozesse völlig transparent ist, wird die Seitenverwaltung aus Übersichtlichkeitsgründen in dieser Arbeit ausgeklammert [19]. Die Seitenverwaltung wird ausführlich in [15] vorgestellt.

2.3 Abstraktes Schutzmodell

[3]auch: *Descriptor Base Register* (DBR)

[4]Es gibt einige Ausnahmen von dieser Regel, so gibt es Segmente in diesem Bereich, die jeweils pro Prozess erzeugt werden und einige andere Sonderfälle [11].

In diesem Kapitel wird das abstrakte Schutzmodell von Multics für den Adressraum vorgestellt. Damit das Schutzmodell funktionieren kann, müssen nach [19] die folgenden drei Annahmen erfüllt sein:

- Für jeden Benutzer, der sich am System anmeldet, wird ein neuer Prozess mit einem virtuellen Adressraum erzeugt. Außerdem wird der Name des Benutzers an den Prozess gebunden, so dass jeder Prozess eindeutig einem Benutzer zugeordnet werden kann. Der Prozess ist die einzige Möglichkeit für einen Benutzer mit dem System zu interagieren, also auf Daten zuzugreifen oder sie zu manipulieren.
- Alle Informationen, die zur Laufzeit zugreifbar sind, sind in Segmente organisiert. Ein Prozess kann auf Segmente nur zugreifen, wenn sie vorher seinem virtuellen Speicher hinzugefügt worden sind und er passende Zugriffsrechte für das Segment hat.
- Alle Benutzer, die Zugriff auf ein Segment haben, sind in einer *Zugriffskontrollliste*[5] genannt, die mit dem Segment verknüpft ist.

In Multics sind diese drei Annahmen erfüllt.

Aus den oben genannten Annahmen folgt, dass bei der Prozesserzeugung der Supervisor dem virtuellen Adressraum des Prozesses neue Segmente hinzufügen muss. Voraussetzung hierbei ist, dass der Name des Benutzers, der mit dem Prozess verknüpft ist, zu einem Eintrag in der Zugriffskontrollliste des Segments passt. Das Hinzufügen zum virtuellen Adressraum geschieht durch das Hinzufügen eines SDW zum Deskriptorsegments des Prozesses. Im SDW werden auch die passenden Zugriffsrechte, die der Prozess für das Segment hat, hinterlegt, so dass diese bei jeder Referenz überprüft werden können. Die Zugriffsrechte für ein konkretes SDW kommen aus dem Eintrag der Zugriffskontrollliste, der zu dem Namen des Benutzers gepasst hat.

2.3.1 Schutzdomänen

Mit den bisher beschriebenen Mechanismus der SDWs im Deskriptorsegment können jedem Prozess bestimmte Zugriffsrechte auf Segmente vergeben werden. Im Laufe der Ausführung können sich die notwendigen Zugriffsrechte eines Programms jedoch ändern. Zum Beispiel wenn ein Programm den Supervisor aufruft, um eine Ein-/Ausgabe Operation anzustoßen, müssen die Rechte beim Übergang zum Supervisor erweitert werden und nach erfolgter Ein-/Ausgabe wieder reduziert werden. Ein anderes Beispiel ist ein Prozess, der einem anderen Prozess über ein geschütztes Subsystem Zugriff auf sensible Daten gewähren möchte.

Eine Menge von bestimmten Zugriffsrechten wird daher oft zusammengefasst und als *Schutzdomäne* bezeichnet. So können zum Beispiel die Menge aller Zugriffsrechte, die in den SDWs des Deskriptorsegments kodiert sind, zusammengefasst als Schutzdomäne bezeichnet werden.

Ein allgemeines Schutzmodell würde eine beliebige Menge von Schutzdomänen pro Prozess erlauben und keine Einschränkungen für die Beschaffenheit der Zugriffsrechte innerhalb einer Domäne treffen. Um jedoch eine effiziente Implementierung in Hardware zu ermöglichen, ist es nötig bestimmte Einschränkungen zu treffen. In Multics wurden die Schutzdomänen so eingeschränkt, dass *Schutzringe* entstanden, die im folgenden Abschnitt vorgestellt werden [19, 7].

[5]engl. *Access Control List* (ACL)

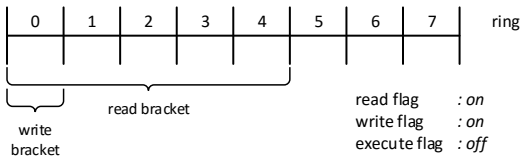

Abbildung 2: Datensegment, dass schreibbar in Ring 0 und lesbar in den Ringen 0-4 ist [19].

