Firebird Datei- und Metadatensicherheit

Geoff Worboys;Daniel Albuschat;Martin Köditz

Version 0.6-de, 08. Juli 2020

Table of Contents

Chapter 1. Einleitung

Wenn Sie über diese Seite gestolpert sind und nichts über Firebird wissen, nutzen Sie bitte diesen Link: www.firebirdsql.org

Dieser Artikel beschreibt die Sicherheit von Firebird-Datenbankdateien und insbesondere den Zugriff auf die in diesen Dateien gespeicherten Metadaten. Er wurde als Antwort auf häufige Fragen in den Firebird-Listen zu diesen Themen geschrieben. Der Artikel vermeidet technische Besonderheiten. Informationen zum Sichern von Dateien in Ihrem jeweiligen Betriebssystem finden Sie in der entsprechenden Dokumentation zu diesem Betriebssystem. Informationen zum Anwenden der Benutzer- und Objektsicherheit in Firebird-Datenbanken finden Sie in der Dokumentation auf der Firebird-Website (siehe oben) oder dem käuflich erwerbbaren *The Firebird Book* von Helen Borrie.

Chapter 2. Hintergrund

Damit eine Anwendung (ein Benutzer) auf eine Firebird-Datenbank zugreifen kann, muss sie eine Verbindung zum Firebird-Serverprozess herstellen. Beim Empfang einer Verbindungsanforderung authentifiziert der Serverprozess die Benutzeranmeldeinformationen gegenüber den in der Sicherheitsdatenbank definierten Serverbenutzern. Wenn die Authentifizierung erfolgreich ist, gewährt der Server der Anwendung Zugriff auf jede angeforderte Datenbank und verwendet dann die in dieser Datenbank definierten Rollen und Berechtigungen, um eine differenzierte Zugriffssteuerung für Objekte in dieser Datenbank bereitzustellen.

Der Benutzer, der eine Verbindung zur Datenbank herstellt, benötigt zu keinem Zeitpunkt direkten Zugriff auf die Datenbankdatei selbst. Der gesamte Zugriff erfolgt über den Serverprozess, der bei Bedarf auf die Datenbankdatei zugreift, um Anforderungen zu erfüllen. Es ist der Server, der den Zugriff auf den angemeldeten Benutzer gemäß den für diesen Benutzer in dieser Datenbank definierten Berechtigungen einschränkt oder zulässt.

> Die “eingebettete” (engl. embedded) Variante des Firebird-Servers funktioniert anders und ist für sichere Installationen nicht geeignet. Die Tatsache, dass die eingebettete Serverversion vorhanden ist, ändert nichts an den in diesem Artikel behandelten Sicherheitsproblemen. Sie unterstreicht lediglich die Bedeutung der Verwendung effektiver Umweltsicherheit in sicheren Installationen. Der eingebettete Server wird später ausführlicher erläutert.

Jede Firebird-Serverinstallation hat einen “SYSDBA” -Benutzer. Dieser Benutzer hat uneingeschränkten Zugriff auf alle für diesen Server verfügbaren Datenbanken. Datenbankspezifische Berechtigungen werden beim Zugriff auf die Datenbank mit SYSDBA effektiv ignoriert. Wenn Sie eine Datenbank zum ersten Mal auf einen Server kopieren, können Sie mit SYSDBA die Berechtigungen an die Benutzer und Anforderungen des Servers anpassen. Dies bedeutet jedoch auch, dass ich, wenn ich direkten Zugriff auf eine Datenbankdatei habe, diese Datei von einem Server kopieren kann, auf dem ich das SYSDBA-Kennwort möglicherweise nicht kenne, auf einen anderen Server, auf dem ich das SYSDBA-Kennwort kenne, und somit uneingeschränkten Zugriff auf sie erhalte die Datenbank.

Wie Sie dieser Beschreibung entnehmen können, basiert die Firebird-Sicherheit auf der Annahme, dass der Firebird-Serverprozess in einer ausreichend sicheren Umgebung ausgeführt wird. Firebird selbst trifft keine Vorkehrungen, um externe Sicherheit zu gewährleisten. Sobald eine Person physischen Zugriff auf eine Datenbankdatei hat, gibt es keine effektive Möglichkeit, diesen Benutzer daran zu hindern, alle Daten (und Metadaten) in dieser Datei zu lesen.

“Angemessene” Sicherheit hängt von der Sicherheitsstufe ab, die für eine bestimmte Installation erforderlich ist. Dies ist von Installation zu Installation sehr unterschiedlich.

