Aktuelle Herausforderungen in der Datenbankforensik

Martin Mulazzani, Edgar Weippl

Zusammenfassung

Datenbankforensik in der heutigen Form vernachlässigt bei der forensischen Untersuchung einige vielversprechende zusätzliche Datenquellen. Wir präsentieren mögliche weitere Datenquellen wie den Index einer Datenbank und zeigen aktuelle Probleme bei der Dateisystemforensik auf.

Schlüsselwörter

Datenbankforensik, B-Baum, Index, Computerforensik

1 Einleitung

Die digitale Forensik hat in den letzten Jahren durch die zunehmende Verbreitung von PCs bei der Untersuchung von Verbrechen stark an Bedeutung gewonnen [3]. Der Inhalt einer Festplatte, also die Dateien, die sich auf dem Computer befinden, sind dabei für Ermittler von besonderem Interesse. Viele Informationen werden, vom Anwender unbemerkt, gespeichert bzw. ist eine gelöschte Datei oft auf der Festplatte nachweisbar oder rekonstruirbar [4]. Wichtige Hinweise können, je nach Szenario, auch der Inhalt des Arbeitsspeichers oder die momentan laufenden Prozesse liefern.

Bei der forensischen Untersuchung von Datenbanksystemen wurden jedoch bis heute viele nützliche Informationen gänzlich außer Acht gelassen. Die Analyse auf der Dateisystemebene ist in vielen Fällen nicht ausreichend, um manipulierte Informationen und deren Auswirkungen zu erkennen. Besonders bei großen Datenbanken, die ihre Dateien oft auf der Festplatte aktualisieren, können Manipulationen im geringen Ausmaß leicht unbemerkt bleiben, was möglicherweise weitreichende Konsequenzen hat. Ein Beispiel:

Ein Angreifer kann in einem Onlineshop seine Rechnungsinformationen manipulieren, zum Beispield mittels SQL-Injection [1]. Damit könnte er seine bestellten Waren als bezahlt markieren, weitere Waren bestellen, ohne zu bezahlen oder dem Unternehmen auf andere Arten erheblichen Schaden zufügen. Würde ein solcher Server nur auf Dateisystemeben untersucht werden, so wären diese Manipulationen sehr schwierig zu entdecken.

Unser Beitrag im Rahmen dieses Papers ist, die folgenden datenbankspezifischen Detailinformationen zu verwenden, um tiefergehende Manipulationen in Datenbanken zu erkennen:

- Verwendung von Datenbankindizes in der forensischen Untersuchung von Datenbanken,
- Erkennen von Manipulationen,
- Aggregation datenbankspezifischer, heterogener Daten für detaillierte Untersuchungen,
- Vergleich der aktuellen mit „historischen“ Informationen (wie Backups oder Snapshots) zur Erkennung von Manipulationen.

In Sektion 2 geben wir einen Überblick über unseren Ansatz, in Sektion 3 erläutern wir die technischen Details. Inwieweit Datenbankindizes nützlich sein können, wird in Sektion 3.1 gezeigt. Abschließend verweisen wir auf ähnliche Arbeiten auf diesem Gebiet in Kapitel 4.

2 Digitale Forensik

Das Ziel der Computerforensik ist es, bei der Aufklärung von Verbrechen zu helfen, bei denen suf irgendeiner Art und Weise ein Computer verwendet wurde, weiters den bzw. die Verbrecher zu identifizieren, die durchgeführten Handlungen nachzuvollziehen und Beweise für mögliche juristische Schritte zu sichern [10]. Sowohl die Rekonstruktion der gesetzten Schritte und zeitlichen Abläufe als auch die verwendeten Programme und ausgenutzten Schwachstellen sind bei Systemeinbrüchen von Interesse. Abgesehen von der Beantwortung spezifischer Fragen im Rahmen einer Ex-Post-Analyse können die Erkenntnisse auch zur Prävention verwendet werden. So können Schwachstellen beseitigt und veränderte Informationen rückgängig gemacht werden. Besonders Firmen haben daran ein besonders starkes Interesse, da manipulierte Informationen großen wirtschaftlichen Schaden anrichten können.

