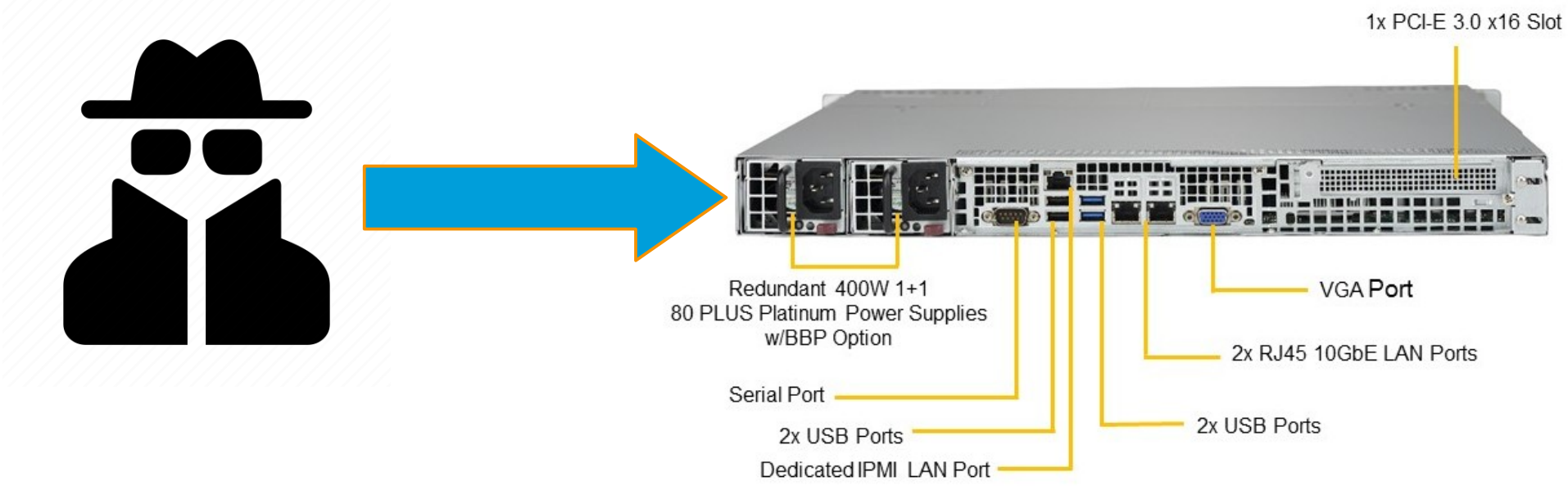
Анализ поверхности атаки приложений и программных систем

Довгалюк Павел, ИСП РАН
t.me/pavel_dovgalyuk

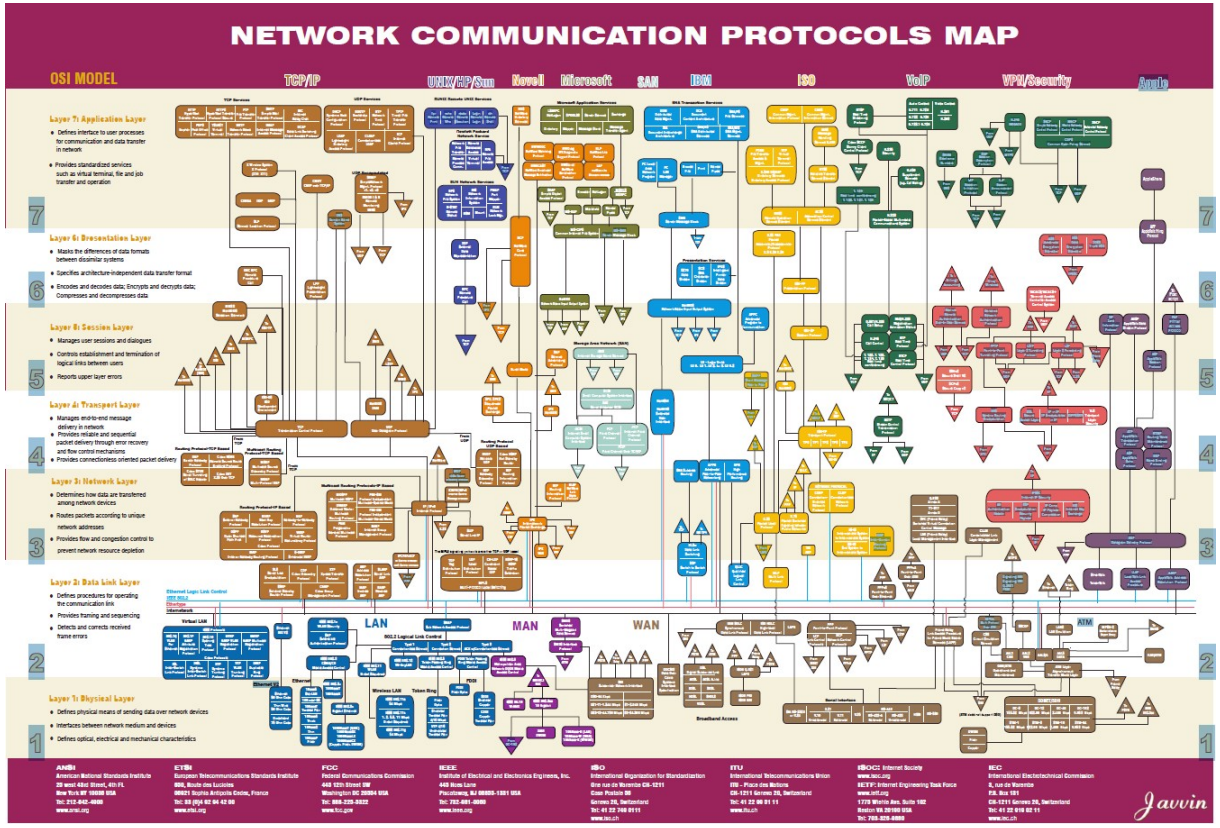
План доклада

- Поверхность атаки
- Статический анализ
- Динамический анализ
- Зачем искать потоки данных
- Примеры инструментов для анализа
- Использование Natch для поиска поверхности атаки

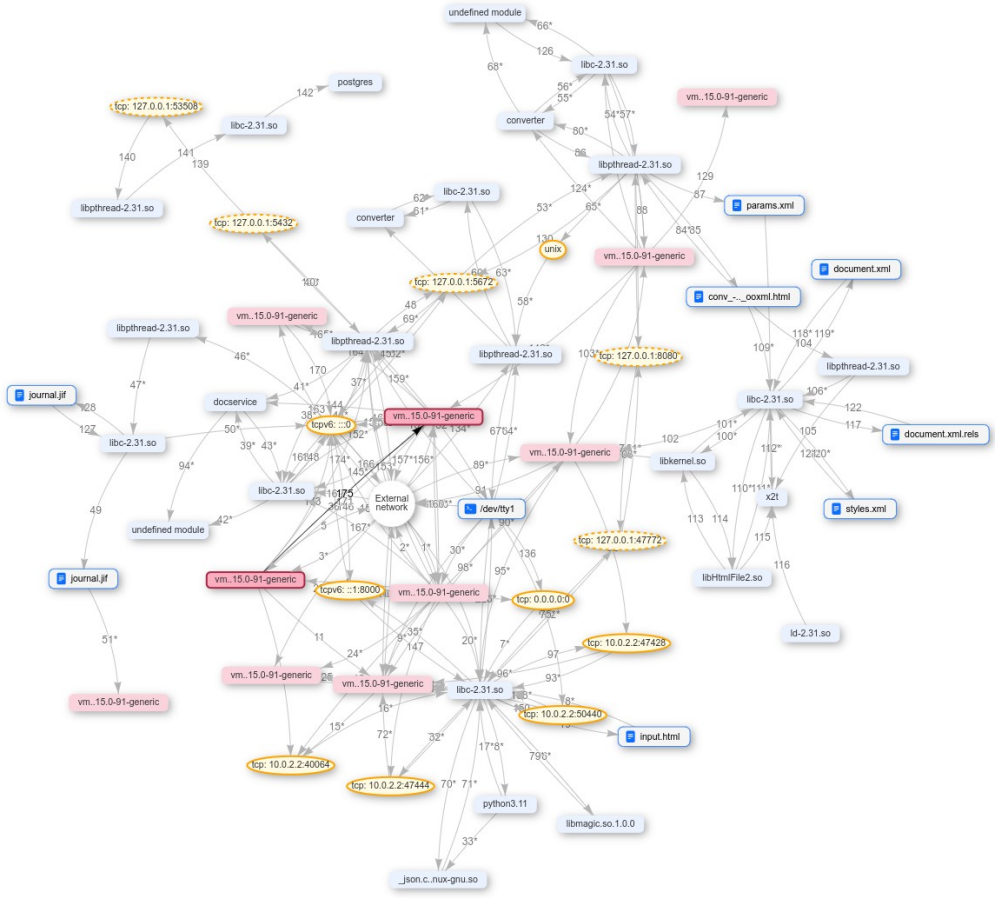
Что такое поверхность атаки



Поверхность атаки – протоколы



Поверхность атаки – не только интерфейсы



Что такое поверхность атаки

- Уязвимые места системы
- Интерфейсы
  - Порты
  - Домены
  - Адреса
  - Файлы
- Программы
  - Заимствованные компоненты
  - Собственные участки кода

Зачем искать поверхность атаки

- Повышение защищённости
  - Тестирование
  - Сокращение уязвимых мест
- Поиска только ошибок в коде недостаточно
  - Ошибок слишком много
  - Не все приводят к сбоям
  - Не все будут уязвимостями