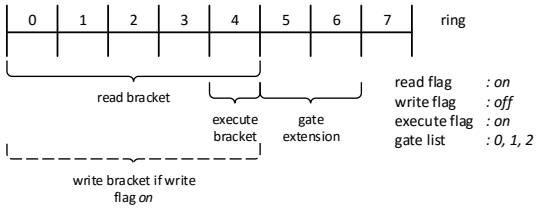

Abbildung 3: Ausführbares, nicht schreibbares Segment (Codesegment) mit Gates und einer Gate Extension [19].

2.3.2 *Schutzringe*

Das Schutzringkonzept hat die folgenden Eigenschaften:

- Die Zahl der Schutzringe pro Prozess wird auf eine feste Zahl r festgelegt, wobei jedem Schutzring eine Zahl zwischen 0 und $r-1$ zugeordnet wird.
- Die Menge der Zugriffsrechte in einem Ring m muss immer eine Teilmenge der Zugriffsrechte aller Ringe n mit $n < m$, sein. Dass heißt der Ring 0 hat die größte und der Ring $r-1$ die kleinste Menge von Zugriffsrechten.

Im Grunde ist das Schutzringkonzept eine Erweiterung des Vorgesetzter/Untergebener[6] Prinzips. Daraus folgt, dass ein Prozess die höchsten Zugriffsrechte hat, wenn er in Ring 0 und die niedrigsten, wenn er in Ring $r-1$ ausgeführt wird.

Mit der Einführung der Schutzringe müssen die oben beschriebenen SDWs noch um die Information erweitert werden, in welchem Ring welche Zugriffsrechte gelten. Durch die Bedingung, dass die Menge der Zugriffsrechte eines Ringes eine Teilmenge der Zugriffsrechte des nächstniedrigeren Ringes sein muss, reicht es anzugeben, bis zu welchem Ring ein Zugriffsrecht gilt. Es entstehen also Rechtebereiche vom höchstpriviligierten Ring 0 bis zu einem Ring x, der die minimal benötigte Privilegien bestimmt. Es gibt jeweils einen Bereich der angibt, in welchen Ringen ein Prozess Lesen, Schreiben und Ausführen darf.

Zusätzlich werden noch drei Flags eingeführt, die für jedes der drei Rechte angeben, ob es überhaupt aktiv ist. Ist das Flag gesetzt, gilt der angegebene Rechtebereich. Ist das Flag nicht gesetzt, darf der Prozess aus keinem Ring das jeweilige Recht in Anspruch nehmen.

Abbildung 2 zeigt beispielhaft ein Datensegment mit acht Ringen. Da das Ausführen Flag nicht gesetzt ist, darf kein Code aus dem Segment ausgeführt werden und es kann nur Daten aufnehmen. Lesen ist von Ring 0 bis 4 erlaubt, Schreiben ist nur in Ring 0 erlaubt.

Die Einführung von Schutzringen macht es notwendig ein neues Recht einzuführen, nämlich das Recht den Ring zu wechseln. Durch die Architektur der Schutzringen ist es nicht nötig den Wechsel in höhere Ringe zu kontrollieren, da dadurch nur Rechte abgegeben, aber niemals hinzugewonnen werden können. Der Wechsel in niedrigere Ringe muss jedoch kontrolliert werden, da ein Prozess so mehr Rechte erlangen kann.

Die Kontrolle von Ringwechsel wird in Multics darüber implementiert, dass ein Programm einen niedrigeren Ring nur dann betreten darf, wenn er es über eine bestimmte Stelle tut, also ein bestimmtes Programm ausführt. Stellen, die einen Ringwechsel ermöglichen, werden *Gates* genannt. Weil ein Ring nur über bestimmte Programme betreten werden kann, kann genau kontrolliert werden, welche Aktionen ein Programm in einem bestimmten Ring ausführen kann und wie es die neu hinzugewonnen Zugriffsrechte verwenden kann.

Gates werden durch eine Liste von Programmstellen spezifiert, die jedem Deskriptorsegment angehängt werden. Die Liste kann auch leer sein, so dass ein Prozess keine Möglichkeit hat den Ring bzw. in einen bestimmten Ring zu wechseln.

Zusätzlich soll es möglich sein die Ringe zu spezifizieren, denen es erlaubt ist einen Ringwechsel (über ein Gate) durchzuführen. Dazu gibt es einen extra Rechtebereich, der sich an den Rechtebereich für das Ausführenrecht anschließt, genannt *Gate Extension*.