Dies bedeutet, dass Installationen aus Gründen einer angemessenen Sicherheit sicherstellen sollten, dass die Datenbankdateien angemessen gesichert sind. In den meisten Fällen bedeutet dies, dass der Firebird-Serverprozess als sein eigener spezifischer Betriebssystembenutzer ausgeführt werden sollte und nur dieser Benutzer (und wahrscheinlich der Administrator / Root-Benutzer)

direkten Betriebssystemzugriff auf die Datenbankdateien haben sollte. Die Datenbankdateien sollten sich nicht in Verzeichnissen befinden, auf die über das Netzwerk oder andere lokale Benutzer als speziell autorisiertes Personal zugegriffen werden kann.

Damit die Sicherheit effektiv ist, wird es auch bevorzugt, dass der Servercomputer an einem sicheren Ort gespeichert wird, um physische Manipulationen zu verhindern, die es nicht autorisierten Benutzern ermöglichen könnten, über die Einschränkungen des Betriebssystems hinaus auf die Festplatte zuzugreifen.

Die obige Erklärung hilft Entwicklern jedoch nicht unbedingt, die Datenbanken für die Verteilung und Installation an Remote-Standorten geschrieben haben und möglicherweise das in ihren Datenbanken enthaltene geistige Eigentum schützen möchten. Solche Bedenken können speziell die Metadaten (Tabellenstrukturen und -beziehungen, gespeicherte Prozeduren und Trigger) umfassen, können aber auch spezifische Daten enthalten, die in einigen Tabellen enthalten sind. Diese Bedenken stellen den Hauptzweck dieses Artikels dar.

Chapter 3. Das Problem

Ein Entwickler erstellt eine Datenbank (und normalerweise eine zugehörige Clientanwendung) für die Installation auf Servern an Remotestandorten. An solchen Standorten hat eine Person an diesem Standort normalerweise vollen Zugriff auf den Computer, auf dem der Firebird-Server ausgeführt wird, um Sicherungen und andere Wartungsaufgaben ausführen zu können. Wie in den Hintergrundinformationen beschrieben, bietet der direkte Zugriff auf die Datenbankdatei die Möglichkeit, vollständigen und uneingeschränkten Zugriff auf alle Informationen in der Datenbank zu erhalten - sowohl auf Daten als auch auf Metadaten.

In solchen Fällen vertraut der Entwickler den Benutzern an diesen Remotestandorten möglicherweise nicht, um das in der Datenbank dargestellte geistige Eigentum vertraulich zu behandeln. Es besteht die Befürchtung, dass die Benutzer die Datenbank für ihre eigenen Zwecke zurückentwickeln oder dass diese Remotestandorte nicht die Sicherheit gewährleisten, die erforderlich ist, um den unbefugten Zugriff auf die Datenbank zu verhindern.

Dies führt zu den allgemeinen Fragen in den Firebird-Listen wie folgt:

“Ich will…”

> “…mein Datenbankdesign (Tabellenstrukturen, gespeicherte Prozeduren, Trigger usw.) bei einer Remote-Installation vor allen Benutzern der Datenbank schützen. Wie kann ich das mit Firebird machen?”

“I will…”

> “…verhindern, dass Benutzer einer Remote-Installation auf diese bestimmten Datentabellen zugreifen können. Sie enthalten proprietäre Informationen, die von der Anwendung intern verwendet werden.”

Firebird (zumindest bis Version 1.5) bietet keine integrierten Verschlüsselungsfunktionen. Die einfache Antwort auf diese beiden Fragen lautet, dass dies mit den aktuellen Funktionen von Firebird nicht möglich ist. Ein Benutzer, der direkten Zugriff auf die Datei erhalten kann, erhält Zugriff auf alle Details in dieser Datei.

In erster Linie ist keine Problemumgehung möglich, da der Server selbst die Metadaten lesen muss. In der zweiten Instanz ist es durchaus möglich, clientseitige Verschlüsselungs- / Entschlüsselungsfunktionen zu implementieren, aber dann verlieren Sie die Fähigkeit, Datenbankindizes und Suchfunktionen effektiv zu nutzen - und die Schlüsselverwaltung bleibt ein Hauptproblem (siehe unten).

Chapter 4. Die Lösung

Es gibt wirklich nur eine mögliche Lösung für diese Anforderungen: Hosten Sie die Datenbank und den Server an Ihrem eigenen Standort und lassen Sie die Clients über DFÜ- oder Interneteinrichtungen usw. eine Remoteverbindung zu Ihrem Server herstellen. Terminalserverfunktionen (Windows oder Linux / Unix) könnten eine nützliche Möglichkeit sein, solche Anforderungen zu implementieren.

