Universität Heidelberg
Sommersemester 2019

Lehrstuhl für Parallele und Verteilte Systeme (PVS)
Diego Costa, Artur Andrzejak

6. Übung zur Vorlesung „Betriebssysteme und Netzwerke“ (IBN)

Abgabedatum: 04.06.2019, 11:00 Uhr

Aufgabe 1 (2 Punkte)

In der Vorlesung wurden invertierte Seitentabellen vorgestellt. Diese haben den Vorteil, dass sie bei großem virtuellen Adressraum (z.B. 64 Bit) die Seitentabelle klein halten. Ein Eintrag in der invertierten Seitentabelle enthält aber keine Informationen über den Status einer Seite (im Hauptspeicher oder nicht). Nennen und diskutieren Sie die möglichen Konsequenzen dieser Eigenschaft.

Aufgabe 2 (2 Punkte)

Betrachten Sie ein System mit virtuellem Speicher, das drei Seitenrahmen verwendet, die anfangs leer seien. Geben Sie bei folgenden Aufgaben Ihre Rechenschritte an.

a) Es werden nacheinander Speicheradressen aus folgenden Seiten angefragt (in dieser Reihenfolge): $3, 0, 1, 2, 1, 3, 2, 0, 3$. Geben Sie für folgende zwei Seitenersetzungsstrategien an, welche drei Seiten nach Ende der Anfragen im Speicher geladen sind: FIFO bzw. Least Recently Used (LRU).

b) Geben Sie ein einfaches Beispiel einer Seitenzugriffsreihenfolge an, bei der die erste Seite, die als Opfer ausgewählt wird, für Clock und LRU unterschiedlich ist. Nehmen Sie an, die Seitenzugriffsreihenfolge Seitennummern aus der Menge $\{0, \ldots, 9\}$ enthält.

Aufgabe 3 (2 Punkte)

Ein Maschinenbefehl, der ein 32-Bit-Wort aus dem Speicher in ein Register lädt, enthält die Speicheradresse des Wortes, das geladen werden soll. Wie viele Seitenfehler kann der Befehl maximal auslösen? Erklären Sie, wie Sie auf diese Anzahl kommen.

Aufgabe 4 (1 Punkt)

Wie hängt die Rate der Seitenfehler des optimalen Algorithmus OPT mit der Seitenfehlerrate von LRU zusammen? Hinweis: Suchen Sie eine Ähnlichkeit zwischen den beiden Algorithmen.

Aufgabe 5 (6 Punkte)

Implementieren Sie ein Programm, dass Seitenersetzungsalgorithmen simulieren kann. Als Parameter erwartet es die Anzahl an Seitenrahmen und als Input Referenzfolgen von Seitenzugriffen. Das Programm soll in die Standardausgabe nach und nach eine Repräsentation des jeweiligen Status der Seitenrahmen ausgeben. Implementieren Sie die Strategie *Clock* und starten Sie das Programm jeweils einmal mit den Referenzfolgen A und B aus Vorlesung 11 für drei Seitenrahmen. Reichen Sie den Quelltext als auch ein Log der Programmausgaben als Lösung ein.

Aufgabe 6 (1 Punkt)

Kann eine Seite gleichzeitig zu zwei verschiedenen Arbeitsbereichen (working sets) gehören? Warum, bzw. warum nicht?

Aufgabe 7 (2 Punkte)

Geben Sie Pseudocode für ein Programm an, das ein größtes Working Set (WS) in einer Referenzfolge von Seitenzugriffen berechnet, wobei die Definition eines WS aus der Vorlesung 12 verwendet werden soll. In Detail: das Programm bekommt als Eingabe die Anzahl Δ der letzten zu berücksichtigenden Speicherzugriffe und den Dateinamen einer Datei mit einer Referenzfolge (z.B. pro Zeile: ein Index der zugegriffenen Seite). Es liest die Datei ein, verarbeitet diese, und gibt ein größtes WS (für Δ) aus. Geben Sie die relevanten Datenstrukturen und Variablen an. Bei den I/O-Operationen können Sie dagegen vereinfachen, z.B. reicht die Beschreibung „lese Inhalt der Datei mit Dateinamen *name* in ein Array *pageAccesses* ein“.

Aufgabe 8 (Bonus, 4 Punkte)

In den letzten Jahren ist ein Problem mit DRAM-Arbeitsspeicher bekannt geworden, das den Namen „Rowhammer” bekommen hat. Das Team von *Google Project Zero*, das sich mit Sicherheitslücken beschäftigt, veröffentlichte ein proof of concept[1] zur Ausnutzung.

Es werden zwei *Exploits* vorgestellt - wir betrachten nur den zweiten (*kernel privilege escalation,* über manipulierte Seitentabellen).

a) Lesen Sie die Abschnitte *Overview* und *Introduction to the rowhammer problem.* Welches unerwünschte Phänomen bei der Benutzung des DRAM-Hauptspeichers wird ausgenutzt? Beschreiben Sie, welche Aktionen dieses Verhalten auslösen können und welche Nebenbedingungen ein gezielter Angriff einhalten muss.

[1] http://googleprojectzero.blogspot.de/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

b) Lesen Sie die Abschnitte *Kernel privilege escalation* (ohne den Unterabschnitt *Break it down: exploit steps*) und *Exploiting write access to page tables.* Angenommen, Sie können gezielt Bits Ihrer Wahl flippen. Wie wird diese Fähigkeit vom vorgestellten Exploit ausgenutzt? Erklären Sie in eigenen Worten und einem Diagramm, was schrittweise erreicht wird. Welchem Prozess gehört der angegriffene Speicherbereich?

c) Warum reicht es, nur einzelne Bits zu manipulieren? Auf welcher Art von Seitentabellen basiert der Angriff, und würde er auch mit anderen funktionieren? Können andere Prozesse oder das Betriebssystem etwas vom Angriff mitbekommen?