Статический и динамический анализ

- Статический
  - Без запуска приложения
  - Весь код доступен
  - Неизвестно, как в реальности идёт выполнение
  - Входные данные неизвестны
  - Сложно анализировать взаимодействие программ
- Динамический
  - Программа работает в обычном режиме
  - На входе реальные данные
  - Не все функции активируются
  - Сценарии работы могут быть длительными
  - Замедление работы из-за анализа

Точную поверхность атаки найти невозможно

- Сложные библиотечные зависимости

```
sprintf(name, “prefix_%s_suffix.so“, input);
h = dlopen(name, RTLD_NOW);
```

- Изменчивость работы программы

```
if (time(NULL) % 1000 == 0)
    recvfrom(sfd, buf, BUF_SIZE, 0, &addr, &addrlen);
```

Точную поверхность атаки найти невозможно

- Инвариантные вычисления

```
x = read();
y = x + 3;
z = y − x;
```

- Неявные потоки данных

```
if (x & 1)
    y = 1;
else
    y = 0;
```

Уровни анализа поверхности атаки

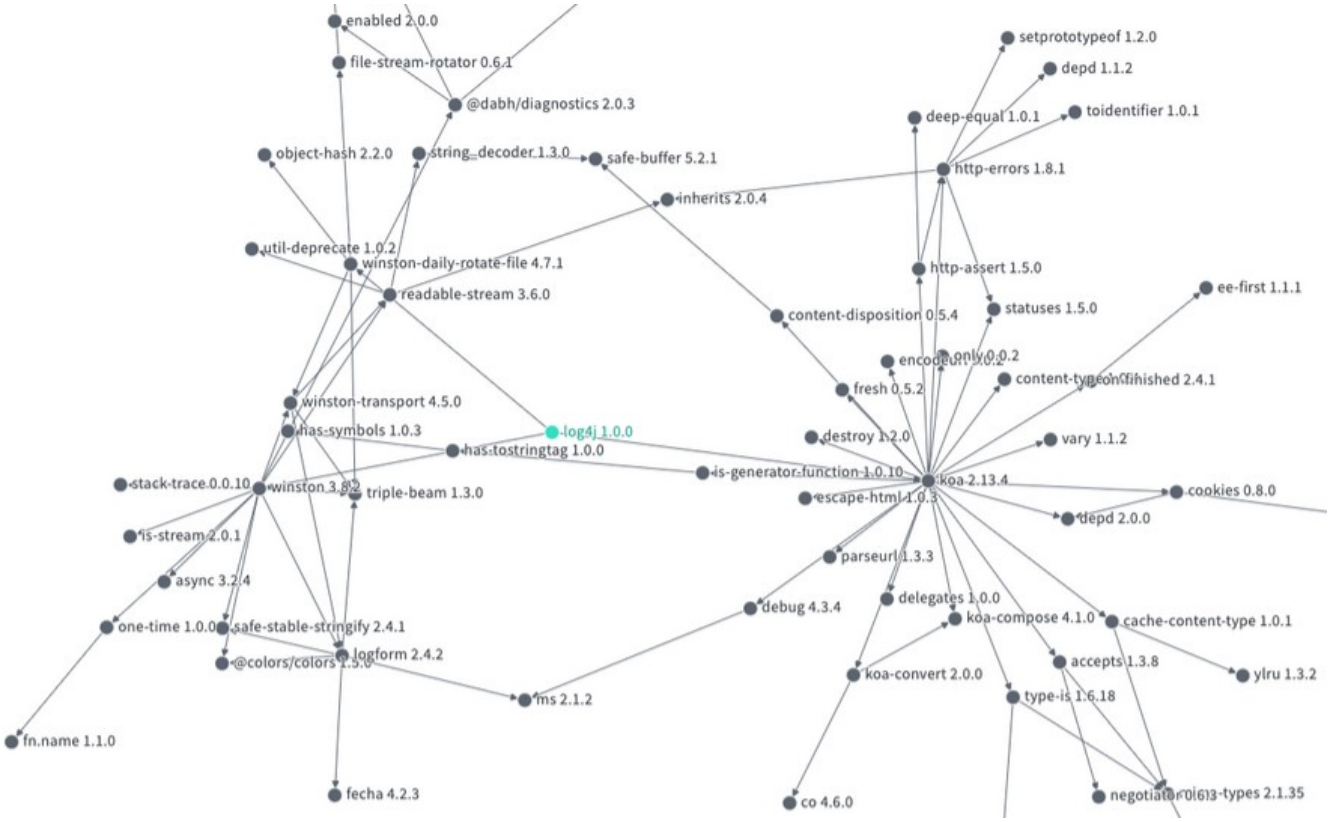
- Интерфейсы системы
- Интерфейсы приложения
  - явно и неявно определённые
- Компоненты системы
- Компоненты приложения
  - в том числе заимствованные
- Функции приложения

Уровни анализа поверхности атаки

- Интерфейсы системы
- Интерфейсы приложения
  - явно и неявно определённые
- Компоненты системы
- Компоненты приложения
  - в том числе заимствованные
- Функции приложения

Динамическое и статическое сканирование

Использование уязвимого пакета



Уровни анализа поверхности атаки

- Интерфейсы системы
- Интерфейсы приложения
  - явно и неявно определённые
- Компоненты системы
- Компоненты приложения
  - в том числе заимствованные
- Функции приложения

Динамическое и статическое сканирование

Анализ потоков данных

Внешние подключения

- Адреса сайтов
  - Кто туда обратился?
- Открытые порты
  - Какое приложение обрабатывает запрос?
- Web API
  - Что открыто и кто за это отвечает?

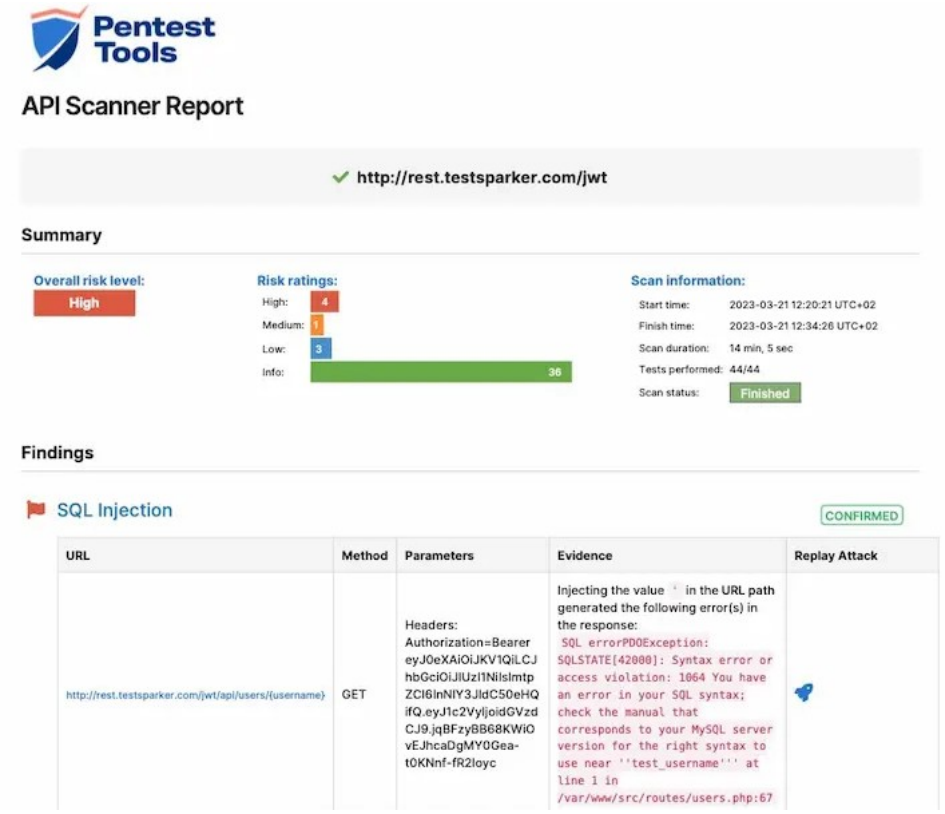
Пример проблемы

- API9:2023 Improper Inventory Management

- Бета-версия сайта с открытым отладочным API
- Забытая старая версия после переезда
- Временный ftp-сервер

Сканеры API

- Pentest tools
- BurpSuite
- Nikto
- …



Pentest Tools

API Scanner Report

http://rest.testsparker.com/jwt

Summary

Overall risk level: High

Risk ratings: High: 4, Medium: 1, Low: 3, Info: 36

Scan information:

Start time:	2023-03-21 12:20:21 UTC+02
Finish time:	2023-03-21 12:34:26 UTC+02
Scan duration:	14 min, 5 sec
Tests performed:	44/44
Scan status:	Finished

Findings

SQL Injection — CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://rest.testsparker.com/jwt/api/users/{username}	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc	Injecting the value ' in the URL path generated the following error(s) in the response: SQL errorPDOException: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''test_username''' at line 1 in /var/www/src/routes/users.php:67	

Уровень приложений (интерфейсы)