Auf diese Weise behalten Sie die Kontrolle über die Datenbankdatei und können den Zugriff auf die verschiedenen Funktionen und Strukturen Ihrer Datenbank mithilfe der üblichen internen Firebird-Sicherheitsfunktionen (Rollen und Berechtigungen usw.) einschränken.

4.1. Schwierigkeiten

Es sei darauf hingewiesen, dass es auch in dieser Situation Schwierigkeiten gibt, wenn Sie die Struktur Ihrer Datenbank schützen möchten.

4.1.1. Anforderungen der Zugriffsschicht

Verschiedene Datenbankentwicklungsbibliotheken fragen Metadaten wie Primärschlüssel, Domäne und ähnliche Strukturinformationen ab, um die Entwicklung von Clientanwendungen zu vereinfachen. Infolgedessen stellen Sie möglicherweise fest, dass Sie Benutzer nicht daran hindern können, auf Metadaten zuzugreifen, ohne dass Ihre Anwendung auch die erforderlichen Informationen sammelt.

Dies kann bedeuten, dass Sie wählen müssen, ob Metadatendetails über eine ausgefeilte Datenzugriffsschnittstelle von Ihrem Server übertragen werden sollen oder ob Sie viel Zeit für die Entwicklung einer Anwendung mit einer weniger ausgefeilten Zugriffsbibliothek benötigen.

4.1.2. “Leaking” durch Inferenz und Deduktion

Es gibt auch das Problem, dass die meisten Clientanwendungen von Natur aus strukturelle Informationen über die Datenbank “leaken”, mit der sie interagieren. Es ist sehr selten, dass eine datenbankzentrierte Anwendung über eine Schnittstelle verfügt, die nicht viele Details zu den verwendeten Tabellenstrukturen enthält.

Einige Details können hinter Ansichten und auswählbaren gespeicherten Prozeduren verborgen sein, aber das Definieren solcher Funktionen nur zum Versuch, strukturelle Informationen zu verbergen, ist eine frustrierende Übung. Es ist wahrscheinlich sowieso sinnlos, da einige Details verschwinden, was auch immer Sie versuchen.

4.2. Benutzerdaten schützen

Bevor ich mit anderen Diskussionen zur Verschlüsselung von Firebird-Daten fortfahre, möchte ich hervorheben, dass Benutzer ihre Datenbanken mit Verschlüsselung schützen können. Dies hilft Entwicklern nicht, die Informationen vor legitimen Benutzern verbergen möchten, kann jedoch dazu beitragen, die Anforderungen von Kunden zu erfüllen, die die Sicherheit ihrer Datenbanken

erhöhen möchten.

In einigen Bürosituationen ist es möglicherweise nicht praktikabel, den Firebird-Servercomputer in einer wirklich sicheren Umgebung zu platzieren. In Zeiten, in denen das Büro besucht wird, ist die Wahrscheinlichkeit, dass jemand auf den Computer zugreifen kann, um die Datenbankdateien zu kopieren (oder den Computer oder die Festplatte zu stehlen, um die Dateien später abzurufen), sehr gering. Außerhalb der normalen Arbeitszeit (Nächte und Wochenenden) kann dies jedoch eine andere Sache sein. Jemand könnte Zugriff auf das Büro erhalten, die Festplatte aus Ihrem Computer herausnehmen (oder den gesamten Computer herausnehmen) und sie wegnehmen, um auf die Datenbank zuzugreifen.

4.2.1. Verschlüsselung

Während Firebird selbst keine integrierten Verschlüsselungsfunktionen bietet, gibt es einige hervorragende Produkte, die dies tun. Sie können Software installieren, die ein verschlüsseltes Volume auf Ihrem Computer erstellt, und die Datenbankdatei (und alle anderen vertraulichen Daten) auf diesem Volume suchen. Wenn der Computer heruntergefahren wird, sind alle Daten in einer verschlüsselten Datei vorhanden und können ohne den Schlüssel nicht aufgerufen werden. Wenn Sie den Computer starten, müssen Sie das verschlüsselte Volume bereitstellen und den geheimen Schlüssel angeben, bevor auf die Daten zugegriffen werden kann. Dieser zusätzliche und notwendigerweise manuelle Schritt im Startvorgang mag unpraktisch sein, bietet jedoch eine hervorragende Sicherheit für unbeaufsichtigte Computersysteme.

Software mit diesen Funktionen: TrueCrypt (www.truecrypt.org), Bestcrypt von Jetico (www.jetico.com) und PGPDisk (www.pgpi.org/products/pgpdisk/) — Beachten Sie, dass dieser Link zu einer alten Freeware-Version führt. Diese Website enthält Links zu neueren kommerziellen Versionen des Produkts. Es gibt andere, aber die letzten beiden habe ich selbst benutzt.