Bis jetzt geht jeder Rechtebereich, ausgenommen die Gate Extension, von Ring 0 bis zu einem zu spezifierenden Ring x. Man hat jedoch beobachtet, dass Prozeduren normalerweise genau einen Ring haben in dem sie üblichweise ausgeführt werden (sollen). Um unbeabsichtigte Ringwechsel in niedrigere Ringe als nötig zu vermeiden, wird das Ausführenrecht zusätzlich zur oberen Grenze mit einer flexiblen unteren Grenze versehen. Es kann nun also auch eingeschränkt werden, in welchem Ring ein Programm höchstens laufen darf und damit die maximal möglichen Rechte, die ein Prozess während der Ausführung des Programms besitzt.

Dieses erweiterte Ringmodell wird durch Abbildung 3 am Beispiel eines Codesegments veranschaulicht. Das Segment ist ausführbar aus Ring 4 (das üblicherweise für Benutzerapplikationen reserviert ist) und es ist möglich aus den Ringen 5 und 6 über eines der spezifizierten Gates in Ring 4 zu wechseln. Es ist explizit nicht möglich, dass aus den Ringe 0 bis 3 und 5 bis 7 Code aus dem Segment ausgeführt wird. Das Segment ist nicht schreibbar, was sinnvoll für ein Codesegment ist, und lesbar aus den Ringen 0 bis 4.

Die Liste mit den Gates und die Rechtebereiche bzw. Flags für jedes SDW kommen aus der, am Anfang beschrieben, Zugriffskontrollliste, die jedem Segment zugeordnet ist. Wird ein neues Segment einem Prozess zugeordnet, werden die zu diesem Prozess passenden Einträge aus der Liste herausgesucht und in das SDW im Deskriptorsegment geschrieben. Die Zugriffskontrollliste wird in der Datei, die das Segment repräsentiert, gespeichert. Wie die Umsetzung von Einträgen in SDWs geschieht, ist dabei abhängig von der verwendeten Hardware- und Softwareversion.

[6] engl. Master/Slave

2.3.3 Schutzringe in der Praxis

Das in den obigen Abschnitten beschriebene Schutzmodell lässt eine sehr flexible Nutzung zu. Insbesondere die Zuordnung von Prozeduren zu bestimmten Ringen lässt viele Freiheitsgrade. In diesem Kapitel soll nun eine mögliche praktische Aufteilung der Ringe beschrieben werden, die in [19, S. 14] vorgestellt wird. Der Vorschlag basiert auf acht Ringen, wie sie in einer Version von Multics implementiert wurden[7].

Durch die Erweiterung des vorher üblichen Vorgesetzter/ Untergebener Modells ist es nun möglich auch für den Supervisor mehrere Schutzebenen zu benutzen. So schlagen die Autoren vor, dass nur Funktionen für die Ein-/Ausgabekontrolle, die Prozessoraufteilung, die Speicherverwaltung und die Rechteverwaltung im höchstpriviligierten Ring 0 (auch *hard core* genannt) laufen sollen. Alle weiteren Subsysteme des Supervisors, zum Beispiel das Messen bestimmter Kenngrößen, gepufferte Ein-/Ausgabeoperationen oder das Suchen im Dateisystem, sollen im Ring 1 implementiert werden. Ringwechsel in Ring 0 oder 1 sollen nur aus den Ringen 2 bis 5 möglich sein und natürlich nur über bestimmte Gates (vergleichbar mit Systemaufrufen in modernen Betriebssystemen). Ein Zugriff aus Ring 6 und 7 war nicht vorgesehen.

Durch die Zugriffskontrolllisten und den Deskriptorsegmenten pro Prozess ist es möglich, nicht allen Prozessen alle Gates zur Verfügung zu stellen. Zum Beispiel müssen bestimmte Gates zur Benutzerverwaltung nur Prozessen von Administratoren zugänglich sein.

Zusätzlich müssen nicht allen Gates dieselbe Gate Extension zugeordnet sein. Für bestimmte Gates kann durch die Gate Extension spezifiziert werden, dass sie nur aus Ring 1 aus aufgerufen werden können und damit eine Schnittstelle innerhalb des Supervisors implementieren.

Für Benutzerprozesse war Ring 4 vorgesehen, womit Ring 2 und 3 für andere Zwecke verfügbar waren. Die Autoren schlagen vor hier von Benutzer geschriebene Subsysteme zu implementieren. So wäre es möglich ein Loggingsubsystem zu schreiben, das jeden Zugriff auf bestimmte Daten loggt. Dabei genießt das Subsystem denselben Schutz vor Benutzerprozessen, wie der Supervisor vor Benutzerprozessen und Subsystemen, ohne jedoch in den Supervisor selbst aufgenommen werden zu müssen. Damit lassen sich Subsysteme effektiv schützen, ohne die Sicherheit des Supervisors durch absichtliche oder unabsichtliche Fehler zu kompromittieren.