Oftmals ist eine eingehende Analyse der veränderten Informationen in ihrem jeweiligen Kontext nötig. Gefälschte Emails müssen anders untersucht

werden als Rootkits oder Sniffer, Server anders als Workstations. Wenn der Angreifer ein Mitarbeiter des Unternehmens war oder ist, standen ihm andere Möglichkeiten offen als von außerhalb. All diese Informationen bezeichnen den Kontext einer Untersuchung. Die verschiedenen Gebiete wie Recht, Hardware und Software müssen gesamtheitlich untersucht werden. Dies macht die Forensik zu einem sehr interessanten und breit gefächerten Gebiet.

Bei Datenbanksystemen gibt es zwar bereits Literatur zur forensischen Untersuchung, diese ist jedoch stark auf die Hersteller der Datenbank zugeschnitten (Details siehe Kapitel 4). Unser Ansatz unterscheidet sich darin, dass er auf fast alle Datenbanksysteme anwendbar ist: Die Verwendung einer vielen Datenbanken zugrundeliegenden Datenstruktur, der B*-Baum wird als zentrales Artefakt angenommen. Der B-Baum [5] ist ein balancierter Suchbaum und ermöglicht die effiziente Suche und Manipulation von Daten. Viele Datenbanksysteme verwenden als grundlegenden Datenbankindex einen B*-Baum [14], und er ist für alle möglichen Arten moderner Datenbank-Indizes einsetzbar [12]. Unser Ansatz ist, diese B-Bäume in die forensischen Untersuchungen einzubinden.

Die Grundlage dafür ist die Tatsache, dass verschiedene Anordnungen desselben Inputs in verschieden Bäumen resultieren. Die Details dazu werden in Kapitel 3.1 erklärt. Folgende Daten lassen sich damit, zusätzlich zu momentan gängigen Methoden, aus Datenbanken extrahieren:

- Überprüfung der Datenbank auf unzulässige Manipulationen,
- zeitliche Einordnung der Manipulationen,
- Kombination und Aggregation der gewonnenen Daten mit weiteren Daten, zum Beispiel Logfiles, SQL Abfrage Cache, Dateisystem, etc.

3 Datenstrukturen in der Datenbankforensik

Zuerst erläutern wir, wie B-Bäume in der Datenbankforensik verwendet werden, anschließend zeigen wir, wie „historische“ Informationen (Backups, Logfiles oder Snapshots) bei Aussagen über den zeitlichen Rahmen einer unerlaubten Manipulation helfen können.

3.1 B-Bäume

Ein B-Baum besteht aus einer Wurzel und beliebig vielen Kinderknoten. Jeder Knoten hat höchstens t Kinder, wobei t auch die ‘“Ordnung“ des B-Baumes genannt wird. In jedem Knoten werden Schlüssel gespeichert, wobei

die Schlüssel die Unterbäume anhand ihrer Werte unterteilen. Dadurch, dass jeder Knoten höchstens t Kinder hat, enthält jeder Knoten höchstens $t-1$, aber zumindest $\lceil \frac{t}{2} \rceil - 1$ Schlüssel. B-Bäume sind balanciert, alle Blätter (das sind die Knoten ohne Kinder) sind auf einer Ebene.

Schlüssel werden unterhalb der Wurzel eingefügt. Wenn ein Schlüssel in einen Knoten eingefügt werden soll, dieser aber schon t Knoten enthält, dann wird der Knoten aufgespalten. Der mittlere Schlüssel wandert eine Ebene näher zur Wurzel, die beiden Teilknoten werden zu seinen Kindern. Der einzufügende Schlüssel wird dann in dem entsprechenden Kind integriert. Wenn der sog. Elternknoten auch schon t Knoten enthält wird weiter aufgespalten in Richtung Wurzel, und so weiter bis ein Knoten noch nicht t Schlüssel enthält. Sollte auch die Wurzel mit t Schlüssel gefüllt sein, wird die Wurzel geteilt und der mittlere Schlüssel der alten Wurzel wird zur neuen Wurzel. B-Bäume wachsen im Gegensatz zu Binärbäumen von unten nach oben.