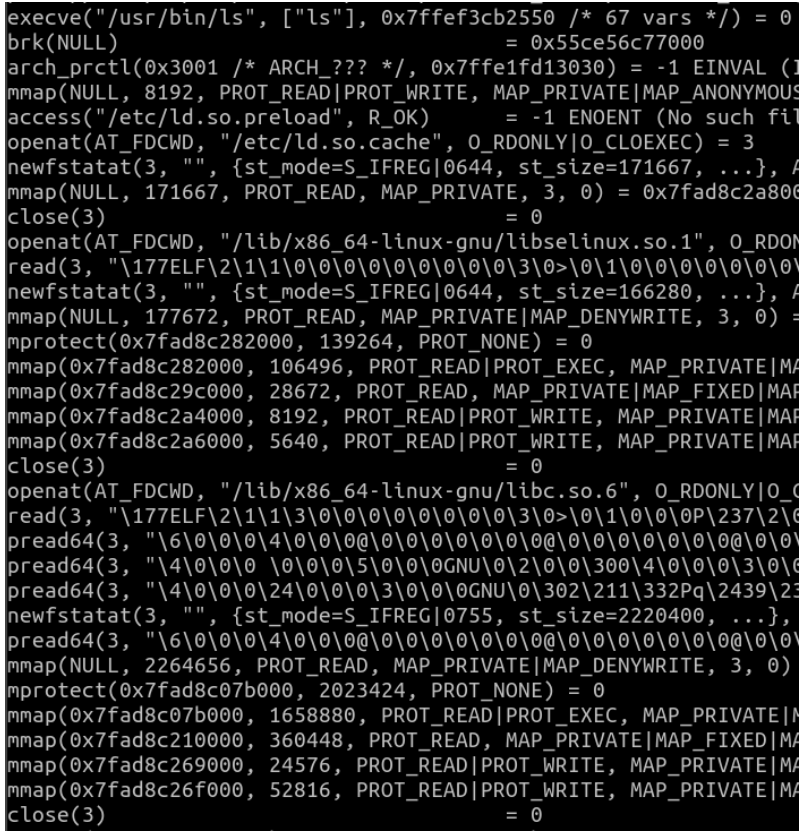
- Сокеты
- Файлы
- Системные вызовы
- Разделяемая память
- Права доступа

Примеры проблем

- Подозрительные сетевые соединения
- Утечка чувствительных данных в лог
- Запуск процессов с избыточными правами
- Некорректная конфигурация контейнеров

strace

- Обращения к файлам
- Работа с сокетами
- Взаимодействие с процессами
- Запуск программ



auditd

- Отслеживает системные вызовы и действия с файлами
- Можно создавать фильтры для нетипичных событий, чтобы расследовать инциденты или реагировать на них

```
## User, group, password databases
-w /etc/group -p wa -k etcgroup
-w /etc/passwd -p wa -k etcpasswd
-w /etc/gshadow -k etcgroup
-w /etc/shadow -k etcpasswd
-w /etc/security/opasswd -k opasswd
-w /etc/adduser.conf -k adduserconf
```

Microsoft Attack Surface Analyzer

- Записывает изменения в системе после установки приложения
- Файлы, ключи реестра, сетевые шары, различные хэндлы, запущенные процессы и сервисы, элементы автозагрузки

https://github.com/microsoft/AttackSurfaceAnalyzer

Выводы: анализ интерфейсов

- Можно построить поверхность атаки
  - Web API
  - Открытые файлы
  - Сетевые подключения
  - Запускаемые процессы
  - Используемые пременные окружения
- Возможно потребуется создание дополнительных скриптов с собственными правилами

Уровень приложений (структура)

- Подгружаемые пакеты и библиотеки
- Транзитивные зависимости
- Заимствованный код
- Секреты в коде

Пример проблемы

Special thanks to @casdr for the logo

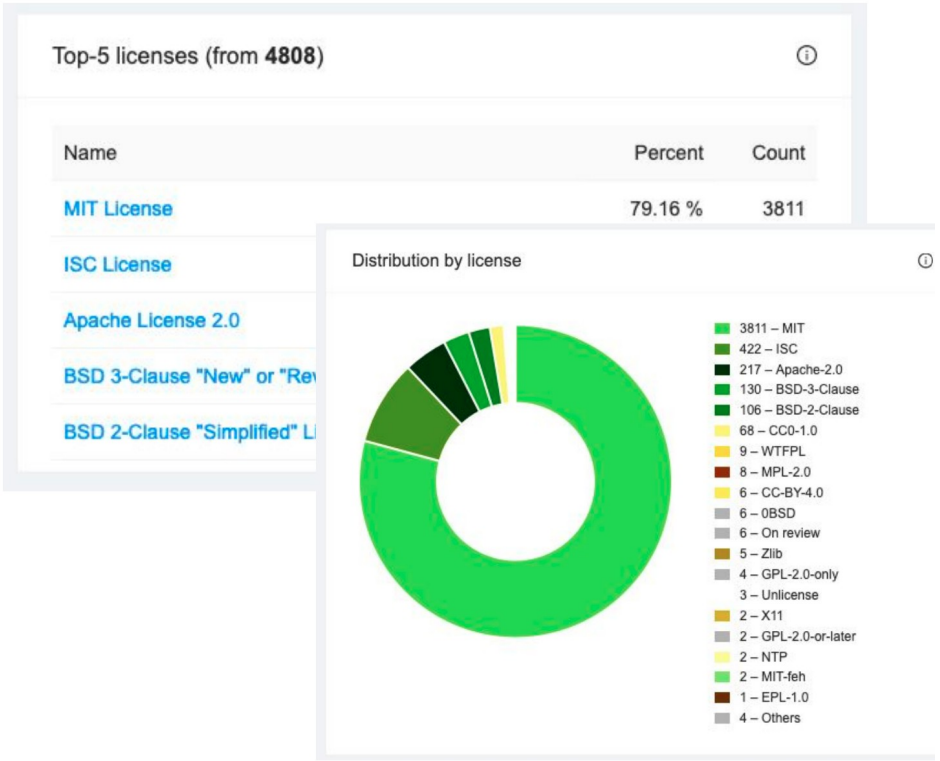
is-thirteen

build unknown GITTER join chat

An npm package to check if a number is equal to 13.

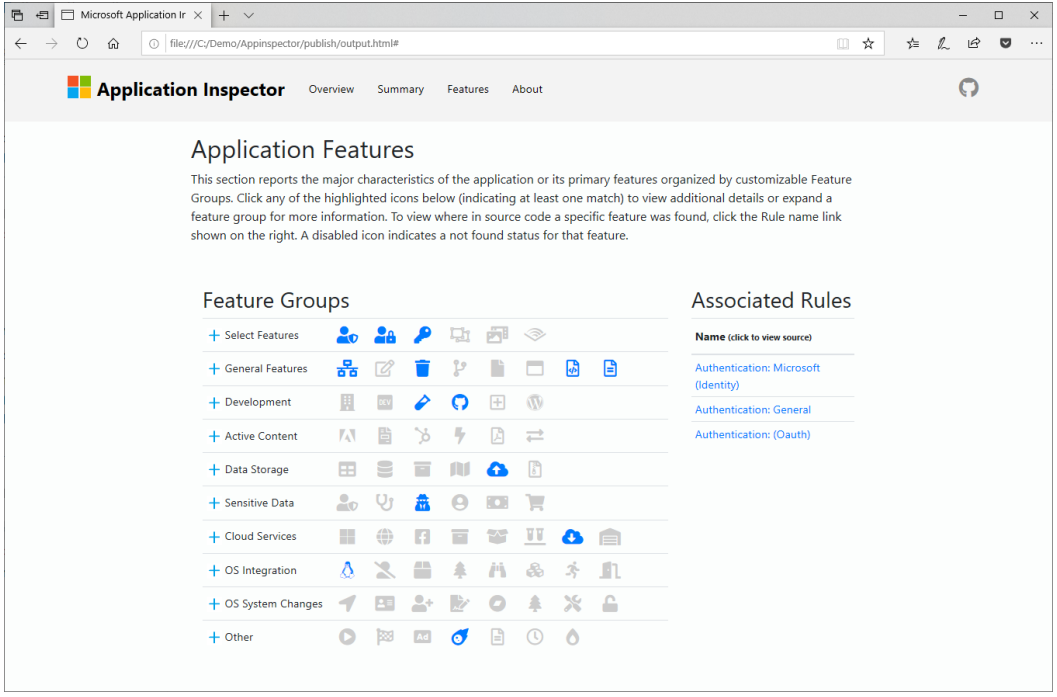
Композиционный анализ (CodeScoring)

- Инвентаризация кода
- Определение лицензий
- Оценка качества



Microsoft Application Inspector

- Поиск „интересных“ мест в коде по шаблонам
- Нет анализа потоков данных



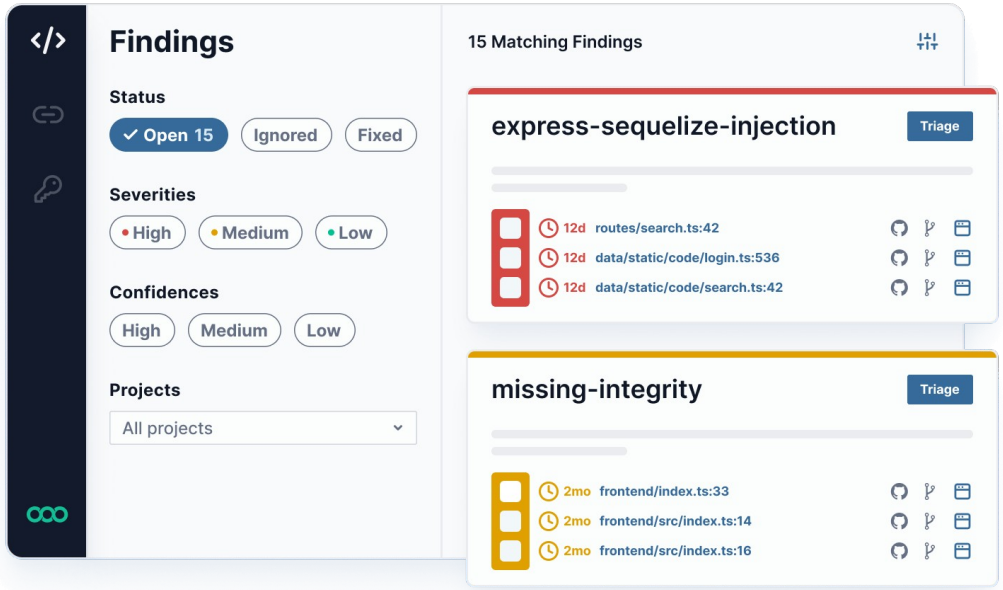
https://github.com/microsoft/ApplicationInspector