Die Ringe 5 bis 7 sind für den *Selbstschutz* des Benutzers vorgesehen. So ist es zum Beispiel möglich ein selbstgeschriebenes Programm in Ring 5 auszuführen, während dem Programm nur Zugriffsrechte auf unbedingt benötigte Segmente eingeräumt werden und nicht auf alle für den Benutzer zugreifbaren Segmenten. So lassen sich schnell Adressierungsfehler aufdecken. Außerdem kann es dazu genutzt werden, von anderen Benutzern geschriebene Programme auszuführen.

Da Ring 6 und 7 keinen Zugriff auf Supervisorfunktionen haben, sind sie perfekt isoliert und können für nicht vertrauenswürdige Programme genutzt werden. Andererseits ist es nicht ganz einfach einen Anwendungszweck für die Ringe zu finden, da nur sehr wenige Programme komplett ohne Zugriff auf den Supervisor auskommen. Um die Ringe effektiv nutzen zu können, müsste der Benutzer selbst bestimmte Gates implementieren, die dann Prozessen in Ring 6 und 7 zur Verfügung gestellt werden, weswegen diese Designentscheidung diskussionwürdig ist [11].

[7] Je nach Hard- und Softwareversion waren es mehr als 8 Schutzringe

3. AUTHENTIFIZIERUNG

Wie in den Voraussetzungen in Kapitel 2.3 beschrieben, wird jedem Prozess zugeordnet, zu welchem Benutzer er gehört. Nötig war das, um zu entscheiden welche Einträge der Zugriffskontrollliste auf diesen Prozess zutreffen und damit zu entscheiden, welche Rechte ein Prozess bekommt.

Da Multics immer mit einem Fokus auf Sicherheit und Vertraulichkeit entwickelt wurde, folgt daraus, dass Maßnahmen zur Authentifizierung von Benutzern getroffen werden mussten. Dazu wurde jedem Benutzer eine eindeutige Kennung zugeordnet, die in der Regel aus seinem Nachnamen und ein bis zwei Initialen bestand. Einmal vergebene Kennungen waren für immer gültig und konnten nicht gelöscht und erneut vergeben werden. Alle Prozesse mussten entweder interaktiv mit einem achtstelligen (nur ASCII-Zeichen) Passwort authentifiziert werden oder aus einem bereits authentifizierten Prozess entstehen.

Auf die Sicherheit des Passwortes wurde besonderen Wert gelegt. So wurden technische Maßnahmen ergriffen, dass das Passwort niemals ausgedruckt werden sollte, weder absichtlich noch unabsichtlich. Dazu wurden die Passwörter nicht im Klartext, sondern verschlüsselt im Speicher abgelegt, um eine versehentliche Ausgabe zu verhindern. Musste das Passwort eingegeben werden, sollte der Drucker nur sinnlose Zeichen ausgeben, anstatt des eingetippten Passworts. Ziel war es, dass durch den Entwurf verhindert wird, dass jemand sein Passwort an einen Dritten preisgibt bzw. preisgeben muss.

Da es unter Umständen sehr aufwändig war neue Jobs in die Anlage zu bringen, haben viele Forscher diese Aufgabe an Dritte übertragen. Für die Ausführung des Jobs waren aber in der Regel die Rechte des Forschers nötig, weswegen Multics sogenannte *Proxy-Logins* anbietet. Dabei gibt der Forscher einmal an, dass ein bestimmter anderer Benutzer das Recht hat, Prozesse in seinem Namen zu starten. Dann konnte sich der Dritte mit seinem eigenen Passwort anmelden und den Prozess mit den Rechten des Forschers zur Ausführung bringen.

Der Name, der einem Prozess zugeordnet wird, besteht aus den folgenden drei Teilen:

Benutzerkennung : **Gruppe** : **Abteil**

Dabei beschreiben die drei Teile folgendes:

Benutzerkennung Die Benutzerkennung mit der sich der Benutzer authentifiziert hat.

Gruppe Das Feld ähnelt den Gruppen in Unix/Linux. Ein Benutzer konnte in beliebig vielen Gruppen Mitglied sein, musste sich jedoch bei der Prozesserzeugung für eine entscheiden. Ein Prozess kann also immer nur in einer Gruppe gleichzeitig sein. Ein Gruppenadministrator kann Benutzer in eine Gruppe aufnehmen oder löschen.

