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 скрытие 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.

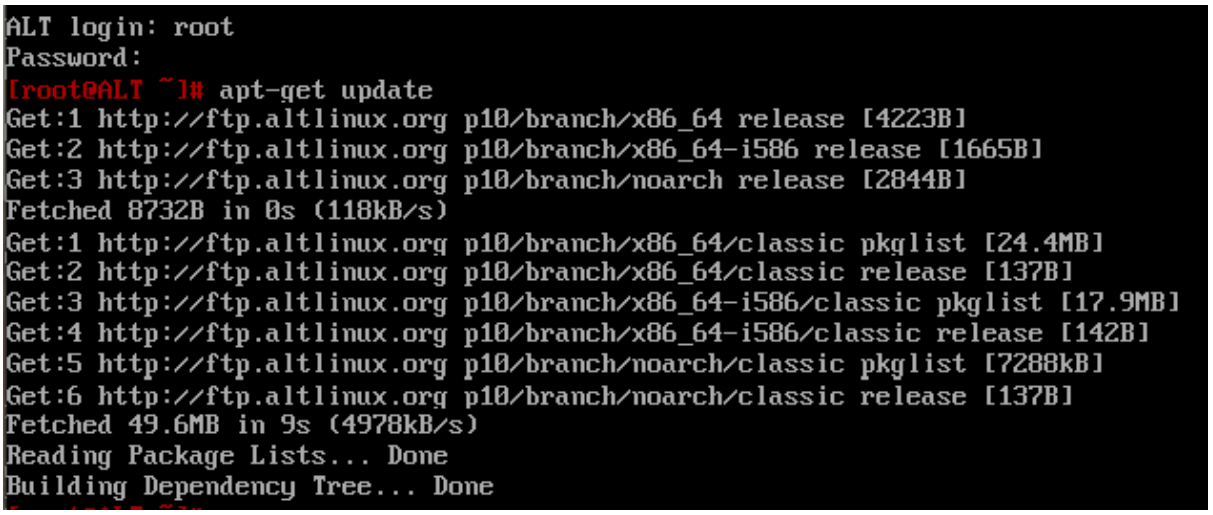

Рисунок 1. Обновление локальных списков пакетов.