Warum bietet Firebird keine Verschlüsselung an?

Aufgrund der oben beschriebenen Anforderungen fordern Benutzer häufig an, dass Firebird in einer zukünftigen Version die Möglichkeit zum Verschlüsseln von Metadaten, ausgewählten Benutzerdaten oder sogar der gesamten Datenbank hinzufügen soll. Da ich kein Firebird-Kernentwickler bin, kann ich nicht kategorisch sagen, dass dies nicht passieren wird. Es geht jedoch nicht wirklich darum, ob eine Verschlüsselung praktikabel oder nützlich ist, sondern darum, ob die Schlüsselverwaltung eine Lösung für die von uns untersuchten Probleme bietet.

Die Verschlüsselung kann nur so gut sein wie der für die Entschlüsselung erforderliche geheime Schlüssel. Es kann schlimmer sein, aber es kann nicht besser sein. Es gibt mehrere ausgezeichnete Verschlüsselungsalgorithmen, die verwendet werden könnten. Wenn eine gute Verschlüsselung verwendet wird, richten sich Angriffe wahrscheinlich eher gegen den Schlüssel als gegen die Verschlüsselung selbst.

Wie könnte Verschlüsselung funktionieren?

Schauen wir uns also an, wie die Dinge funktionieren würden, wenn Firebird die Metadaten in einer Datenbank verschlüsseln würde …

Bevor auf die Datenbank zugegriffen werden kann, muss der geheime Schlüssel angegeben

werden. Es wäre sinnlos, dem Benutzer den Entschlüsselungsschlüssel zu geben, um einfach zum ursprünglichen Problem zurückzukehren. Wenn der Kunde den Server neu startet, ruft er vermutlich den Entwickler an, der sich dann einwählt und den erforderlichen Schlüssel eingibt. Selbst wenn dies praktikabel wäre, wird es das Problem nicht unbedingt lösen. Beispielsweise könnte der Kunde eine Überwachungssoftware auf seinem Server installieren, um den eingegebenen Schlüssel zu erkennen.

Es gibt hardwarebasierte Lösungen, die einen Schlüssel für einen Entschlüsselungsprozess bereitstellen. Aber auch dies müsste im Besitz des Clients sein, und wenn wir dem Client nicht vertrauen, können wir ihn nicht davon abhalten, ihn zu verwenden, um von einem anderen Server, auf dem das SYSDBA-Kennwort bekannt ist, auf die Datenbank zuzugreifen.

Firebird ist ein Open Source Produkt. Wenn die Verschlüsselungsfunktionen integriert wären oder Open Source-Plug-In-Bibliotheken verwendet würden, könnten Benutzer ihre eigenen Versionen des Servers oder Plug-Ins erstellen, die nicht nur die erforderliche Ver- und Entschlüsselung für den Zugriff auf die geschützte Datenbank durchführen Sie können aber auch den Schlüssel ausgeben oder einfach die entschlüsselten Details direkt ausgeben. Der Entwickler, der nicht die Kontrolle über den Server hat, kann solche Aktivitäten weder erkennen noch verhindern.

Sie können eine eigene Version des Firebird-Servers mit dem in der ausführbaren Datei versteckten Entschlüsselungsschlüssel erstellen. Dekompilierer sind jedoch verfügbar. Es würde nicht lange dauern, den Schlüssel zu finden, indem Sie einfach die dekompilierten Versionen Ihres benutzerdefinierten Firebird-Builds mit der normalen, unverschlüsselten Version vergleichen.

Es gibt verschiedene Datenbankprodukte, die eine starke Verschlüsselung vorgeben sollen. Möglicherweise ist die Verschlüsselung stark, aber wenn die Schlüsselverwaltung nicht vorhanden ist, um diese Funktion zu unterstützen, wird die Verschlüsselung nicht den gewünschten Effekt erzielen. Es kann Sie ermutigen zu glauben, dass Sie geschützt sind, aber Sie müssen das Schlüsselmanagement studieren, um herauszufinden, ob dies wirklich wahr ist.

Die *schmerzhafte Wahrheit* ist, dass, sobald Sie die Kontrolle über die Hardware verlieren, auf der die Ver- und Entschlüsselung stattfindet, alle Wetten ungültig sind. Wenn der Entschlüsselungsschlüssel nicht zuverlässig sicher gehalten werden kann, wird selbst eine gute Verschlüsselung kaum mehr als Sicherheit durch Dunkelheit.