Das bedeutet aber auch, dass die Reihenfolge, in der die Schlüssel eingefügt werden, die Form des Baumes bestimmt. Je nach Inputpermutation werden Knoten unterschiedlich aufgespalten, wodurch sich bei denselben Schlüsseln in unterschiedlicher Reihenfolge unterschiedliche Bäume ergeben können. Als Beispiel haben wir in einen B-Baum der Ordnung 5 die Zahlen 1 bis 20 in zwei verschiedenen Permutationen eingefügt, die daraus entstandenen Bäume werden in den Abbildungen 1 und 2 gezeigt. Es gibt noch weitere Inputpermutationen welche anders aussehende Bäume entstehen lassen; die genaue Anzahl an möglichen Bäumen bietet Platz für weitere Untersuchungen. Die Anzahl der möglichen Inputpermutationen ist die Fakultät der Anzahl der Elemente, in unserem Fall von gerade einmal 20 Schlüsseln entspricht das bereits $20! = 2{,}4 \cdot 10^{18}$ Permutationen. Bei einem Baum der Ordnung t mit n Elementen gibt es mindestens $\frac{n}{\lceil \frac{t}{2} \rceil - 1}$ Knoten und maximal $\frac{n}{t-1}$ Knoten

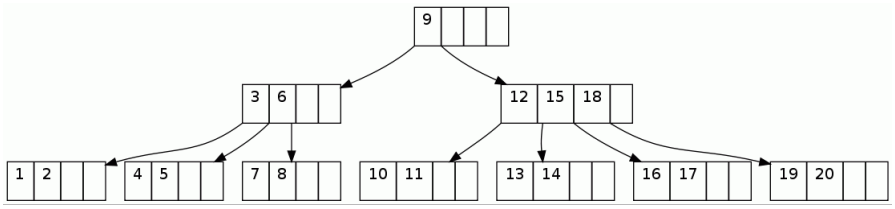

Abbildung 1: Inputpermutation 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

Um diese Informationen forensisch ausnützen zu können, betrachtet man den Baum für sich, und kann anhand der Form des Baumes bzw. der Anordnung der Schlüssel innerhalb des Baumes feststellen, ob der Baum „natürlich“ gewachsen ist. Ein Beispiel: Wenn ein Wert gelöscht und danach wieder eingefügt wird, kann der Baum nachher anders aussehen als wenn der Wert nicht

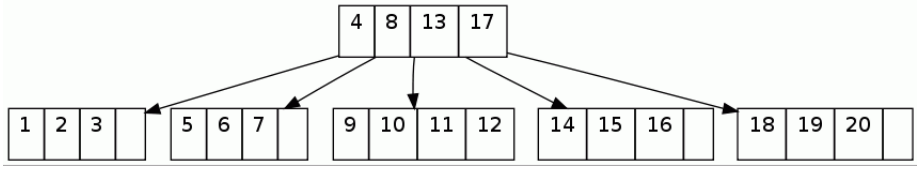

Abbildung 2: Inputpermutation 8,16,5,13,3,4,11,12,6,14,1,9,10,7,17,20,18,2,19,15

gelöscht worden wäre. Abbildung 3 zeigt den B-Baum nach dem Einfügen der Werte 1 bis 6. In Abbildung 4 wurde der Wert 2 gelöscht, wodurch der Baum nicht mehr balanciert ist. Abbildung 5 zeigt den Baum nach der Rebalancierung. Wenn jetzt der Wert 2 wieder eingefügt wird, entsteht ein anders aussehender Baum als der ursprüngliche - siehe Abbildung 6.