Статический анализ на уровне исходного кода

- Пересекается с поверхностью атаки
- Уязвимые функции
  - Статические анализаторы
- Заимствованные фрагменты
  - Могут быть скопированы из проектов с уязвимостями

Semgrep

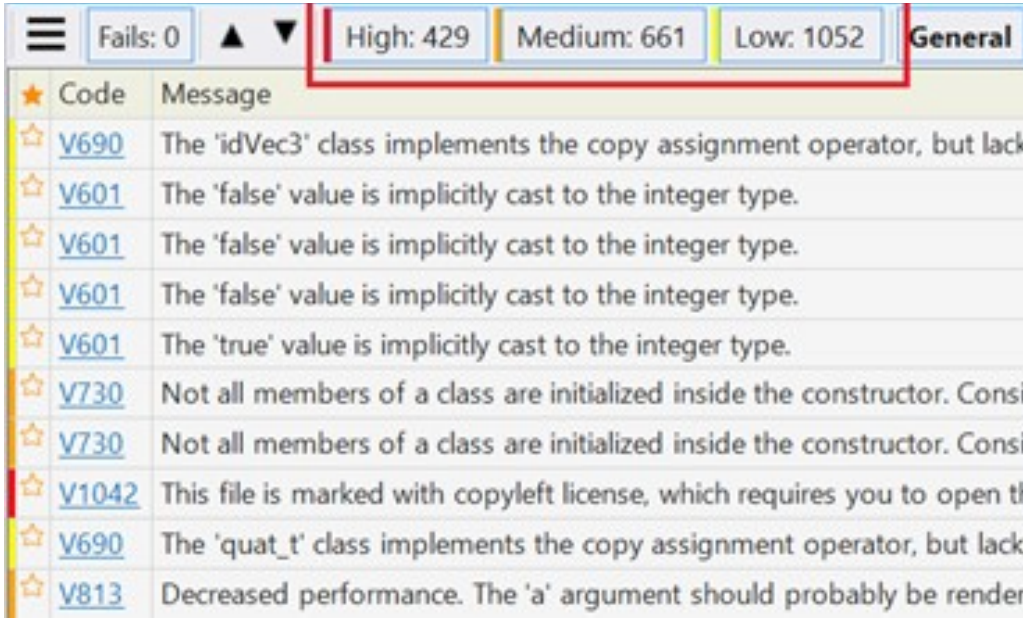
- Анализ кода по шаблонам
- Уязвимости безопасности - инъекции SQL, XSS, небезопасные вызовы функций, использование устаревших алгоритмов шифрования
- Ошибки в коде - неправильное использование операторов, ошибки стиля
- Проблемы производительности - медленные запросы к базе данных, ненужные вычисления, длинные циклы



https://semgrep.dev/

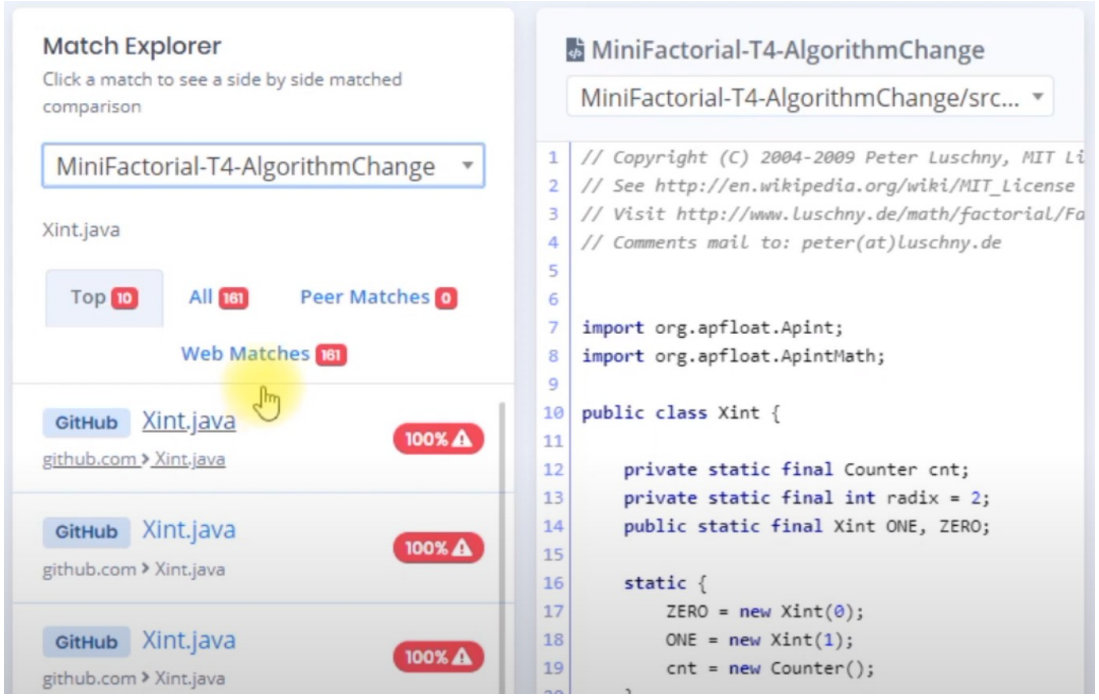
Статические анализаторы

- Много предупреждений
- Не привязать ошибки к поверхности атаки
- Слабые возможности межмодульного анализа – нет реальных входных данных



Анализ заимствований (Codequiry)

- Ищет фрагменты кода, взятые из открытых проектов
- Принадлежность фрагментов к поверхности атаки придётся определить самостоятельно



https://codequiry.com/

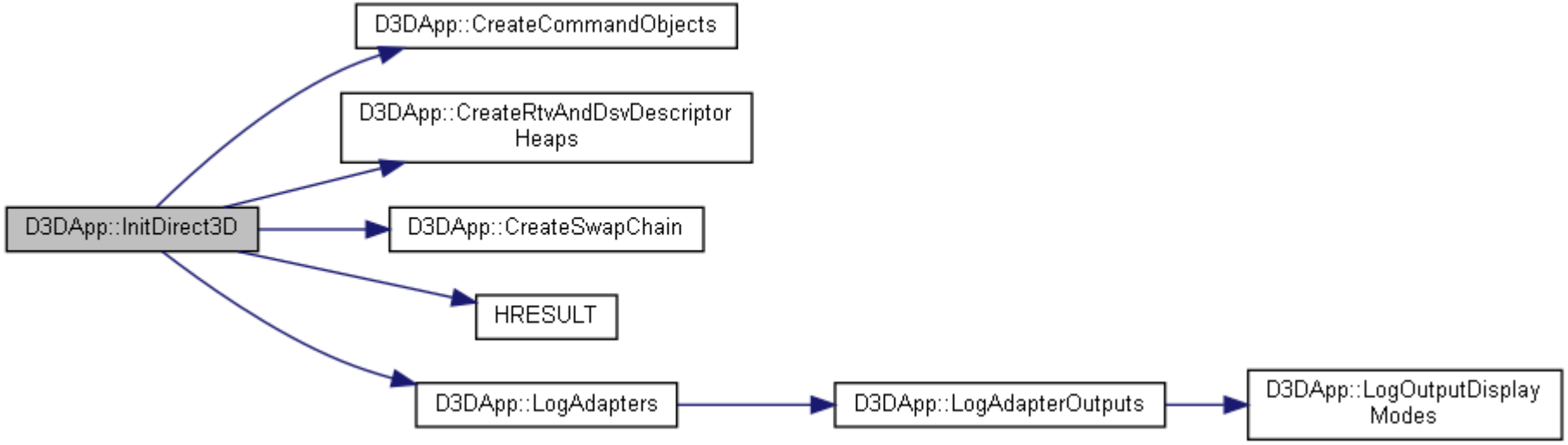
Ручной анализ

- IDE
- Генераторы документации
- Дизассемблеры

Граф зависимостей/вызовов

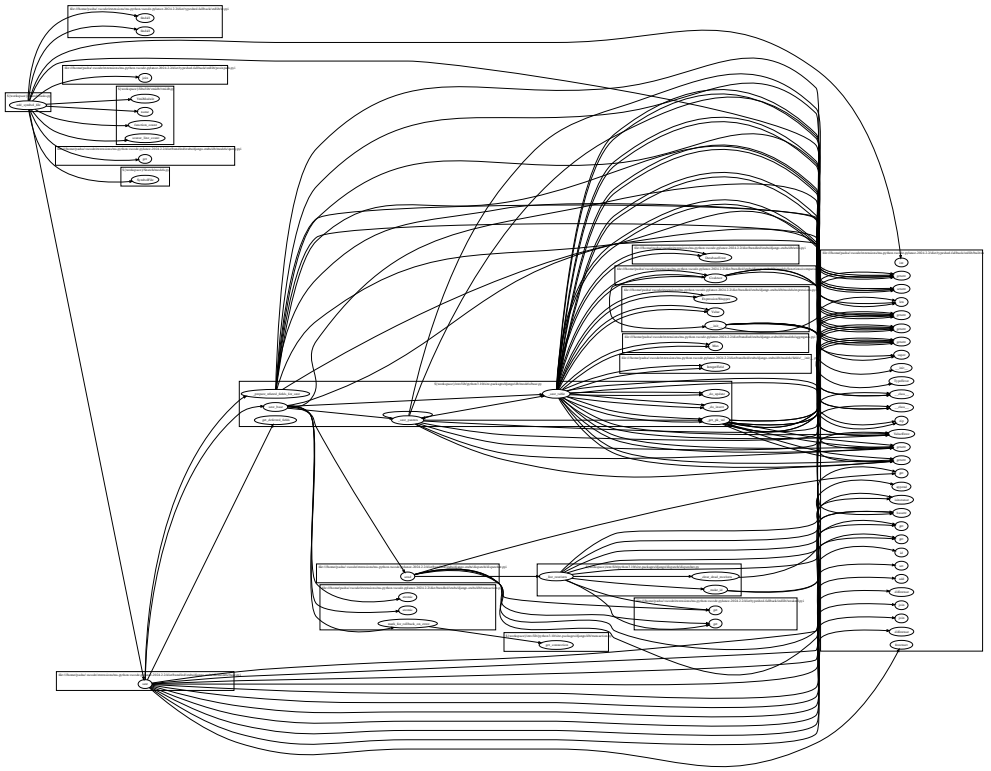
- Графы зависимостей
- Графы вызовов
- Doxygen
- Плагин Call graph для VS Code