4.2.2. Einschränkung der Datenverteilung

Einige Personen fordern eine Verschlüsselung der Datenbankdaten an, damit sie versuchen können, die Verbreitung von Daten einzuschränken. Sie freuen sich, dass der bestimmte autorisierte Benutzer die Daten sieht, möchten jedoch die Fähigkeit dieses Benutzers einschränken, die Daten an andere Personen zu verteilen.

Stellen Sie sich für einen Moment vor, dass alle oben beschriebenen Schlüsselverwaltungsprobleme gelöst wurden, so dass es für den Benutzer unpraktisch geworden ist, nur die Datenbank zu kopieren. In solchen Fällen schrieb der Benutzer einfach ein kleines Programm, das die Daten, an denen er interessiert war, extrahierte (vom rechtmäßig installierten Server) und diese Daten in seine eigene Datei oder Datenbank kopierte.

Ich denke, es ist möglich, dass Firebird in Zukunft eine Art Anwendungsauthentifizierungssystem

bereitstellt, das es ermöglicht, diese Form der Datenextraktion einzuschränken, jedoch bestehen die meisten der gleichen Probleme. Wenn Sie den Server nicht steuern, können Sie den Benutzer nicht daran hindern, eine Version des Servers zu installieren, für die keine Authentifizierung erforderlich ist.

4.2.3. SYSDBA-Zugriff entfernen

Zu verschiedenen Zeiten wurde vorgeschlagen, dass das Entfernen des SYSDBA-Zugriffs auf eine Datenbank die Lösung sein könnte. Die Idee dahinter ist, dass das Verschieben der Datenbank auf einen neuen Server, auf dem das SYSDBA-Kennwort bekannt ist, der Person nicht hilft, da SYSDBA ohnehin keinen Zugriff hat. Einige haben diesbezüglich nur begrenzte Erfolge gemeldet, indem sie einen SQL-Rollennamen von SYSDBA erstellt und sichergestellt haben, dass dieser keinen Zugriff auf die Datenbankobjekte hat.

Es löst das Problem jedoch nicht wirklich. Die Datenbankdatei kann mit einem Hex-Viewer oder einem ähnlichen Dienstprogramm angezeigt und die Liste der verfügbaren Benutzernamen ermittelt werden. (Das Erkennen der Eigentümer der Datenbankobjekte wäre besonders nützlich.) Sobald diese Namen bekannt sind, können sie dem neuen Server hinzugefügt und direkt verwendet werden.

Eine noch einfachere Problemumgehung könnte darin bestehen, die eingebettete Version des Firebird-Servers (siehe unten) zu verwenden oder eine eigene Version des Firebird-Servers zu kompilieren, bei der Sicherheitsbeschränkungen ignoriert werden.

4.2.4. Benutzerdefinierte Namen für SYSDBA

Es gab einige Vorschläge, wie der SYSDBA-Benutzername geändert werden kann. Dies bietet möglicherweise einen eingeschränkten Schutz vor Brute-Force-Netzwerkangriffen gegen das SYSDBA-Kennwort, da bei solchen Angriffen sowohl der Benutzername als auch das Kennwort erraten werden müssen, das System jedoch nicht vor einer Person mit direktem Zugriff auf die Datenbankdatei geschützt wird .

4.2.5. Quellcode gespeicherter Prozeduren und Trigger löschen

Wenn Sie eine gespeicherte Prozedur oder einen Trigger für eine Firebird-Datenbank schreiben und definieren, speichert der Server eine fast vollständige Kopie des Prozedurquellcodes zusammen mit einer “kompilierten” Kopie, die als BLR (Binary Language Representation) bezeichnet wird. Es ist das BLR, das vom Server ausgeführt wird, der Quellcode wird nicht verwendet.

Einige Entwickler versuchen, zumindest einen Teil ihrer Datenbankmetadaten zu schützen, indem sie den Quellcode vor dem Verteilen der Datenbank aus der Datenbank löschen (eine einfache direkte Aktualisierung anhand der relevanten Metadatentabellenfelder). Ich empfehle Ihnen, dies aus zwei Gründen nicht zu tun …

1. BLR ist eine ziemlich vereinfachte Codierung des Quellcodes. Es wäre nicht schwierig, die BLR wieder in eine für Menschen lesbare Form zu dekodieren. Eine solche Dekodierung enthielte zwar keine Kommentare und Formatierung, aber die SQL, die in einer gespeicherte Prozedur oder einem Trigger stünde, ist selten so kompliziert, dass dies ein großes Problem verursachen

würde. Daher ist der Schutz, den das Entfernen von Quellcode bietet, nicht sehr bedeutend.