Abteil [8] Mit dieser Angabe konnte sich ein Benutzer selbst nochmals einschränken. Denkbar wäre zum Beispiel ein Abteil *streng vertraulich* einzurichten und alle sehr wichtigen Dokumenten nur zugreifbar zu machen, wenn

[8] engl. Compartment

man sich in dieses Abteil einloggt. Im normalen Betrieb könnte man dann nicht versehentlich wichtige Dokumente löschen, verändern oder anderen zugänglich machen.

Ein beispielhafter Name für einen Prozess wäre:

MustermannM : **Inventar** : **Normal**

Dabei hat sich der Benutzer mit der Kennung *MustermannM* eingeloggt und möchte in der Gruppe *Inventar* im Abteil *Normal* arbeiten.

Jeder Eintrag einer Zugriffskontrollliste besteht aus denselben drei Feldern, kann statt einem konkreten Wert aber auch ein Asterisk enthalten, womit für dieses Feld keine Einschränkung einhergeht. Mit diesem Mechanismus lassen sich schnell eine ganze Menge der üblichen Schutzbedürfnisse erfüllen.

Mit dem Eintrag

MustermannM : ***** : *****

bekommt nur ein Prozess des Benutzers mit der Kennung *MustermannM* Zugriff auf das Segment, jedoch unabhängig davon in welcher Gruppe und in welchem Abteil er eingeloggt ist.

Mit dem Eintrag

***** : **Inventar** : *****

bekommen alle Prozesse Zugriff, die in der Gruppe *Inventar* arbeiten, egal von welchem Benutzer und aus welchem Abteil [16].

4. SCHUTZKONZEPTE IM DATEISYSTEM

Multics war eines der ersten, wenn nicht das erste, Betriebssystem, dass ein hierarchisches Dateisystem implementiert hatte [12]. Heuzutage nutzt nahezu jedes moderne Betriebssystem hierarchische Dateisysteme. Da es noch einen eigenen Vortrag zum Thema Dateisystem in Multics geben wird, beschränke ich mich hier auf eine kurze Beschreibung der zugrundeliegenden Konzepts.

Eine Datei besteht aus einer geordneten Menge von Elementen[9]. Dateien können nur durch die Benutzung des Dateisystems erstellt, verändert und gelöscht werden. Zugriff auf den Inhalt einer Datei erfolgt über ihren symbolischen Namen und den Index in die Datei.

Beim Zugriff auf Dateien des Dateisystems gelten im Grunde dieselben Voraussetzungen, wie beim Zugriff von Segmenten im Adressraum. Inhalte sollen von anderen Benutzern komplett isoliert werden können und es soll möglich sein, anderen Benutzern kontrolliert Zugriff zu gewähren. Konsequenterweise greifen die Schutzkonzepte des Adressraums und des Dateisystems nahtlos ineinander.

Multics speichert für jede Datei des Dateisystems eine eigene Zugriffskontrollliste, so dass für jede Datei gesonderte Rechte vergeben werden können. Verknüpfungen auf Dateien erben die Zugriffskontrollliste der Datei, auf die sie zeigen. Ein Eintrag der Zugriffskontrollliste enthält den Benutzer (oder die Gruppe von Benutzern), die Zugriffsrechte und den Bereich in welchen Ringen diese Rechte gelten. Es gibt ein Ausnahmeattribut und vier verschiedene Zugriffsrechte: Lesen, Schreiben, Ausführen und Anhängen.

[9]Maschinenwörter, Zeichen, Bits, ...

Ist das Ausnahmeattribut in einem Eintrag gesetzt, werden beim Zugriff durch den zugehörigen Benutzer eine oder mehrere Ausnahmeprozeduren aufgerufen, die in einer Ausnahmeliste[10] verwaltet werden. Den Ausnahmeprozeduren werden alle relevanten Informationen übergeben (Name der aufrufenden Prozedur, Dateiname, ...). Der Rückgabewert spezifiert welche der vier Zugriffsrechte gelten sollen. Dabei können die Standardzugriffsrechte des Eintrags überschrieben werden. Ausnahmeprozeduren können dynamisch zur Laufzeit zur Ausnahmeliste hinzugefügt oder entfernt werden. Um Einträge zur Ausnahmeliste hinzufügen zu können, muss in der Zugriffskontrollliste des überliegenden Ordner das Schreiben Zugriffsrecht gesetzt sein. Durch das Ausnahmeattribut konnte das Standardrechtesystem leicht erweitert werden und zum Beispiel benutzerspezifische Sperren eingerichtet werden.

Die Bedeutung der vier Zugriffsrechte hängt von der Art des Eintrags ab, für den sie gelten (Datei oder Ordner). Die Bedeutung für Dateien entspricht den intuitiven Namen der Zugriffsrechte, also das Lesen Zugriffsrecht für das Recht zu Lesen usw. Es ist nur zu beachten, dass das Schreiben Zugriffsrecht kein Vergrößern der Datei und das Anhängen Zugriffsrecht kein Verändern des schon vorhandenen Inhalts erlaubt.