Граф зависимостей/вызовов



Недостатки графов зависимостей

- Нет динамических зависимостей
- Слишком сложные картинки
- Нет удобной навигации
- Не связан с поверхностью атаки



Динамический анализ

- Интерактивная отладка (gdb)
- Построение покрытия (gcov)
- Создание инструментов для анализа (valgrind, dynamo rio, panda)
- Автоматический анализ поверхности атаки (natch)

Интерактивный отладчик

- Анализ конкретного сценария

```
(gdb) bt
#0  __pthread_kill (threadid=<optimised out>, signo=signo@entry=6) at ../sysdeps/unix/sysv/linux/pthread_kill.c:56
#1  0x00005573ec8a1ac9 in my_write_core (sig=sig@entry=6) at /home/roel/mysql-server_dbg/include/my_thread.h:88
#2  0x00005573eb4aacdc in handle_fatal_signal (sig=6) at /home/roel/mysql-server_dbg/sql/signal_handler.cc:171
#3  <signal handler called>
#4  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
#5  0x00007f24e59d3859 in __GI_abort () at abort.c:79
#6  0x00005573ecd0ddfd in ut_dbg_assertion_failed (expr=expr@entry=0x0,
    file=file@entry=0x5573edf64f40 "/home/roel/mysql-server_dbg/storage/innobase/ut/ut0ut.cc", line=line@entry=555)
    at /home/roel/mysql-server_dbg/storage/innobase/ut/ut0dbg.cc:99
#7  0x00005573ecd1a91c in ib::fatal::~fatal (this=0x7f24d07591d0, __in_chrg=<optimised out>) at /home/roel/mysql-server_dbg/storage/innobase/ut/ut0ut.cc:555
#8  0x00005573ec980550 in innobase_post_ddl (thd=<optimised out>) at /usr/include/c++/9/ext/new_allocator.h:89
#9  0x00005573eb3cef6b in mysql_create_table (thd=thd@entry=0x7f2424d05f30, create_table=create_table@entry=0x7f2424d0de50,
    create_info=create_info@entry=0x7f24d07595f0, alter_info=alter_info@entry=0x7f24d0759730) at /home/roel/mysql-server_dbg/sql/sql_table.cc:9938
#10 0x00005573eb8d8bdb in Sql_cmd_create_table::execute (this=0x7f2424d0ed60, thd=0x7f2424d05f30) at /home/roel/mysql-server_dbg/sql/sql_cmd_ddl_table.cc:406
#11 0x00005573eb2f204c in mysql_execute_command (thd=thd@entry=0x7f2424d05f30, first_level=first_level@entry=true)
    at /home/roel/mysql-server_dbg/sql/sql_parse.cc:3426
#12 0x00005573eb2f5bcc in dispatch_sql_command (thd=thd@entry=0x7f2424d05f30, parser_state=parser_state@entry=0x7f24d075ab90)
    at /home/roel/mysql-server_dbg/sql/sql_parse.cc:5000
#13 0x00005573eb2f8069 in dispatch_command (thd=thd@entry=0x7f2424d05f30, com_data=com_data@entry=0x7f24d075bba0, command=COM_QUERY)
    at /home/roel/mysql-server_dbg/sql/sql_parse.cc:1841
#14 0x00005573eb2fab48 in do_command (thd=thd@entry=0x7f2424d05f30) at /home/roel/mysql-server_dbg/sql/sql_parse.cc:1320
#15 0x00005573eb498c17 in handle_connection (arg=arg@entry=0x5573f1bfb7d0) at /home/roel/mysql-server_dbg/sql/conn_handler/connection_handler_per_thread.cc:301
#16 0x00005573ed0ec690 in pfs_spawn_thread (arg=0x5573f1b2d150) at /home/roel/mysql-server_dbg/storage/perfschema/pfs.cc:2898
#17 0x00007f24e636a609 in start_thread (arg=<optimised out>) at pthread_create.c:477
#18 0x00007f24e5ad0293 in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95
(gdb)
```

Инструментирование

- Добавление вспомогательного кода для анализа
  - Отслеживание потоков данных
  - Логирование
  - Получение сведений из внутренностей ОС
- Замедляет работу программы

Инструментирование в QEMU

0xb7707010: mov %ebx,%edx

0xb7707012: mov 0x8(%esp),%ecx

0xb7707016: mov 0x4(%esp),%ebx

0xb770701a: mov $0x21,%eax

0xb770701f: int $0x80

Инструментирование в QEMU

0xb7707010: mov %ebx,%edx

0xb7707012: mov 0x8(%esp),%ecx

0xb7707016: mov 0x4(%esp),%ebx

0xb770701a: mov $0x21,%eax

0xb770701f: int $0x80

---- b770701f 00000000

movi_i32 tmp3,$0xffffffffb770701f

st_i32 tmp3,env,$0x20

movi_i32 tmp11,$0x2

movi_i32 tmp12,$0x80

call raise_interrupt,$0x0,$0,env,tmp12,tmp11

Инструментирование в QEMU

0xb7707010: mov %ebx,%edx

0xb7707012: mov 0x8(%esp),%ecx

0xb7707016: mov 0x4(%esp),%ebx

0xb770701a: mov $0x21,%eax

0xb770701f: int $0x80

---- b770701f 00000000

movi_i64 tmp13,$0xb7707020

movi_i64 tmp14,$0x7fef9a788670

call start_system_call,$0x0,$0,tmp13,tmp14

movi_i32 tmp3,$0xfffffffb770701f

st_i32 tmp3,env,$0x20

movi_i32 tmp11,$0x2

movi_i32 tmp12,$0x80

call raise_interrupt,$0x0,$0,env,tmp12,tmp11

Построение покрытия (gcov)

- Полный набор выполненных функций и операторов
- Пересечение с поверхностью атаки нужно искать отдельно

```
int performCheck()
{
    vector<SimEvent*> listOfSimEvents = getSimEventList();
    vector<SimEventSummary*> eventSummaries;

    for (auto it = listOfSimEvents.begin(); it != listOfSimEvents.end();
        ++it)
    {
        ArchState currentState = (*it)->getArchState();
        if(debug) {
                STDOUT << "Operating on SimEvent from Processor "
                       << currentState->getProcId()->toString() << endl;
        }

        if(currentState->getOperation() == OpType::DMA_TYPE_1) {
                processDMA(currentState);
                continue;
        }

        if(isInvalid(eventSummaries, currentState)) {
                dumpSimLog(currentState->toString());
                return FAIL;
        }

        eventSummaries.push_back((*it)->getSummary());
        processState(currentState);
        ...
    }
    return PASS;
}
```

Valgrind, DynamoRio

- Уровень приложений
- Можно использовать как основу для собственных инструментов
- Нет готовых решений для поиска поверхности атаки

```
aqsa@aqsa-VirtualBox:~$ g++ -Wall -pedantic -g -o file1 file1.c
aqsa@aqsa-VirtualBox:~$ valgrind ./file1
==5143== Memcheck, a memory error detector
==5143== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5143== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==5143== Command: ./file1
==5143==
==5143== Conditional jump or move depends on uninitialised value(s)
==5143==    at 0x48D5AD8: __vfprintf_internal (vfprintf-internal.c:1687)
==5143==    by 0x48BFEBE: printf (printf.c:33)
==5143==    by 0x1091F2: main (file1.c:11)
==5143==
==5143== Use of uninitialised value of size 8
==5143==    at 0x48B981B: _itoa_word (_itoa.c:179)
==5143==    by 0x48D56F4: __vfprintf_internal (vfprintf-internal.c:1687)
==5143==    by 0x48BFEBE: printf (printf.c:33)
==5143==    by 0x1091F2: main (file1.c:11)
```