2. Der Quellcode kann für andere Zwecke nützlich sein. Damit können Fixes direkt auf die Datenbank angewendet werden, ohne dass die vollständige Quelle von einer anderen Stelle abgerufen werden muss (und dann daran gedacht wird, sie erneut zu entfernen, wenn der Fix angewendet wird). Der Quellcode wird auch von verschiedenen Dienstprogrammen verwendet, beispielsweise von meiner eigenen DBak-Anwendung - einem alternativen Sicherungsprogramm zu “gbak”. Ich habe mir zu diesem Zeitpunkt noch nicht die Mühe gemacht, einen eigenen BLR-Decoder zu schreiben. Daher ist DBak auf die Verfügbarkeit des Quellcodes angewiesen, um ein DDL-Skript zum Rekonstruieren einer Datenbank erstellen zu können.

Chapter 5. Embedded Firebird Server

Es gibt eine spezielle Version des Firebird-Servers, die als “eingebettet” (engl. embedded) bezeichnet wird. Dies ist eine spezielle Clientbibliothek, die den Server selbst enthält. Wenn eine Anwendung eine Verknüpfung zu dieser Bibliothek herstellt, wird der Server geladen und der direkte Zugriff auf alle Datenbanken ermöglicht, auf die auf dem lokalen Computer zugegriffen werden kann. Diese Version des Servers verwendet keine Sicherheitsdatenbank. Der während der “Anmeldung” angegebene Benutzername (es erfolgt keine Kennwortauthentifizierung) wird zum Verwalten des Benutzerzugriffs auf Datenbankobjekte (über SQL-Berechtigungen) verwendet. Wenn dieser Benutzername jedoch SYSDBA (oder der Eigentümer der Datenbank) lautet, ist der uneingeschränkte Zugriff möglich möglich.

Die Verwendung des Embedded-Modus ist nützlich für Entwickler, die einfach zu verteilende Einzelbenutzeranwendungen erstellen möchten, die keine Sicherheit benötigen.

Aus dieser kurzen Beschreibung geht hervor, dass die Installation eines eingebetteten Server-Clients auf einem Server, auf dem andere Datenbanken gehostet werden, ein großes Sicherheitsrisiko darstellen kann. In Wirklichkeit ist das Risiko nicht größer als wenn der eingebettete Client nicht existiert hätte.

Wenn eine Anwendung den eingebetteten Server lädt, arbeitet der Server im Sicherheitskontext der Anwendung (und damit des Benutzers). Dies bedeutet, dass der eingebettete Server nur auf Datenbankdateien zugreifen kann, auf die der Benutzer direkt über das Betriebssystem zugreifen kann. Es ist in jedem Fall eine schlechte Nachricht, einem nicht vertrauenswürdigen Benutzer Zugriff auf die Installation von Programmen auf einem sicheren Server zu gewähren. Vorausgesetzt, Sie haben die entsprechenden Dateiberechtigungen für sichere Datenbanken angegeben, ist der eingebettete Server selbst keine Bedrohung.

Die Bedrohung kommt von all den anderen Dingen, die der Benutzer installieren könnte.

Die Tatsache, dass der eingebettete Server vorhanden ist, dient nur dazu, hervorzuheben, was beim direkten Zugriff auf eine Datenbankdatei möglich ist, insbesondere in einer Open Source-Umgebung. Wenn es nicht schon existiert hätte, wäre es sicherlich möglich, dass jemand eine gleichwertige Fähigkeit kompiliert.

Chapter 6. Andere Formen der Verschleierung

Verschiedene andere Formen der Sicherheit durch Verschleierung wurden vorgeschlagen. Solche Dinge wie spezielle Ereignisse, die beim An- und Abmelden ausgelöst werden, um Benutzerfunktionen aufzurufen, um den Zugriff zu verhindern oder zu verweigern. Solche Funktionen bieten möglicherweise eine begrenzte Verwendung für Closed-Source-Systeme, bei denen die Unklarheit der Implementierung dazu beiträgt, genau zu verbergen, wie Informationen geschützt werden. Bei einem Open-Source-System besteht die Problemumgehung für solche Hacks darin, einfach eine eigene Version des Servers zu erstellen, die das Ereignis oder den Code umgeht, der den Zugriff verhindert. In einem Open-Source-System ist es schwierig, Dunkelheit zu bieten.