Die Bedeutung der Zugriffsrechte für Verzeichnisse ist weniger intuitiv. Das Lesen Zugriffsrecht ermöglicht die Auflistung aller Einträge des Ordners, also aller Dateien, Verknüpfungen und Unterordner. Es ist u.a. nötig, um herauszufinden, ob und welche Rechte man auf eine Datei hat. Das Ausführen Zugriffsrecht ermöglicht es in dem Verzeichnis zu suchen, um eine bestimmte Datei zu finden und seinem Adressraum hinzufügen zu können. Das Schreiben Zugriffsrecht erlaubt das Ändern vorhandener Einträge, also zum Beispiel das Hinzufügen neuer Ausnahmeprozeduren zu Einträgen, das Löschen von Einträgen usw., jedoch nicht das Hinzufügen neuer Einträge zu dem Verzeichnis. Dafür ist das Anhängen Zugriffsrecht nötig.

Die Rechte eines Verzeichnisses gelten immer nur für das Verzeichnis und dessen Einträge, nicht jedoch für darunterliegende Verzeichnisse. Um den Verwaltungsaufwand bei der Erzeugung neuer Dateien und Verzeichnisse zu reduzieren, gibt es zwei initiale Zugriffskontrolllisten (eine für Dateien, eine für Verzeichnisse), die neu erzeugten Dateien bzw. Verzeichnissen zugeordnet werden. Damit können Standardzugriffsrechte vergeben werden.

Eingebunden und zugreifbar wurde das Dateisystem, indem jede Datei als ein Segment gesehen werden konnte und umgekehrt jedes Segment als eine Datei. Damit ein Datei als Segment in den Adressraum eines Prozesses eingebunden werden durfte, muss es einen passenden Eintrag in der Zugriffskontrollliste geben. Die Zugriffsrechte aus dem Eintrag werden im SDW des Segments hinterlegt.

5. B2 SICHERHEITSFREIGABE

Das US-Militär stellte in den 1970er Jahre fest, dass alle existierenden Computersysteme nicht den Anforderungen genügten, die das Militär an den Schutz von sensiblen Daten stellte. Vor allem anfangs wurden deswegen jeder Anwendung, die sicher betrieben werden musste, ein eigenes Computersystem zugeteilt oder bevor eine neue Anwendung geladen werden konnte, das System komplett zurückgesetzt

[10]engl. *Trap List*

werden. Das war auf Dauer jedoch nicht praktikabel und das Militär wollte echten Mehrbenutzerbetrieb etablieren, wobei Personen unterschiedlicher Sicherheitsfreigabe auf demselben System arbeiten können sollten [17]. Es initiierte mehrere Projekte, die ein explizites Modell für den Schutz von Computersystemen bzw. den dort gespeicherten Daten entwickeln sollten. Die Ergebnisse mündeten im Dokument *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), dass 1983 fertiggestellt und 1985 überarbeitet und erneut veröffentlicht wurde [5]. Es beschreibt ein systematisches Vorgehen zur Bewertung der Sicherheit in Computersystemen. Umgangssprachlich wurde das Dokument wegen seines orangen Umschlags *Orange Book* genannt.

Bereits in den 1970er Jahren hatte Multics viele Sicherheitsfeatures implementiert. Trotz einiger bekannten Schwächen waren sich die Entwickler damals relativ sicher, dass eine korrekt installierte und administrierte Installation sicher betrieben werden konnte [16, S. 398ff.].

Allerdings sorgten einige der Schwächen dafür, dass es noch bis 1985 dauerte, bis Multics eine Zertifizierung durch das Militär bekam. Es gibt etliche Papiere und Artikel, in denen Fehler und Schwächen der Schutzkonzepte aufgearbeitet werden [10, 13, 16, 8, 9]. Die größte Schwäche war die Codebasis des Ring 0. Aus Leistungsgründen wurden viele Funktionen, die aus der Sicherheitsperspektive nicht in Ring 0 hätten laufen müssen, trotzdem in Ring 0 implementiert. Vor allem auf der GE-645, die noch keine Hardwareringe implementierte, war der Einfluss der Schutzringe auf die Leistung immens[11].

