PostgreSQL の監査ログ

データベースのセキュリティ脅威を検知する

技術を知る

近年、顧客情報の流出などセキュリティ事故が増加する一方で、企業においては、情報漏えいやサイバー攻撃への対策を最優先に考える必要があります。特に、個人情報や機密情報を管理するデータベースにおいては、セキュリティ対策が急務となっています。セキュリティの国際評価基準（ISO15408）では、データベースのセキュリティ脅威として「不正な接続（なりすまし）」、「不正なアクセス」、「情報漏えい」が想定されています。これらのセキュリティ脅威に対しては、「予防する」、「阻止する」、「検知する」といった施策が必要となりますが、監査ログは、セキュリティ脅威を「検知する」ための機能に位置づけられます。

ここでは、PostgreSQL データベースおよび Fujitsu Enterprise Postgres（以降、Enterprise Postgres と略します）における監査ログについて解説します。

1. 監査ログとは

監査ログは「いつ」、「誰が」、「どこから」、「何に対して」、「どんな処理を」、「実行結果は」という、データベースに対するアクセスの記録です。これらの記録を追跡し、分析することで、「データが不正に変更された」や「不正な接続元からアクセスされた」などの情報を得ることができます。セキュリティ事故の中で、最も社会的影響が大きいといわれる情報漏えいは、「なりすまし」や「不正アクセス」などを要因として起こりますが、もし、情報漏えいが発生しても、監査ログを追跡することで原因や影響範囲を特定でき、被害を最小限に食い止めることができます。また、監査ログを定期的に監視することで、セキュリティ被害の未然防止にもつながります。このように、監査ログは、データベースのセキュリティ対策において、重要な役割を担う機能といえます。
では、PostgreSQL と Enterprise Postgres では、監査ログをどのように取得するのでしょうか？

1.1 監査ログの取得方法と課題

PostgreSQL において監査ログを取得するには、以下の 2 つの方法があります。

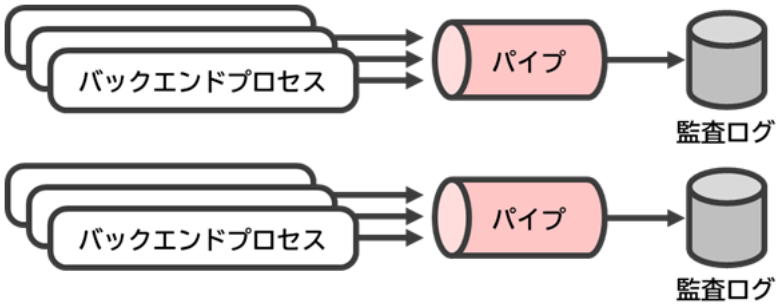
- **log_statement パラメーターによる取得**
  postgresql.conf にログの出力を制御する log_statement パラメーターを設定することで、監査ログに相当するログが取得できます。このパラメーターを使用することで、実行した SQL をサーバーログとして取得できます。しかし、取得したログをデータベース監査の目的で使用する場合、収集内容が不足しており、監査ログとしての役割を十分に果たしているとは言えません。また、サーバーログとして出力されるため、メッセージや運用ログが混在しており、監査を行い難いといった課題もあります。さらに、特に、長い SQL 文を複数同時に実行すると、サーバーログへの出力が膨大な量となり、ファイルへの書き込み処理が集中するため、データベースの性能劣化を引き起こす可能性もあります。
- **拡張モジュール“pgaudit” による取得**
  監査ログを出力するための拡張モジュールである“pgaudit”で取得します。「どの操作をどのレベルで出力するのか」といった詳細なログ取得の設定や、log_statement パラメーターによる取得では不足している情報（オブジェクト特定のためのスキーマ名など）が取得できます。pgaudit にはさまざまな派生版がありますが、いずれも出力先はサーバーログであり、log_statement パラメーターによる取得と同様の課題があります。

1.2 課題に対する Enterprise Postgres の取り組み

Enterprise Postgres では、pgaudit に対してコネクションに関する情報やエラーメッセージといった SQL 文の実行結果情報を取得できるように機能強化された「pgaudit refactored 版」をベースに、以下の機能を拡張した監査ログ機能を提供しており、log_statement パラメーターや pgaudit における課題を解決しています。

