CASR: анализ coredump файлов в ОС Linux и составление отчётов об ошибках

Федотов А.Н., к.т.н., fedotoff@ispras.ru
Курмангалеев Ш.Ф., к.ф.-м.н., kursh@ispras.ru

25 сентября 2020 г.

ИСП РАН

Сбор отчётов об ошибках и их анализ

- Не все ошибки в программе могут быть исправлены на стадии тестирования ПО.
- Пользователь/разработчик не всегда в состоянии воспроизвести ошибку.
- Автоматический анализ отчётов об ошибках позволяет упростить/ускорить дальнейшее исправление ошибок.

Существующие инструменты по сбору отчётов ошибках

- WER(Dr. Watson) – система сбора отчётов об ошибках в ОС Windows.
- Apport – система сбора отчётов ошибках ОС Ubuntu.
- Mozilla Crash Reporter, Crashpad – системы сбора отчётов об ошибках в браузерах.

Требования к инструменту сбора отчётов об ошибках

- Инструмент должен обеспечивать оценку критичности аварийных завершений.
- Инструмент должен обеспечить сбор информации, которая поможет разработчику понять причины возникновения ошибки.

Мотивация к разработке: периодически аварийно завершался компонент *auplink* из пакета auf-tools, используемый в *docker*

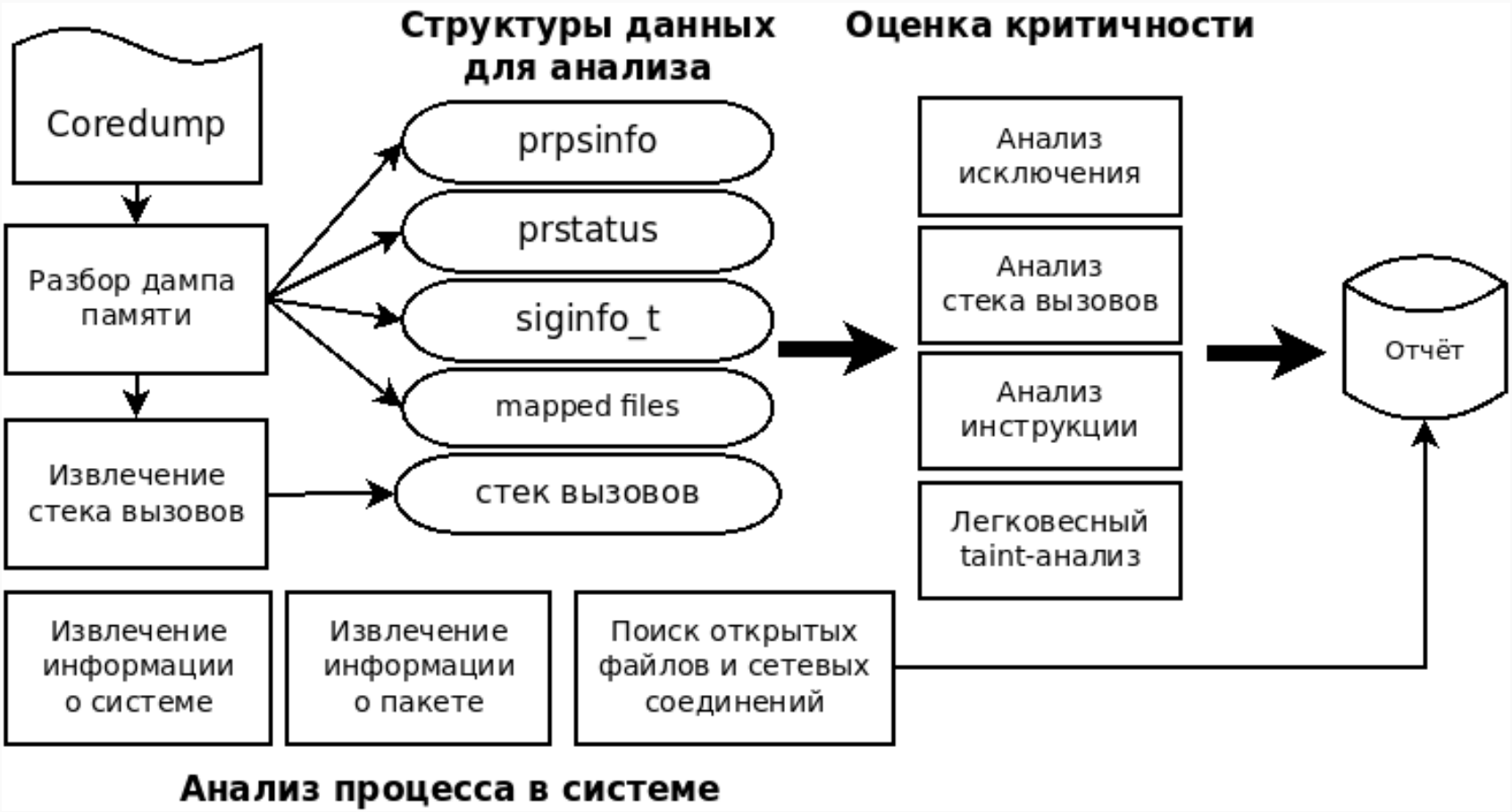
Coredump в ОС Linux

Coredump – представляет собой содержимое рабочей памяти процесса, а также полезную дополнительную информацию:

- значения регистров для всех потоков выполнения;
- информация об исключениях;
- строка запуска программы;
- карта загруженных исполняемых модулей процесса.