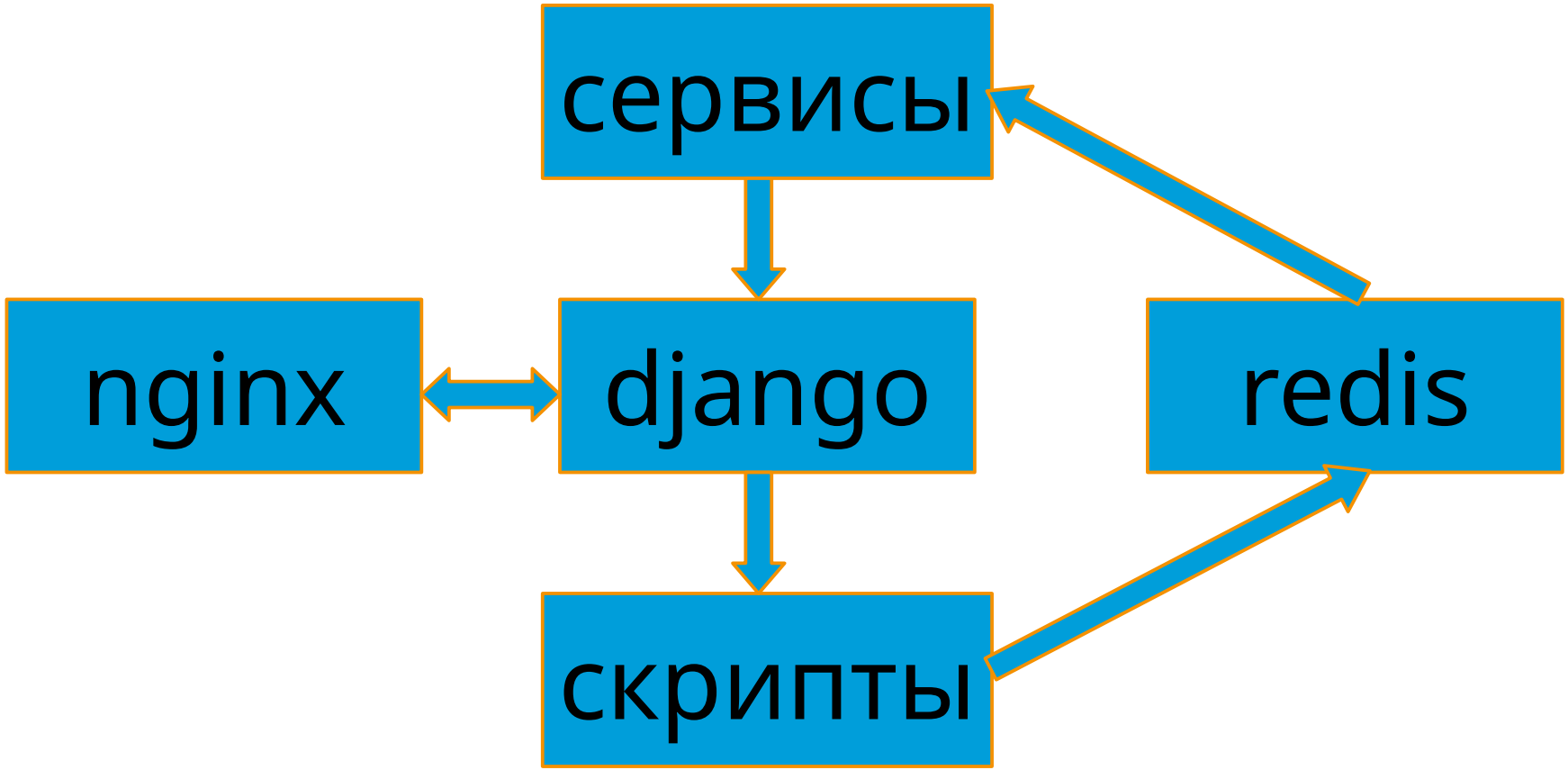
https://valgrind.org/

https://dynamorio.org/

Выводы: анализ кода

- Поверхность атаки автоматически не ищется
  - Ручное тестирование нельзя встроить в CI/CD
- Поверхность атаки без анализа потоков данных
  - Заимствованные компоненты
  - Покрытие кода

Многокомпонентные системы

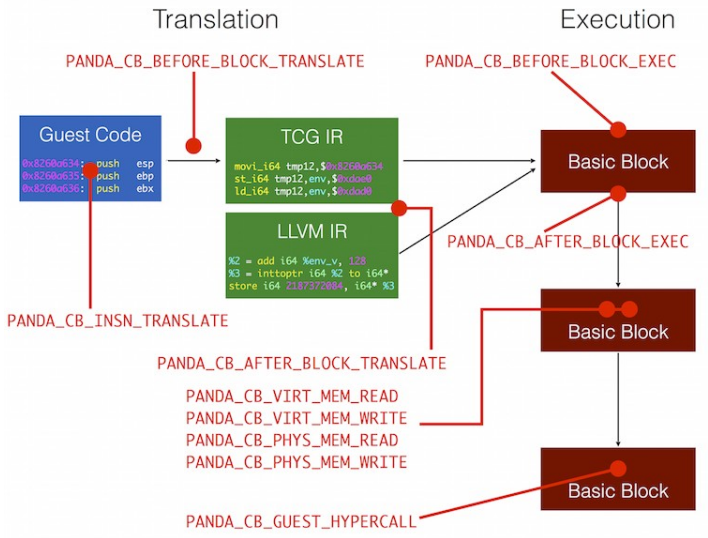


Полносистемный анализ

- Приложения связаны – поверхность атаки одного приложения искать бесполезно
- Не всегда используются компилируемые языки
- Эмулятор QEMU
  - Разные аппаратные платформы
  - Можно добавить инструментирование

Panda

- Фреймворк для полносистемного анализа
- Расширяется с помощью плагинов

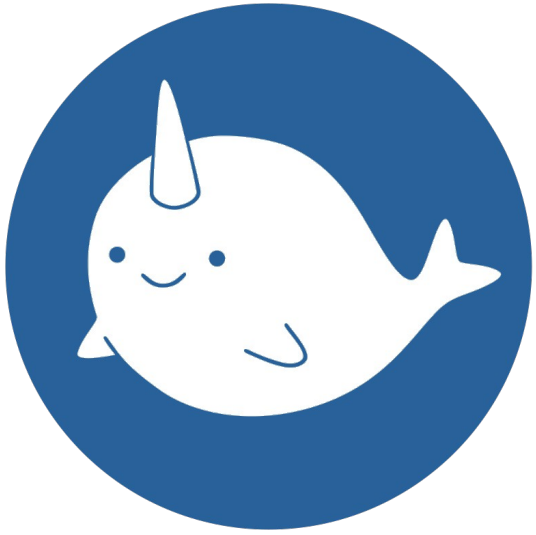


https://github.com/panda-re/panda/

Недостатки Panda

- Нужно разбираться во внутренностях QEMU
- Нет готовых рецептов/плагинов для анализа
- Нет визуализации результатов
- Нет поддержки анализа скриптовых языков и байткода
- Недостаточно гибкая поддержка разных ОС

Natch

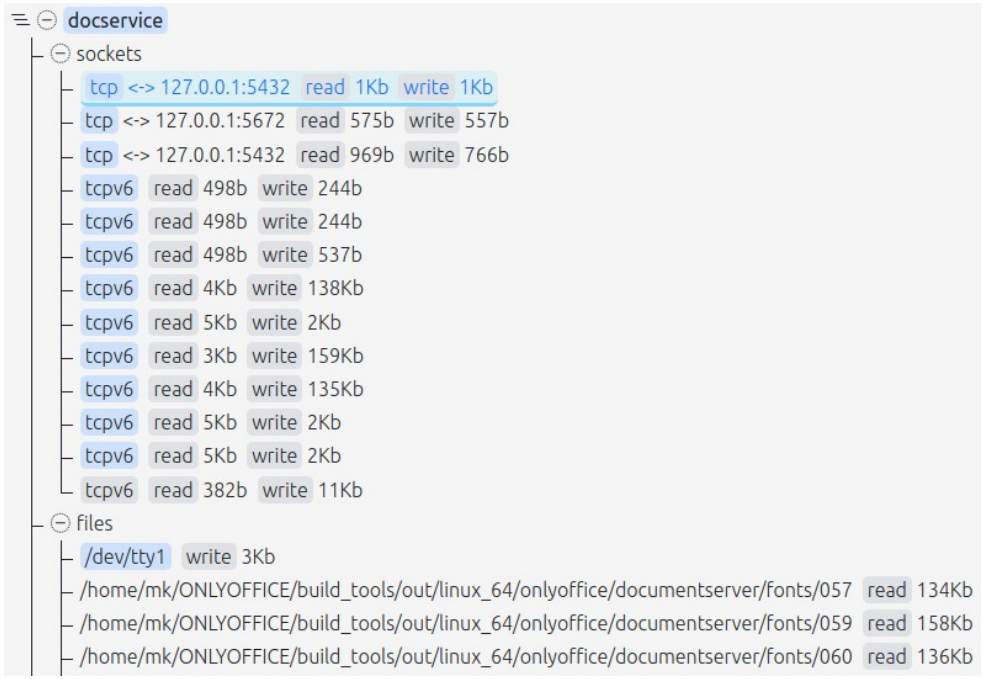


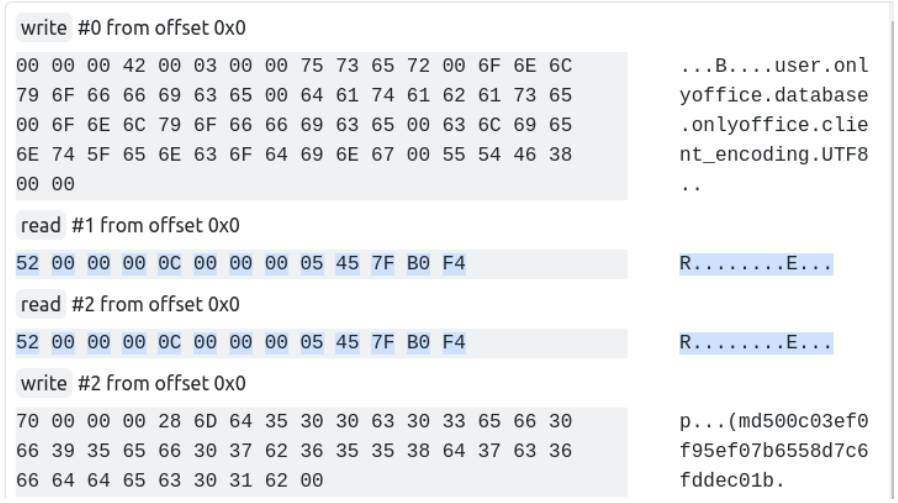
- Полносистемный
- Динамический анализ помеченных данных
- Включает возможности
  - ps
  - strace
  - gcov
  - gdb
  - …