Überlegen Sie auch, was passiert, wenn Sie Ihre kompilierten ausführbaren Dateien verteilen. Kompilierte Programme sind gute Beispiele für Dunkelheit. Es wird (normalerweise) keine Verschlüsselung verwendet, alle Schritte des Codes können von jedem mit Zeit und Wissen analysiert werden, und tatsächlich stehen Dekompilierer zur Verfügung, die diesen Prozess unterstützen. Sobald eine Person herausfindet, mit welchen Bibliotheken Ihr Code kompiliert wurde, beschleunigt das Isolieren der Ergebnisse auf nur Ihren eigenen “geheimen” Code den gesamten Prozess erheblich. Haben Sie an Borland, Microsoft oder an jemanden geschrieben, der verlangt, dass er seine kompilierten Binärdateien irgendwie verschlüsselt?

Chapter 7. Akzeptabele niedrige Sicherheit

Meine bisherigen Kommentare waren auf die Idee einer starken Sicherheit gerichtet, und ich denke, das Konzept der Sicherheit durch Verschleierung wurde mit einiger Verachtung geschrieben. Manchmal ist jedoch nur schwache Sicherheit alles, was Sie wollen. Manchmal sind die Daten einfach nicht so wertvoll. Sie möchten den gelegentlichen Browser stoppen und ihn für den fortgeschritteneren Dieb zumindest unpraktisch machen.

Ich habe solche Schemata selbst an verschiedenen Orten angewendet. Oft macht es keinen Sinn, Twofish, AES oder was auch immer auf solche Schemata zu werfen, weil es um starke Verschlüsselung geht. Sie sind mit einem hohen Verarbeitungsaufwand und Komplikationen im Zusammenhang mit der Aufrechterhaltung der Sicherheit verbunden. Ein einfaches XOR gegen eine bekannte Zeichenfolge (den Schlüssel) kann ausreichend sein. Wenn der Schlüssel vom Dieb entdeckt werden kann, spielt es keine Rolle, ob Sie eine schwache oder eine starke Verschlüsselung verwendet haben. Das Spiel ist trotzdem vorbei.

Die meisten einfachen XOR-basierten Algorithmen können mit geringem Aufwand gebrochen werden. Weitere Informationen und andere Optionen finden Sie in einer guten Verschlüsselungsreferenz.

Chapter 8. Wählen der Verschleierung

Die Sache mit Sicherheit durch Verschleierung ist, dass sie unklar sein muss! Wenn Firebird eine Art Verschlüsselung in seine Lese- und Schreibvorgänge auf der Festplatte implementieren würde, wäre dies nicht unklar, da es sich um ein Open-Source-Projekt handelt. Es würde fast keine Zeit dauern, die Quelle neu zu kompilieren, um den bereitgestellten Schlüssel zu ermitteln, und alles geht verloren.

Wenn Sie diese Funktion wirklich benötigen, erhalten Sie die Firebird-Quelle, fügen Ihren eigenen undurchsichtigen Code in die Lese- und Schreibmethoden der Festplatte ein und kompilieren Ihre eigene Variante des Firebird-Servers. (Ein solcher Code könnte durch Dekompilieren der ausführbaren Datei entdeckt werden, aber es braucht einen ziemlich ernsthaften Dieb, um dies zu versuchen.)

Versuchen Sie vorher herauszufinden, ob dies Ihr Problem tatsächlich lösen würde, wenn der Benutzer zusammen mit der Datenbank auch eine Kopie der speziell kompilierten ausführbaren Dateien erstellt. oder wenn es einem Benutzer weiterhin möglich ist, die Geheimnisse direkt von Ihrem laufenden Server zu extrahieren.

Chapter 9. Das philosophische Argument

Es gibt auch die philosophische Frage, warum Sie ein Open-Source-Datenbankserverprodukt zum Erstellen einer Closed-Source-Datenbank auswählen würden. Viele Menschen haben zu dem Projekt beigetragen, weil sie der festen Überzeugung sind, dass Open Source der beste Weg ist, Software bereitzustellen.

Vor allem aber, wenn es um die Speicherung von Benutzerdaten geht, bin ich fest davon überzeugt, dass die Benutzer auf der Möglichkeit bestehen sollten, auf ihre eigenen Daten zuzugreifen - was häufig die Notwendigkeit einschließt, die Strukturen und Prozesse zu verstehen, über die Sie verfügen gebaut (die Metadaten). Wenn Sie Ihr Geschäft aufgeben oder anderweitig nicht mehr verfügbar sind, kann es von entscheidender Bedeutung sein, dass die Benutzer zumindest ihre eigenen Daten (in geeigneten Formaten) extrahieren können, um auf alternative Systeme umsteigen zu können.

Können Sie den Benutzern vertrauen, dass sie Ihr geistiges Eigentum respektieren, während Sie noch im Geschäft sind und verfügbar sind? Stellen Sie die notwendigen Dienstleistungen und Einrichtungen bereit und hoffentlich auch. Wenn nicht, besteht eine gute Chance, dass Sie wenig tun können, um sie zu stoppen.