Unter anderem untersuchte die US Luftwaffe die Sicherheit von Multics und schaffte es 1977 eine Sicherheitslücke in der Entwicklungsinstallation auszunutzen, die auf einem Fehler eines Ring 0 Programmes beruhte. Sie konnten dauerhaft ein Programm installieren, dass jede Stelle im Speicher patchen konnte und jede Datei lesen konnte. Um die Machbarkeit zu demonstrieren, lasen sie die Passwortdatei aus, entschlüsselten die Passwörter und händigten Tom Van Vleck[12] während einer Besprechung sein eigenes Passwort aus [13].

Um die Ring 0 Codebasis zu reduzieren, wurde 1975 das *Project Guardian* geschaffen, dessen Aufgabe es war einen möglichst minimalen Sicherheitskernel zu schaffen, der komplett prüffähig gewesen wäre. Das Projekt schaffte es die Ring 0 Codebasis zu halbieren[13] und es gab sogar noch weiteres Potenzial Code zu verschieben. Allerdings war der Einfluss auf die Leistung schon zu diesem Zeitpunkt so immens, dass die Ergebnisse des Projekts nicht in das Produkt einflossen und das Projekt 1976 schließlich wieder eingestellt wurde.

Im Orange Book werden verschiedene Zertifizierungsstufen beschrieben: D, C1, C2, B1, B2, B3, A1. D stellt die niedrigste mögliche Stufe dar und wird vergeben, wenn ein System keiner anderen Stufe genügt. A1 stellt die höchstmögliche Stufe dar und fordert ein formal beweisbar sicheres Betriebssystem. Multics erlangte die Zertifizierung B2, die u.a. ein klar definiertes und dokumentiertes Sicherheitsmodell und eine verbindliche Zugriffskontrolle auf alle Objekte vorsieht. Darüber hinaus implementierte Multics auch einige, aber nicht alle, Anforderungen aus der B3 Zertifizierung, zum Beispiel Zugriffskontrolllisten [18, 10]. So fehlte für die B3 Zertifizierung zum Beispiel der komplett prüfbare Sicherheitskernel, der das Ziel des Project Guardian gewesen ist.

Seit 1996 werden Systeme nicht mehr nach Kriterien des Orange Book bewertet, sondern nach der *Common Criteria for Information Technology Security Evaluation* (CC)[14] [2]. Die CC stellen eine Harmonisierung der us-amerikanischen, kanadischen und europäischen Bewertungssystemen dar und sollen Mehrfachzertifizierungen vermeiden.

Die Vorgehensweise der CC ist grob in zwei Teile untergliedert. In eine produktunabhängige Erstellung eines Sicherheitsprofils anhand der geplanten Funktionalität. Aus dem Profil werden dann konkrete Sicherheitsanforderungen an das Produkt abgeleitet und deren Implementierung evaluiert. Das *Evaluation Assurance Level* (EAL) gibt an, wie tiefgehend diese Evaluierung ist und entspricht etwa den Sicherheitsleveln des Orange Books, wobei EAL1 etwa D bis C1 und EAL7 etwa A1 entspricht. Die Kombination der abgeleiteten Sicherheitsanforderungen und der EAL gibt die Vertrauenswürdigkeit eines Produkts an. Die CC sollen für alle IT Produkte geeignet sein, so dass damit nicht nur Betriebssysteme, sondern auch Applikationen, Hardware, usw. evaluiert werden können.

Da durch die formale Vorgehensweise eine Zertifizierung mit relativ viel Aufwand verbunden ist, gab und gibt es Kritik an den CC. Insbesondere in den Anfangsjahren führte das zu wenigen Zertifizierungen. Im Zeitraum von 1999 bis inklusive 2006 gab es insgesamt nur 36 zertifizierte Produkte, 19 davon erreichten ein zu Multics vergleichbares Sicherheitslevel[15].

Heute sind Zertifizierungen jedoch in der Geschäftswelt unumgänglich, weswegen die Zahl der Zertifizierungen förmlich explodierte und von 1999 bis heute knapp 2000 Zertifizierungen ausgestellt wurden, wobei jedoch nur ein Produkt die höchste Sicherheitsstufe EAL7+ erreicht hat[16]. Maßgeblich treibend für die Entwicklung für zertifizierte Sicherheit sind Deutschland (591 Zertifizierungen) und Frankreich (518 Zertifizierungen), die zusammen knapp über 50 Prozent aller Zertifizierungen ausmachen.

Die höchste Zertifizierung im Bereich Betriebssysteme liegt bei EAL5+, was in etwa dem Niveau von Multics entspricht. Allerdings sind das in der Regel Spezialsysteme[17] und keine Universalbetriebssysteme wie Multics [1].