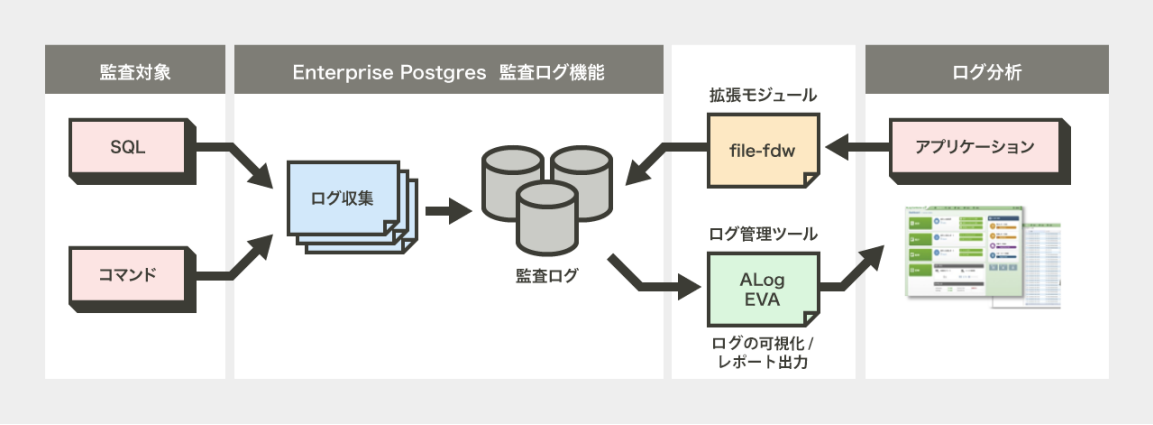
- **監査ログの出力先を専用ログファイルにするかサーバーログにするか選べる**
  pgaudit refactored 版に対して、監査ログを専用ログファイルに出力する機能を追加しています。専用ログファイルに出力することで、監査ログだけを扱うことができるようになり、ログの解析が行いやすくなります。さらに、サーバーログへのログ出力量が抑えられるため、データベースの性能劣化も抑止できます。

- **監査ログの出力先を複数設定できる**
  pgaudit refactored 版に対して、監査ログを複数の専用ログファイルに分割して出力する機能を追加しています。これにより、長い SQL 文を持つ多重度が高い処理において、性能劣化を抑止でき、システムの安定稼働ができるようになりました。
  監査ログファイルが 1 つの場合、SQL を処理するバックエンドプロセスからのログ書込み処理が集中すると、出力機構であるパイプへの負荷が高まる可能性があります。本機能によりログファイルを分割することで複数の出力機構を設け、処理を分散させることで、性能の向上が期待できます。



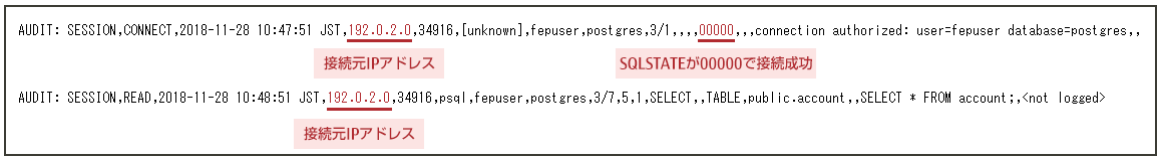
本機能の性能を評価するため、トランザクションの構成が更新 66%、参照 33%で、各 SQL 文の長さが 800 バイト、DB への接続数が 1600 のモデルを使用して検証を行いました。このモデルでは、複数の出力機構を設定した場合、出力機構が 1 つの場合と比べてスループット性能が約 3 倍に改善されました。

- **SQL での参照可**
  PostgreSQL の拡張モジュールである file-fdw を利用して、監査ログをアプリケーションから SQL で参照できます。また、ログ管理ツール（ALog EVA など）と連携して、GUI ベースでの監査を行うこともできます。



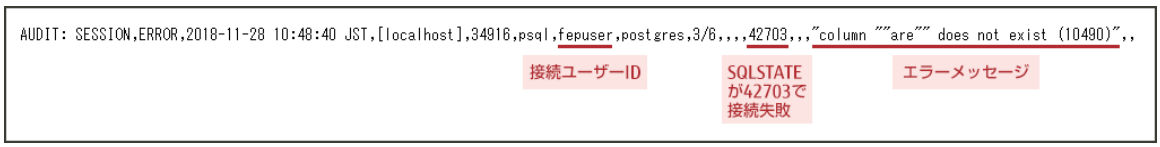
- **収集できるログの種類を追加**
  Enterprise Postgres の監査ログは pgaudit refactored 版をベースとしているため、PostgreSQL や pgaudit では収集できない「接続に関するログ」や「SQL の実行結果に関するログ」が収集できます。たとえば、データを参照する SELECTが実行された場合、pgaudit の監査ログでは、「SELECTが発行された」までしか分析できませんが、Enterprise Postgres の監査ログでは、「xxxx という不正な接続元から SELECTが発行された。SELECTが成功しているので情報が流出した可能性がある」というところまで分析できます。