Chapter 10. Schlussfolgerungen

Das Problem war, dass zu viele Menschen die Sicherheit nicht verstehen und wie schwierig es ist, sie gut zu machen. Bedauerlicherweise gab es viele Softwareprodukte, die solche Missverständnisse begünstigten, indem sie eher Dunkelheit als echte Sicherheit implementierten. Erleben Sie die Anzahl der Unternehmen in der Umgebung, die “Datenwiederherstellungs”-Dienste anbieten. Damit ist gemeint, dass die *vermeintliche Sicherheit verdeckter Daten umgangen oder verletzt* wird.

Verschlüsselung ist kein Allheilmittel für die Sicherheit. Wenn Sie nicht die Kontrolle über die Umgebung haben (die Hardware, das Betriebssystem und die gesamte auf diesem System ausgeführte Software), haben Sie keine Kontrolle über die Sicherheit - unabhängig davon, welche Verschlüsselungsschemata möglicherweise vorhanden sind. Dies ist der Fall, wenn Sie Ihre Datenbank an Remoteserverinstallationen verteilen.

Wenn Sie die Daten oder Metadaten in Ihrer Datenbank wirklich schützen müssen, müssen Sie die Kontrolle über die Datenbankdatei und die Umgebung behalten, in der auf sie zugegriffen wird. Keine andere Lösung bietet Ihnen das gleiche Sicherheitsniveau.

Chapter 11. Danksagung

Ich möchte mich bei den verschiedenen Personen bedanken, die diesen Artikel überprüft und kommentiert haben. Ich möchte mich auch bei den vielen Personen bedanken, die zur Firebird-Supportliste beitragen, aus der viele Informationen in diesem Artikel stammen.

Appendix A: Dokumenthistorie

Die exakte Dokumenthistorie — beginnend bei Version 0.5 — ist im Git-Repository des `firebird-documentation`-Repository zu finden; siehe https://github.com/FirebirdSQL/firebird-documentation

Revision History

N/A	14. Feb 2005	G W	Erste Edition
N/A	11. Apr 2005	G W	Der Abschnitt “Akzeptable niedrige Sicherheit” wurde überprüft, um zu versuchen, einfache XOR-Algorithmen als schwach hervorzuheben, um sicherzustellen, dass die Leser weitere Untersuchungen durchführen, wenn sie an diesem Ansatz interessiert sind.
N/A	26. Apr 2005	G W	Zusätzlicher Abschnitt zum eingebetteten Server (und Verweise darauf). Fußnote in Kursivschrift verschoben, funktionieren Fußnoten nicht gut mit HTML. Ein Inhaltsverzeichnis wurde hinzugefügt.
N/A	4. Dez 2005	G W	Verweis auf TrueCrypt hinzugefügt. *Nutzung dieses Dokumentes*-Abschnitt hinzugefügt. Ein Danksagungsabschnitt wurde hinzugefügt.
0.5	7. Dez 2005	PV	*Documenthistorie* und *Nutzung dieses Dokumentes* wurden in den Anhang verschoben. Versionsnummer zur Verwendung im Firebird-Projekt hinzugefügt. Dokument zum Firebird CVS-Repository hinzugefügt.
0.5-de	2. Jan 2009	DA	Deutsche Übersetzung
0.6	30. Jun 2020	MR	Konvertierung in AsciiDoc, geringfügige Bearbeitung von Texten
0.6-de	8. Jul 2020	MK	Deutsche Anpasungen; Übernahme in AsciiDoc

Appendix B: Lizenzhinweis

Ich habe versucht, dieses Dokument zum Zeitpunkt des Schreibens korrekt zu machen, kann jedoch nicht garantieren, dass keine Fehler vorliegen. Sicherheit ist ein komplexes Thema, bei dem Sicherheit für Ihr Produkt oder Ihre Installation wichtig ist. Sie sollten professionellen Rat einholen.

Ich mache für die Verwendung dieses Dokuments keine besonderen Einschränkungen. Es steht Ihnen frei, dieses Dokument zu reproduzieren, zu ändern oder zu übersetzen. Geänderte Versionen des Dokuments sollten jedoch mit den vorgenommenen Änderungen und dem Namen des Autors der Änderung versehen werden (damit mein Name nicht mit Texten in Verbindung steht, die ich nicht geschrieben habe). --G.W.

Mitwirkende: Daniel Albuschat – siehe Dokumenthistorie.

Mitwirkende: Martin Köditz – siehe Dokumenthistorie.

 Kontakt zum Mitwirkenden: martin dot koeditz at it dash syn dot de.