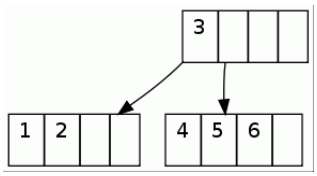

Abbildung 3: Inputpermutation 1,2,3,4,5,6

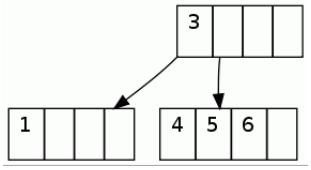

Abbildung 4: Schlüssel 2 gelöscht

Ein Szenario, bei dem dieses Beispiel angewandt werden könnte: Angenommen ein Onlineshop speichert die Kundendaten in einer Datenbank, wobei die Kundennummer eine von der Applikation vergebene, fortlaufende Nummer ist. Ein Angreifer gibt eine Bestellung auf und wartet einige Zeit, bis weitere Kunden angelegt werden. Dann löscht er auf unbefugtem und für die Firma unbemerktem Weg seine Kundendaten aus der Datenbank. Sofern die Applikation die Kundennummer als „nicht vergeben“ entdeckt, wird der nächste Kunde mit der Kundennummer des Angreifers angelegt. Die Mahnungen für die unbezahlte Rechnung des Angreifers werden dann an den neuen Kunden geschickt.

Insbesondere durch Verwendung zusätzlicher historischer Informationen wie Backups, Logfiles oder Snapshots lassen sich die Aussagen verfeinern, so

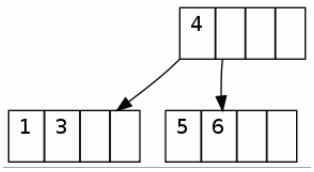

Abbildung 5: Baum wird rebalanciert

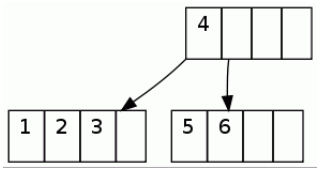

Abbildung 6: Schlüssel 2 wieder eingefügt

in Kapitel 3.3 beschrieben. Andere Baumstrukturen, wie der B+-Baum oder der B*-Baum könnten möglicherweise ähnliche Eigenschaften besitzen, um forensisch verwertbare Metadaten zu gewinnen.

3.2 Offene Punkte in der Dateisystemforensik

Eine offene Problemstellung in der Dateisystemforensik ist u.a. die Verwendung von alternativen Dateisystemen. Linux zum Beispiel unterstützt mehr als 50 verschiedene Dateisysteme, wobei die gängige Software zur forensischen Untersuchung von Festplatten wie *EnCase* [8] oder *The Sleuth Kit* [16] nur wenige davon verarbeiten können. *The Sleuth Kit* zum Beispiel unterstützt nur die gängigen Dateisysteme FAT, NTFS, ext2, ext3 und ISO 9660, neue und experimentelle Dateisysteme wie Btrfs oder ext4 fehlen. IBMs JFS wurde von Knut Eckstein untersucht [6].

Viele aktuelle sowie Dateisysteme, die gerade entwickelt werden, verwenden Baumstrukturen, um die Zugriffszeiten auf die Dateien zu minimieren. Diese Bäume sollten sich auch in der oben beschriebenen Art und Weise verwenden lassen, um im Zuge einer forensischen Untersuchung zusätzliche Informationen zu erhalten. Die folgenden, wichtigen Dateisysteme verwenden Baumstrukturen:

- Das Macintosh *HFS Dateisystem* verwendet B*-Bäume [11].
- *Btrfs* (wird von Oracle entwickelt und soll ext3 als Standarddateisystem unter Linux ablösen) verwendet B-Bäume als grundlegende Struktur [2]. Seit Jänner 2009 ist es offiziell Teil des Linuxkernels und unterstützt zusätzliche Features wie Dateisystemsnapshots, welche bei der

Aggregation historischer Daten besonders interessant sind.

- *Reiser3* und *Reiser4* verwenden B+-Bäume [15].

3.3 Aggregation historischer Daten

Die forensische Aussagekraft der B-Bäume kann noch gesteigert werden, wenn „historische“ Informationen zur Verfügung stehen. Beispiele für solche Informationen sind:

- Backups der Festplatten,
- Backups der Datenbank,
- Snapshots des Dateisystems, zum Beispiel wenn der Server in einer virtuellen Maschine betrieben wird oder bei Dateisystemen die Snapshots ermöglichen, zum Beispiel Btrfs,
- Logfiles.