Natch: список процессов

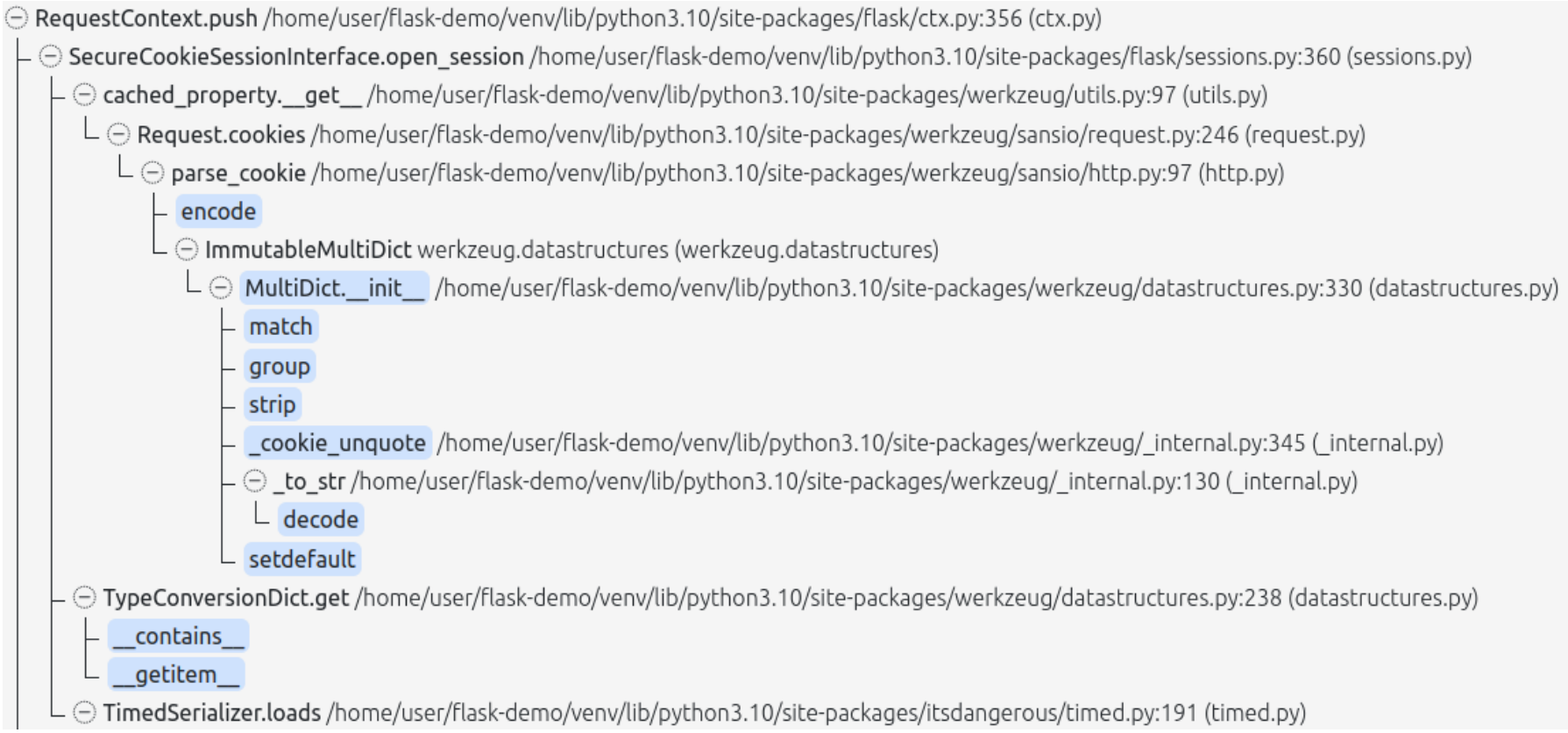
```
systemd  pid: 1  uid: 0 (root)
  login  pid: 460  uid: 0 (root)
    bash  pid: 888  uid: 1000 (mk)
      bash  pid: 1644  uid: 1000 (mk)
        make  pid: 1645  uid: 1000 (mk)
          python  pid: 1646  uid: 1000 (mk)
            python  pid: 1679  uid: 1000 (mk) : /usr/bin/python manage.py runserver
      bash  pid: 1619  uid: 1000 (mk)
        converter  pid: 1620  uid: 1000 (mk) : ./converter
          converter  pid: 1628  uid: 1000 (mk) : /home/mk/ONLYOFFICE/build_tools/out/linux_64/onlyoffice/documentserver/server/FileConverter/converter /s
            x2t  pid: 1704  uid: 1000 (mk) : ../FileConverter/bin/x2t /tmp/ASC_CONVERT2024026-1m0wcjz.38t/params.xml
            x2t  pid: 1704  uid: 1000 (mk) : ../FileConverter/bin/x2t /tmp/ASC_CONVERT2024026-1m0wcjz.38t/params.xml
      bash  pid: 1604  uid: 1000 (mk)
        docservice  pid: 1605  uid: 1000 (mk) : ./docservice
  rabbitmq-server  pid: 1294  uid: 124 (rabbitmq)
    beam.smp  pid: 1298  uid: 124 (rabbitmq) : /usr/lib/erlang/erts-10.6.4/bin/beam.smp -W w -A 64 -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMr
  epmd  pid: 435  uid: 123 (epmd) : /usr/bin/epmd -systemd
  NetworkManager  pid: 381  uid: 0 (root) : /usr/sbin/NetworkManager --no-daemon
  postgres  pid: 626  uid: 122 (postgres) : /usr/lib/postgresql/12/bin/postgres -D /var/lib/postgresql/12/main -c config_file=/etc/postgresql/12/main/postgresql.conf
    postgres  pid: 1702  uid: 122 (postgres) : postgres: 12/main: onlyoffice onlyoffice 127.0.0.1(53508) UPDATE
```

Natch: открытые файлы и сокеты

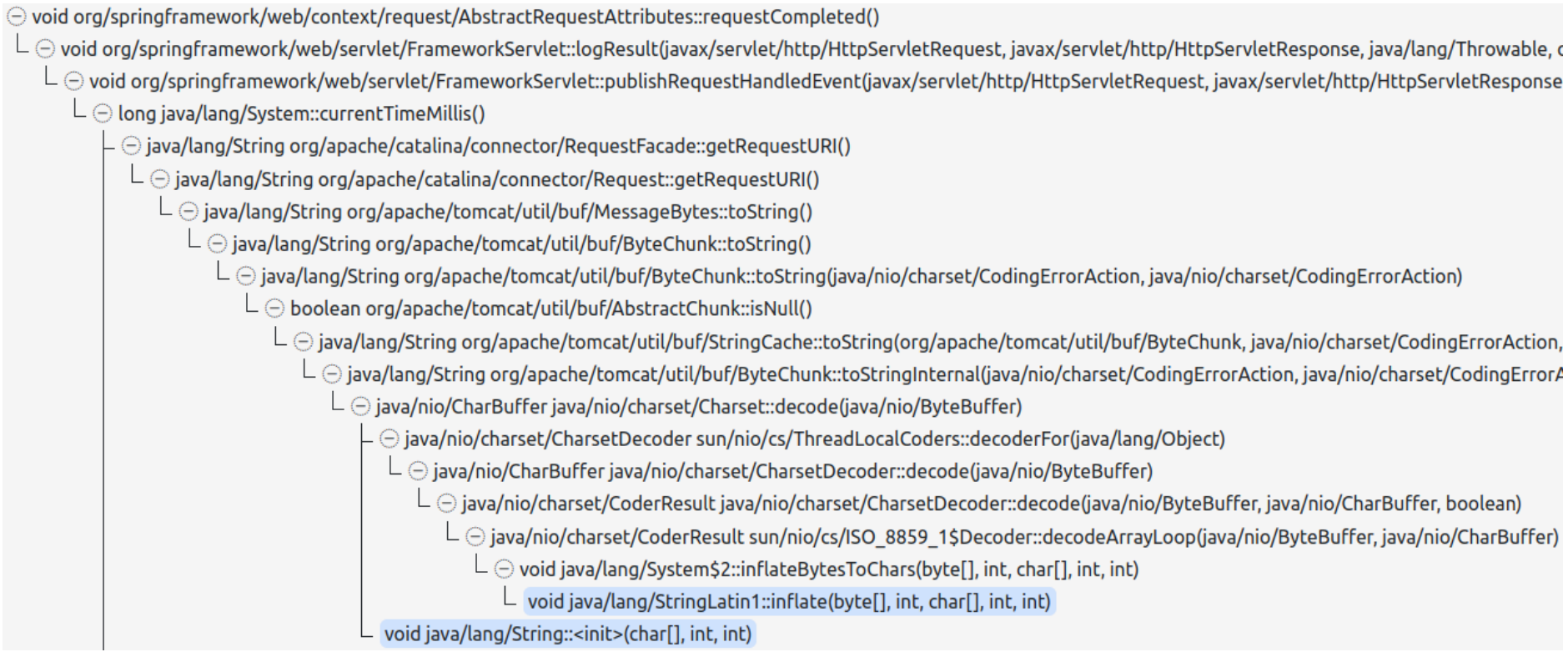




Natch: анализ скриптов (Python)