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
```

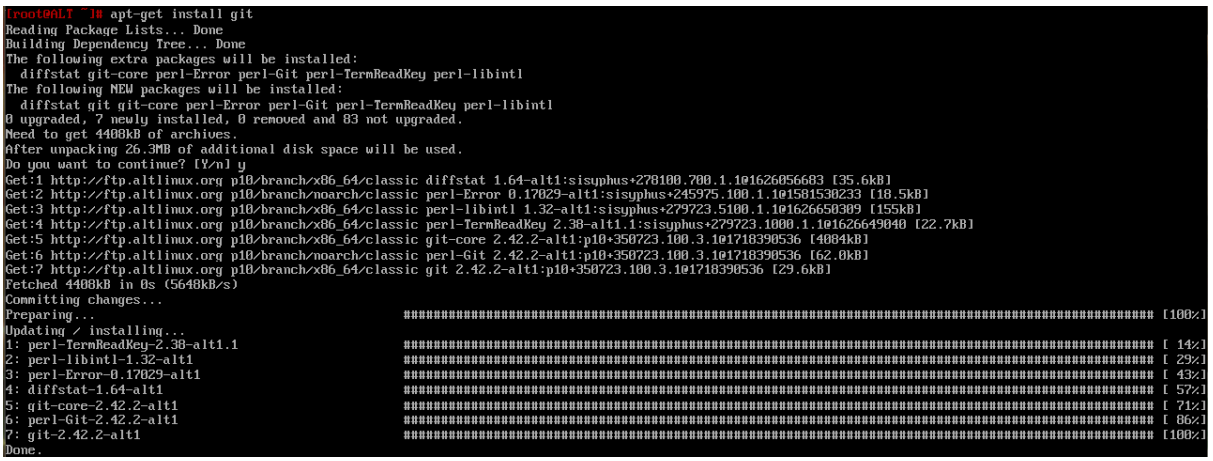

Рисунок 2. Установка утилиты Git.

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.

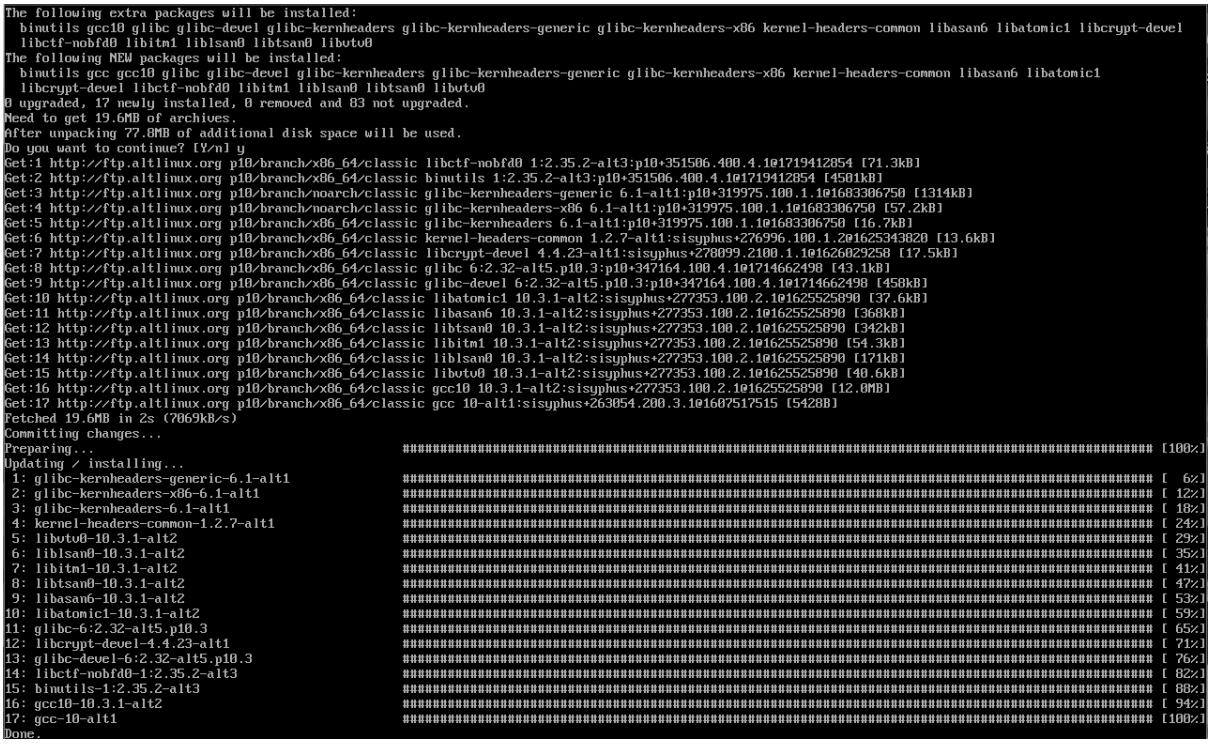

Рисунок 3. Установка утилиты GCC.

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.

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.

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.

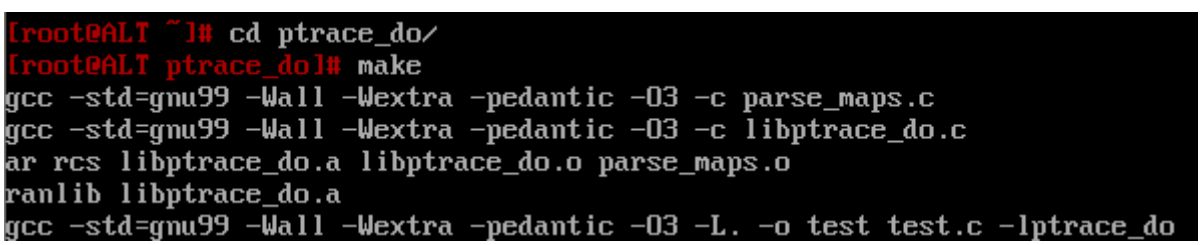

Рисунок 6. Компиляция проекта ptrace_do.

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 скрытия 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.