Diese Informationen können sehr heterogen sein und von unterschiedlichen Quellen wie dem Betriebssystem, der Datenbank oder der Software, die auf der Datenbank aufbaut, stammen. Da Datenbankindizes sehr groß und umfangreich werden können, könnten diese Zustandsinformationen zu früheren Zeitpunkten den Suchraum verringern bzw. die zeitliche Einordnung erleichtern.

Ziel einer forensischen Untersuchung muss es dennoch sein, den zeitlichen Ablauf zu dokumentieren, wofür diese heterogenen Informationen sehr wichtig sind. Inwieweit sie sich automatisiert aggregieren lassen, wurde bis jetzt in der Literatur nicht untersucht.

4 Verwandte Arbeiten

Hier erläutern wir kurz die wichtigsten Arbeiten aus den Gebieten Computerforensik und Datenbankforensik. Auf dem Gebiet der Dateiystemforensik wurden *journaling file systems* untersucht und wie sich Daten im Journal verstecken lassen [7]. Bei der forensischen Analyse von Datenbanksystemen gibt es Bücher über den Microsoft SQL Server [9] und von Oracle [18]. Untersucht wurde auch, wie sich die forensische Analyse von Datenbanken auf die Privacy auswirken kann [17]; eine Übersicht über Analyse von Datenbankforensik wurde in [13] gemacht.

5 Zusammenfassung

Die hier vorgestellte Verwendung von Indexstrukturen wie dem B-Baum bei Datenbanken bietet interessante Möglichkeiten in der forensischen Analyse von Datenbanksystemen. Insbesondere wenn frühere Snapshots der Datenbank oder des Dateisystems zur Verfügung stehen, lassen sich unberechtigte Manipulationen zeitlich gut einordnen. Inwieweit die hier vorgestellten Ideen auf komplexe Datenbanksysteme wie MySQL, Microsoft SQL Server oder Oracle anwendbar sind, ist Gegenstand zukünftiger Untersuchungen.

Literatur

[1] Chris Anley. Advanced SQL Injection In SQL Server Applications, 2002. `http://www.nextgenss.com/papers/advanced_sql_injection.pdf`.

[2] Btrfs File System. `http://btrfs.wiki.kernel.org/`.

[3] Michael A. Caloyannides, Nasir Memon, and Wietse Venema. Digital forensics. *IEEE Security and Privacy*, 7(2):16–17, 2009.

[4] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley, 2005.

[5] Douglas Comer. The ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, 1979.

[6] Knut Eckstein. Forensics for Advanced UNIX File Systems. In *Proceedings of the 5th IEEE information Assurance Workshop*, 2004.

[7] Knut Eckstein and Marko Jahnke. Data Hiding in Journaling File Systems. In *DFRWS*, 2005.

[8] EnCase. `http://www.guidancesoftware.com/`.

[9] Kevvie Fowler. *SQL Server Forensic Analysis*. Addison-Wesley Longman, 2009.

[10] Alexander Geschonneck. *Computer Forensik*. dpunkt.verlag, 2006.

[11] Dominic Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufmann Publishers, Inc., 1998.

[12] Beng Chin Ooi and Kian-Lee Tan. B-trees: bearing fruits of all kinds. In *ADC '02: Proceedings of the 13th Australasian database conference*, pages 13–20, 2002.

[13] Kyriacos E. Pavlou and Richard T. Snodgrass. Forensic analysis of database tampering. *ACM Trans. Database Syst.*, 33(4):1–47, 2008.

[14] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. Mcgraw-Hill Professional, 2002.

[15] Hans Reiser. Kernel korner: trees in the reiser4 filesystem, part i. *Linux J.*, 2002(104):8, 2002.

[16] The Sleuth Kit. `http://sleuthkit.org/`.

[17] Patrick Stahlberg, Gerome Miklau, and Brian Neil Levine. Threats to privacy in the forensic analysis of database systems. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 91–102, 2007.

[18] Paul M. Wright. *Oracle Forensics: Oracle Security Best Practices*. Rampant Techpress, 2009.