6. FAZIT UND AUSBLICK

Diese Arbeit hat einige grundlegende Schutzkonzepte von Multics vorgestellt. Die vorgestellten Konzepte veranschaulichen, dass Multics eines der ersten, wenn nicht sogar das erste, Betriebsystem ist, das versucht hat ein ganzheitliches Schutzkonzept zu entwickeln. Viele der damals identifizierten Anforderungen, wie zum Beispiel der sichere Mehrbenutzerbetrieb, die Isolation bzw. das kontrollierte Freigeben von Daten und viele andere sind bis heute gültig. Faszinierend ist, dass einige der Konzepte, die Multics für den Schutz

[11] In einer Version von 1972 dauerte ein Ringwechsel auf der GE-645 hin und zurück 2-3ms [14, Seite 132]

[12] Tom Van Vleck war der Entwickler, der u.a. den Code für die Verschlüsselung der Passwörter implementiert hatte.

[13] Zu Anfang hatte der Ring 0 und vertrauenswürdige Prozesse etwa 55.000 Codezeilen.

[14] kurz für: *Common Criteria*

[15] EAL4 oder höher

[16] Ein Ein-Wege Kommunikationssystem [6]

[17] v.a. verschiedene Versionen des PR/SM Hypervisor für IBM Z Systeme

eingesetzt hat, bis heute überdauert haben.

Neben diesen überdauerten Konzepten sind aber die Fehler und Schwächen, die während der Entwicklung von Multics gemacht wurden, fast ebenso spannend. Sind die Schwächen eines Schutzsystems bekannt, können geeignete Maßnahmen getroffen werden, um diese Schwächen auszugleichen. Sind sie nicht bekannt, führt das zu konkret ausnutzbaren Sicherheitslücken.

Zusammenfassend war die Entwicklung von Multics ein wichtiger Schritt hin zu sichereren Betriebsystemen im Speziellen und Computersystemen im Allgemeinen war.

7. LITERATUR

[1] Common Criteria Certified Products List - Statistics. http://www.commoncriteriaportal.org/products/stats/, 2015. [Online; abgerufen am 14. Dezember 2015].

[2] Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/, 2015. [Online; abgerufen am 14. Dezember 2015].

[3] R. C. Daley and J. B. Dennis. Virtual memory, processes, and sharing in Multics. *Communications of the ACM*, 11(5):306–312, 1968.

[4] R. C. Daley and P. G. Neumann. A general-purpose file system for secondary storage. In *Proceedings of the Fall Joint Computer Conference*, pages 213–229. ACM, 1965.

[5] Department of Defense. Department of defense trusted computer system evaluation criteria. csrc.nist.gov/publications/history/dod85.pdf, 1985. [Online; abgerufen am 06. Dezember 2015].

[6] Fox IT. DataDiode. https://www.fox-it.com/datadiode/, 2015. [Online; abgerufen am 14. Dezember 2015].

[7] R. M. Graham. Protection in an information processing utility. *Communications of the ACM*, 11(5):365–369, 1968.

[8] P. A. Janson. Dynamic linking and environment initialization in a multi-domain process. *SIGOPS Operating Systems Review*, 9(5):43–50, 1975.

[9] P. Karger, R. R. Schell, et al. Thirty years later: Lessons from the Multics security evaluation. In *Proceedings of the Computer Security Applications Conference*, volume 18, pages 119–126. IEEE, 2002.

[10] Multicians. B2 security evaluation. http://www.multicians.org/b2.html, 2015. [Online; abgerufen am 24. November 2015].

[11] Multicians. Execution environment. http://www.multicians.org/exec-env.html, 2015. [Online; abgerufen am 01. Dezember 2015].

[12] Multicians. Features. http://www.multicians.org/features.html, 2015. [Online; abgerufen am 25. November 2015].

[13] Multicians. Security. http://www.multicians.org/security.html, 2015. [Online; abgerufen am 24. November 2015].

[14] E. I. Organick. *The Multics system: an examination of its structure*. MIT press, 1972.

[15] J. Rabenstein. Paging in Multics. https://www4.cs.fau.de/Lehre/WS15/MS_AKSS/slides/06-Paging-Jonas-Rabenstein.pdf, 2015.

[16] J. H. Saltzer. Protection and the control of information sharing in Multics. *Communications of the ACM*, 17(7):388–402, 1974.

[17] R. Schell. Oral history interview with Roger R. Schell. http://purl.umn.edu/133439, 2012. [Online; abgerufen am 01. Dezember 2015 von der Universität von Minnesota (Digital Conservancy)].

[18] R. R. Schell. Evaluating security properties of computer systems. *Symposium on Security and Privacy*, pages 89–89, 1983.

[19] M. D. Schroeder and J. H. Saltzer. A hardware architecture for implementing protection rings. *Communications of the ACM*, 15(3):157–170, 1972.