ログ出力例（接続に関するログ）



このログから、「不正な接続元（IP アドレス:192.0.2.0）から接続され、SELECT が成功した」ことが判ります。

ログ出力例（SQL の実行結果に関するログ）



このログから、「ユーザーID“**fepuser** ”による SQL の実行が失敗した」ことが判ります。

1.3 Enterprise Postgres　における監査ログの詳細

前述したとおり、監査ログとはデータベースに対する「いつ」、「誰が」、「どこから」、「何に対して」、「どんな処理を」、「実行結果は」の記録です。データベースを監査する際に情報が不足していると、十分な監査ができません。Enterprise Postgres で取得した監査ログの情報と監査の対象項目をマッピングして、情報の十分性を確認してみましょう。

SQL 実行が成功した場合のログ出力例

SQL の実行が成功すると、成功した SQL 種別や SQL 文を含む、以下のような監査ログが出力されます。監査ログが出力されたことで SQL の実行が成功したと判断します。

```
AUDIT: SESSION,READ,2018-11-28 10:48:19 JST,192.0.2.0,34916,psql,fepuser,postgres,3/7,5,1,SELECT,,TABLE,public.account,,SELECT
* FROM account WHERE age >= 40;,<not logged>
```

監査の対象項目		Enterprise Postgres　から出力される情報
いつ	SQL の実行開始時間	2018-11-28 10:48:19 JST
誰が	接続元のユーザー名	fepuser
どこから	アプリケーション名	psql
	プロセス ID	34916
	接続元のホスト名または IP アドレス	192.0.2.0
何に対して	データベース名	postgres
	オブジェクト種別	TABLE

監査の対象項目		Enterprise Postgres　から出力される情報
	オブジェクト名（スキーマ修飾あり）	public.account
どんな処理を	SQL 種別	SELECT
	SQL	SELECT * FROM account WHERE age >= 40
実行結果は	SQLSTATE	出力なし（***SQL* の実行が成功した場合は出力されない***）
	エラーメッセージ	出力なし（***SQL* の実行が成功した場合は出力されない***）

このように、Enterprise Postgres　では、データベースを監査する上で必要な情報が漏れなく収集できていることがわかります。

2. Enterprise Postgres　で監査ログを取得するためには

2.1 監査ログの出力モードを選択する

Enterprise Postgres　の監査ログ機能には、「Session Audit Logging 」と「Object Audit Logging　」の 2 種類の出力モードがあります。監査の目的に応じて出力モードを選択します。

- **Session Audit Logging**
  監査ログを出力するためのルールを指定し、そのルールに合致する監査ログだけを出力します。たとえば、「参照系 SQL（SELECT、COPY FROM）と更新系 SQL（INSERT、UPDATE、TRUNCATE、COPY TO）が実行された場合に監査ログを出力する」や「参照系 SQL（SELECT、COPY FROM）のみが実行された場合に監査ログを出力する」といった指定ができます。
- **Object Audit Logging**
  監査ログ出力の対象となるオブジェクトに対して権限を付与されているロールを指定し、そのロールで実行された操作の監査ログを出力します。たとえば、テーブル A とテーブル B があります。監査ログの出力条件としてロール'auditor'を指定し、ロール'auditor'にテーブル A の SELECT 権限が付与されているとします。この条件のもとで、テーブル A とテーブル B に対して SELECT を実行した場合、テーブル A に対する監査ログのみが出力されます。

2.2 事前準備

Enterprise Postgres の監査ログ機能を使用するために事前準備をします。

監査ログ機能の動作条件を設定する

1. **pgaudit　設定ファイルを準備する**
   監査ログの取得条件や出力先など、監査ログ機能を使用するための動作条件を pgaudit 設定ファイル（ユーザー作成の任意のファイル）に指定します。パラメーターを指定する際には、以下のとおり 3 つのセクションに分けて記述します。ただし、この時点では rule セクションと option セクションを記述することはできません。セットアップ時に必要に応じて指定します。

- output セクション：監査ログの出力先に関する情報を指定します。
- rule セクション：出力する監査ログを絞り込むためのルールを指定します。Session Audit Logging のみで使用します。
- option セクション：Object Audit Logging で使用するロールや監査ログの出力に関するオプションを指定します。

下記が pgaudit 設定ファイルの記述例です。ここでは、監査ログの出力先を指定する logger パラメーターに'auditlog'を指定することで専用ログファイルを使用するようになっています。サーバーログに出力する場合は'serverlog'を指定します。

```
[output]
logger = 'auditlog'
```

その他、pgaudit 設定ファイルで指定できる条件については、Enterprise Postgres の製品マニュアル“FUJITSU Software Enterprise Postgres セキュリティ運用ガイド”の“監査ログ機能”を参照してください。

セットアップ

セットアップの流れを説明します。

1. **postgresql.conf を編集する**
   postgresql.conf を以下のように編集して pgaudit 設定ファイル名を指定（赤字の部分）します。（以下では、監査ログ機能で必要なパラメーターのみを記載しています。）postgresql.conf の編集が完了したら、インスタンスを起動します。

```
shared_preload_libraries = 'pgaudit'
pgaudit_config_file = 'pgaudit.conf' pgaudit 設定ファイル名を指定
log_replication_commands = 'on'
log_min_message = 'error'
  .
  .
```