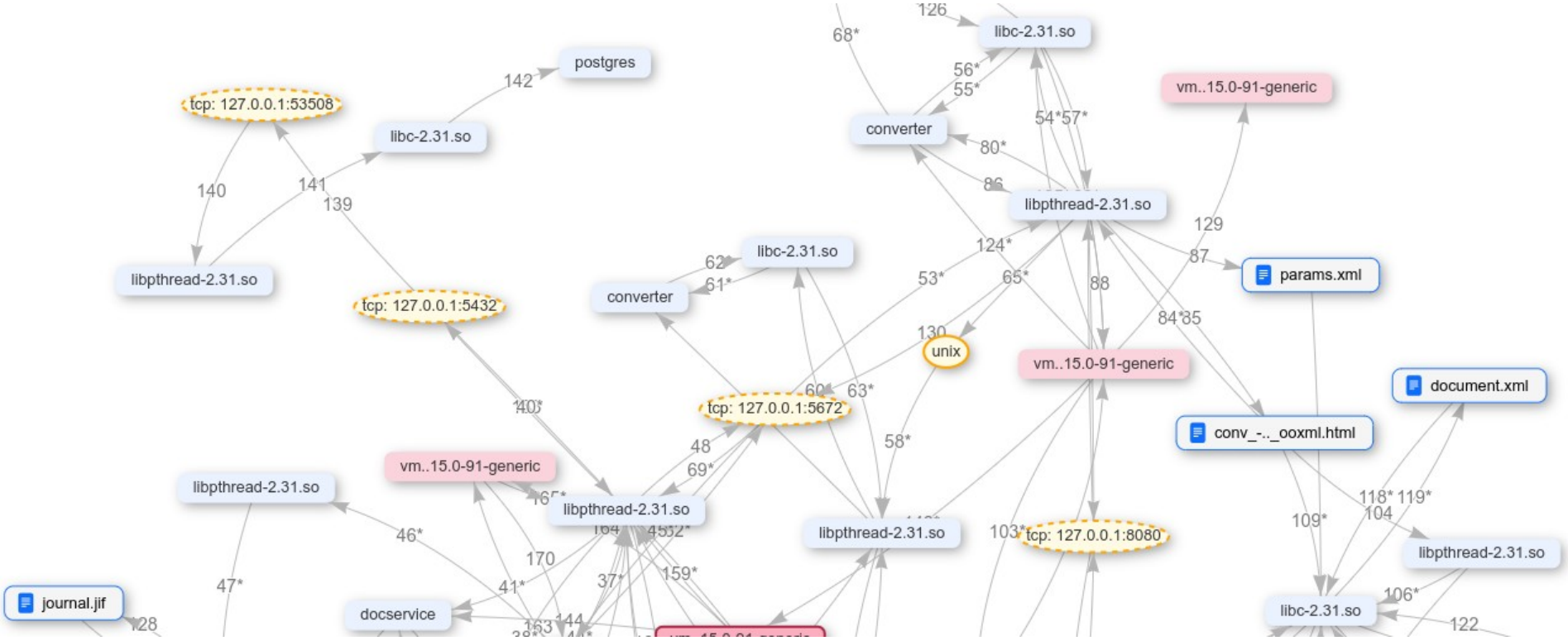
Natch: анализ байткода (Java)



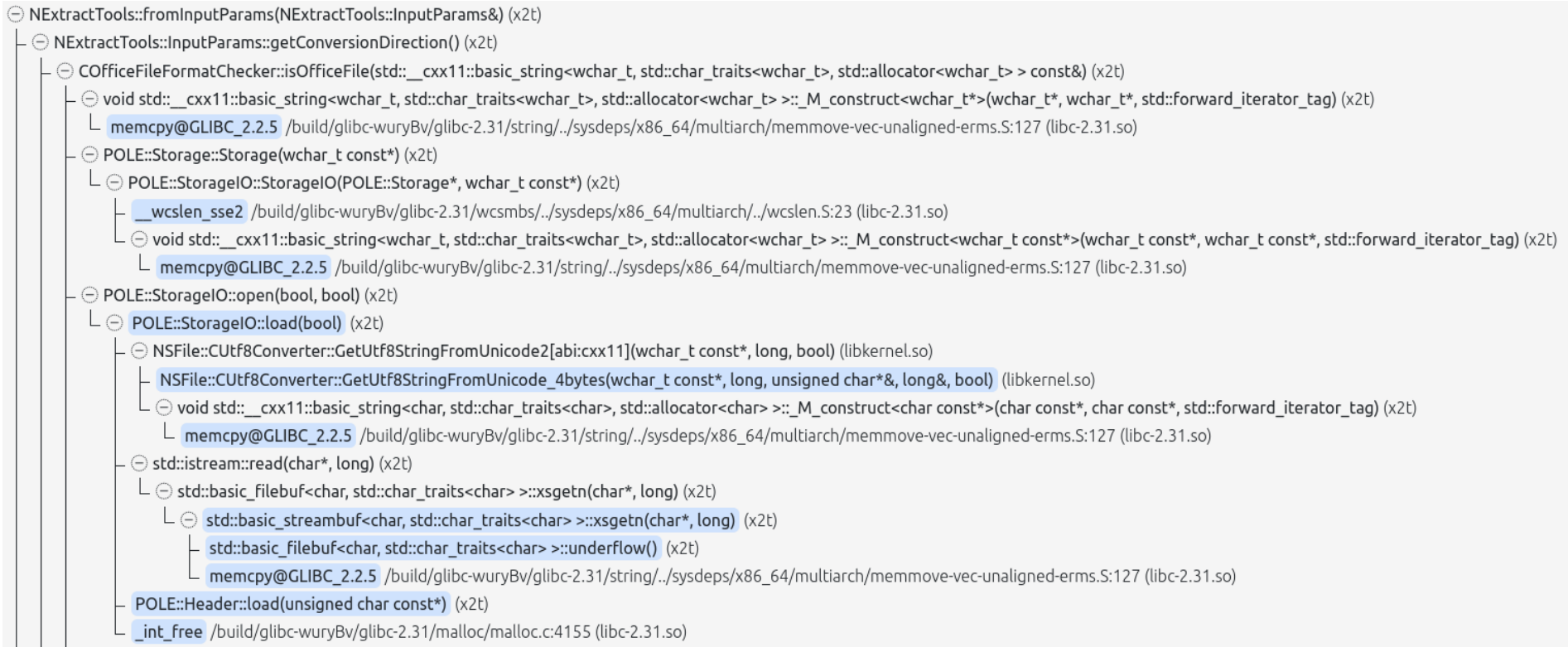
Natch: отслеживание потоков данных

- Источники данных
  - Файлы
  - Сетевые подключения
  - USB
- Получатели данных (в том числе доступные транзитивно)
  - Программы
  - Исполняемые модули
  - Функции
  - Скрипты

Natch: взаимодействие модулей



Natch: поверхность атаки в виде стека вызовов



Natch: регрессионное тестирование ПА

- Скрипты для автоматизации сценариев
- API для работы с полученной ПА

Natch: кейсы на хабре

- Анализировали 3 незнакомых приложения
  - Rizin, Assimp, ONLYOFFICE Docs
- Поверхность атаки для компилируемого кода
- Фаззингом нашли 10 сбоев
- 3 пулреквеста с исправлениями

https://habr.com/ru/companies/isp_ras/articles/788490/

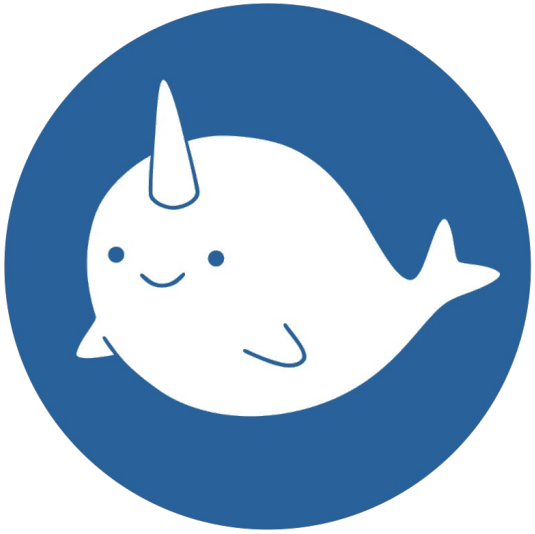
Natch

- Автоматизированный анализ поверхности атаки
  - Интерфейсы приложений
  - Компоненты и функции программ
- x86_64
- C/C++, Go, Python, Java
- Регрессионное тестирование
- Отчёты для аналитиков и тестировщиков

Телеграм-канал Natch

- https://t.me/ispras_natch
- Вопросы от пользователей
- Разборы кейсов
- Анонсы вебинаров
- Уведомления о новых релизах

Natch – система для анализа поверхности атаки



- Natch
  - https://github.com/ispras/natch
  - https://t.me/ispras_natch
- Павел Довгалюк
  - https://t.me/pavel_dovgalyuk

Выводы: анализ кода

- ~~Поверхность атаки автоматически не делается~~
- Natch строит поверхность атаки на основе анализа потоков данных
- Поверхность атаки без анализа потоков данных
  - Заимствованные компоненты
  - Покрытие кода

Иванниковские чтения 2024

- Великий Новгород, 17-18 мая
- Бесплатное участие
- Секция по эмуляторам
- Мастер-класс Natch
- Круглый стол по технологиям разработки безопасного ПО

https://www.ivannikov-ws.org/