Coredump сохраняется в формате **elf.**
Начиная с версии 2.6.19 ядра ОС Linux coredump можно передавать на стандартный поток ввода программе для анализа.
Для этого нужно в файле */proc/sys/kernel/core_pattern* нужно указать:
|<path-to-your-program> <parameters>

Схема работы инструмента CASR



Оценка критичности

Группы классов аварийных завершений:

- **Эксплуатируемые аварийные завершения**. Перехват потока управления максимально вероятен. (Авайрное завершение на инструкции вызова/возврата и т.д.)
- **Потенциально эксплуатируемые**. Перехват управления требует незначительных действий от аналитика. (Нарушение доступа при чтении указателя из памяти).
- **Отказ в обслуживании**. Эксплуатация таких аварийных завершений маловероятна или требует ручного анализа. (Деление на ноль, разыменование нулевого указателя и т.д).

Легковесный анализ помеченных данных

Мотивирущий пример №1:

```
mov rax, qword ptr [rax] ; crash
mov rdx, qword ptr [rbp - 0x20]
mov rdi, rdx
call rax ; control flow hijack
```

Мотивирущий пример №2:

```
mov esi, dword ptr [ebx] ; crash
add esi, 4
mov dword ptr[esi], eax ; possible CWE-123
```

Правила распространения помеченных данных:

- Регистр становится помеченным при чтении по помеченному адресу
- Помеченные значения через память не распространяются
- Анализ до первой инструкции передачи управления

Пример отчёта об аварийном завершении для сервера nginx. CVE–2013–2028

```
fedotoff@tracers-laptop (192.168.88.248) - byobu
File  Edit  View  Search  Terminal  Help

Crash Report for /home/administrator/nginx/nginx
Severity: EXPLOITABLE

▼ Date
  ◦ 2020-08-15T18:07:35.524452133+03:00
▼ Uname
  ◦ Linux astra 4.15.3-1-generic #astra21 SMP Thu Aug 22 12:16:21 UTC 2019 x86_64 GNU/Linux
▼ OS
  ◦ AstraLinuxCE
▼ OSRelease
  ◦ 2.12.22
▼ Architecture
  ◦ amd64
▶ ExecutablePath
▼ ProcCmdline
  ◦ ./nginx/nginx
▼ ProcFiles
  ◦ /var/log/nginx/error.log
  ◦ anon_inode:[eventpoll]
  ◦ /var/log/nginx/access.log
  ◦ /var/log/nginx/error.log
  ◦ /usr/share/nginx/html/index.html
▼ NetworkConnections
  ◦ tcp    LISTEN     0      128     *:http                  *:*                     users:(("nginx",pid=4649,fd=6))
  ◦ tcp    CLOSE-WAIT 1      0      192.168.1.11:http               192.168.1.9:40656               users:(("nginx",pid=4649,fd=8))
  ◦ tcp    LISTEN     0      128    :::http                 :::*                    users:(("nginx",pid=4649,fd=7))
▼ CrashSeverity
  ◦ EXPLOITABLE
  ◦ ReturnAv
  ◦ The target crashed on a return instruction, which likely indicates stack corruption.
▶ ProcMaps
▶ ProcEnviron
▶ ProcStatus
▶ CrashState
▼ Stacktrace
  ◦ #0 0x000000000043f6ca in ngx_http_read_discarded_request_body () from nginx
```

Анализ аварийных завершений в Астре 1.6 Орёл

Для тестирования CASR был проведён поиск ошибок по открытым источникам. Список тестируемых пакетов:

- *Ffmpeg*. Отказы в обслуживании: деление на ноль, разыменование нулевого указателя)
- *imagemagick*. Отказ в обслуживании: вызов функции abort().
- *popler-utils*. Отказы в обсуживании: деление на ноль, нарушение доступа к памяти.

Альтернативные сценарии применения CASR

- Включение в состав HIDS (хостовой системы обнаружения вторжений) с целью автоматической/автоматизированной реакции на попытки эксплуатации с внедрением кода.
- Включение в состав honeypot для обнаружения неизвестных ошибок с целью генерации сигнатур для IDS/IPS.

Реализация инструмента CASR

Инструмент разработан на языке Rust и основан на следующих компонентах с открытым исходным кодом:

- *libunwind* – библиотека для получения стека вызовов https://github.com/xcoldhandsx/libunwind-rs
- *Capstone* – библиотека для дизассемблирования https://github.com/capstone-rust
- *goblin* – библиотека разбора исполняемых файлов https://github.com/m4b/goblin

Поддерживает анализ coredump'ов для архитектур x84/64, armv7

Взаимодействие с открытом сообществом

- Разработка биндингов библиотеки *libunwind* на языке Rust https://github.com/xcoldhandsx/libunwind-rs.
- 2 Pull request (Добавление констант для структур. Исправление ошибок разбора core файлов.)
- 1 Issue (Неправильный тип возвращаемого значения из функции.)

Спасибо за внимание