2. **拡張機能をインストールする**
   CREATE EXTENSION を使用して、現在のデータベースに拡張機能である pgaudit を読み込みます。

```
CREATE EXTENSION pgaudit;
```

3. **pgaudit 設定ファイルを編集する**
   必要に応じて、pgaudit 設定ファイルに条件を追加します。pgaudit 設定ファイルの編集後は、編集内容を有効にするため、インスタンスを再起動します。

2.3 Session Audit Logging で取得してみる

Session Audit Logging で取得する例として、account テーブルを作成し、account テーブルに対する監査ログを取得してみます。Object Audit Logging での取得例については、“FUJITSU Software Enterprise Postgres セキュリティ運用ガイド”の“監査ログ機能”を参照してください。

1. **監査ログの取得条件を指定する**
   pgaudit 設定ファイルに、以下のように指定します。ここでは、rule セクションの class パラメーターに'READ, WRITE'

を指定することで INSERT や SELECT といった SQL の実行結果を取得します。さらに'ERROR'を指定することでエラー終了したイベントを取得します。

```
[output]
logger = 'auditlog'  監査ログの出力先として専用ログを指定
[rule]
class = 'READ, WRITE, ERROR'  ログ出力の対象となる操作をクラスで指定
```

class パラメーターで指定できる値と意味は、以下のとおりです。

READ	SELECT、COPY FROM
WRITE	INSERT、UPDATE、DELETE、TRUNCATE、COPY TO
FUNCTION	関数呼び出し、DO
ROLE	GRANT、REVOKE、CREATE ROLE、ALTER ROLE、DROP ROLE
DDL	ROLE クラスの DDL 以外のすべての DDL（CREATE..、ALTER..など）
CONNECT	接続に関するイベント（要求、認証、切断）
SYSTEM	インスタンスの起動、プライマリーサーバーへの昇格
BACKUP	pg_basebackup
ERROR	エラーで終了したイベント（PostgreSQL のエラーコードが 00 以外）
MISC	その他のコマンド（DISCARD、FETCH、CHECKPOINT、VACUUM など）

2. **データベースに対する操作を行う**
   以下の SQL をクライアントから実行してみます。

```
CREATE TABLE account
(
id int,
name text,
birthday date,
age int
);
```



```
INSERT INTO account (id, name, password, description) VALUES (1, 'USER1', '1974-08-20', '44');
INSERT INTO account (id, name, password, description) VALUES (1, 'USER2', '1978-10-03', '40');
INSERT INTO account (id, name, password, description) VALUES (1, 'USER3', '1994-01-19', '24');
SELECT * FROM account WHERE age >= 40;
DROP TABLE account;
```

3. **取得された監査ログを確認する**

SQL を実行すると以下のような監査ログが取得されます。
SQL では、account テーブルがスキーマ修飾されていませんが、下記の監査ログには「public.account」のように、スキーマ修飾されたテーブル名が出力されています。これにより、同じ名前のテーブルが複数存在する場合に、どのスキーマ配下のテーブルに対する操作なのか？が監査ログから判断できます。さらに、class パラメーターに'ERROR'を指定しましたが、ERROR に関する監査ログは出力されていません。このことから、SQL の実行が成功したことが判ります。

```
AUDIT: SESSION,WRITE,2018-11-28 10:48:19
JST,[local],34916,psql,fepuser,postgres,3/3,2,1,INSERT,,TABLE,public.account,,"INSERT INTO account VALUES ( 1,
'USER1', '1974-08-20', 44 );",<not logged>

AUDIT: SESSION,WRITE,2018-11-28 10:48:27
JST,[local],34916,psql,fepuser,postgres,3/4,3,1,INSERT,,TABLE,public.account,,"INSERT INTO account VALUES ( 1,
'USER2', '1978-10-3', 40 );",<not logged>

AUDIT: SESSION,WRITE,2018-11-28 10:48:34
JST,[local],34916,psql,fepuser,postgres,3/5,4,1,INSERT,,TABLE,public.account,,"INSERT INTO account VALUES ( 1,
'USER3', '1994-1-19', 24 );",<not logged>

AUDIT: SESSION,READ,2018-11-28 10:48:51
JST,[local],34916,psql,fepuser,postgres,3/7,5,1,SELECT,,TABLE,public.account,,SELECT * FROM account WHERE age >=
40;,<not logged>
```

このように、データベースの監査ログを活用することでセキュリティの脅威を検知できるため、“あんしん”に繋がるセキュリティ対策の実現に役立ちます。

2025 年 5 月 26 日