Модуль 23. Повышение привилегий в ОС Linux

Типы привилегий

В *Linux* для каждого файла есть три разрешения: *чтение*, *запись*, *исполнение*. Доступ к каждому отдельному файлу обозначается для его создателя, для группы и для прочих. Доступ каждой категории пользователей задаётся командой `chmod` и цифрой от 0-7. Также есть специальные разрешения:

- *SUID* — запускаем с правами конкретного пользователя,
- *SGID* — запускаем с правами конкретной группы,
- *Sticky bit* — запрещает удалять файлы, если пользователь не является их создателем.

Разница пользователя и root пользователя

root — это имя пользователя или учётная запись, которая по умолчанию имеет доступ ко всем командам и файлам в *Linux* или другой *Unix*-подобной операционной системе. У пользователя с *root* правами нет никаких серьезных ограничений, следовательно никаких механизмов ошибки от неосторожной ошибки, что может привести к проблемам с системой.

Техники

SUID

Метод повышения привилегий, в котором мы находим уязвимую команду с *SUID* битом (например `find` с его флагом `-exec`) и в зависимости от команды различными способами добиваемся выполнения произвольного вредоносного кода. Обычно с целью получения шелла.
Команда для поиска приложения с *SUID* битом — `find / -perm -u=s -type f 2>/dev/null`.

CRON

Cron — это утилита, которая позволяет пользователям *Linux* выполнять определённую задачу в заданное время и дату. Данная техника повышения привилегий предполагает нахождение уязвимой задачи, выполняемой из-под *root* (под критерии уязвимости попадают те, содержимое которых мы по недосмотру системного администратора можем редактировать), и изменение содержимого скрипта данной задачи для, например, получения шелла с правами рута.
Команда для просмотра содержимого *Cron* — `cat /etc/crontab`.

MISSING LIBRARIES

Данная техника повышения привилегий построена на неправильной работе системных администраторов с различными библиотеками. Хрестоматийный пример — ссылка на библиотеку, которой уже нет. В таком случае мы создаём новую библиотеку в директории, на которую указывала ссылка, а в библиотеке указываем код для, например, запуска шелла.
Из важного здесь стоит запомнить библиотеку *ld.so.conf.d*. Почему её? Там мы можем найти информацию о подгружаемых библиотеках.

PATH TRAVERSAL

Directory traversal (обход директории, также известный как *path traversal* — обход пути к файлу) — это уязвимость, позволяющая злоумышленнику читать произвольные файлы на сервере, на котором запущено приложение. Это может привести к компрометации информации и повышению привилегий.

Стоит обращать внимание, когда путь к файлу передаётся в открытом виде, примерно вот так:

```
<img src="/loadImage?filename= 218.png">
```

В таком случае с помощью *Burp suite* можно попробовать перехватить запрос на получение файла и изменить его, добавив, например, путь до файла — *../../etc/passwd*.

INTEGER OVERFLOW

Integer overflow (целочисленное переполнение), также известное как *wraparound* (зацикливание), происходит, когда арифметическая операция выводит числовое значение, которое выходит за пределы выделенного пространства памяти или выходит за пределы диапазона заданного значения целого числа.
Это приводит в том числе к переполнению буфера, повышению привилегий злоумышленником или выполнению вредоносного кода.
Buffer overflow происходит в связи с отсутствием проверки пользовательского ввода и/или его экранирования. Целочисленное переполнение может привести к возникновению переполнения буфера, что в свою очередь позволяет злоумышленнику получить оболочку и повысить свои привилегии.

DIRTY COW

Dirty COW — уязвимость в ядре *Linux*. Данная уязвимость позволяет вносить изменения в содержимое файлов, доступных только для чтения. Этот эксплойт использует *race condition* в процессе обработки функции копирования при записи (*COW*) отображений памяти. Пример использования включает перезапись *UID* пользователя в */ etc / passwd* для получения привилегий *root*.

Как работает сам эксплойт? Сначала мы создаём частную копию (отображение) файла, доступного только для чтения. Во-вторых, пишем в личную копию. Поскольку мы впервые пишем в личную копию, функция *COW* имеет место. Проблема заключается в том, что эта запись состоит из двух неатомарных действий: найти физический адрес и написать на физический адрес.
Это означает, что мы можем попасть прямо посередине (через другой поток) и сказать ядру, чтобы оно выбросило нашу частную копию, используя *madvise*. Это отбрасывание частной копии приводит к тому, что ядро случайно записывает исходный файл, доступный только для чтения.

PROCESS INJECTION

Process injection — это техника обхода защиты, часто применяемая в рамках вредоносных программ, она влечёт за собой запуск специального кода в адресном пространстве другого процесса. Внедрение процесса улучшает скрытность, а некоторые методы также обеспечивают постоянство.
Несколько техник, которые используют данную уязвимость, используют *LD_PRELOAD*, *LD_LIBRARY_PATH* и */proc/[pid]/mem*.

VDSO HIJACKING

VDSO (*Virtual dynamic shared object*) — это библиотека, которая отображается ядром в адресное пространство всех приложений пользовательского пространства. Вызывается из библиотеки *C* и служит для работы с несколькими системными вызовами.
Как *VDSO* может использоваться для повышения привилегий? Путём перехвата *VDSO* можно внедрить вредоносный код в адресное пространство процесса и таким образом разобраться с защитой механизмами, что может привести в том числе к повышению привилегий злоумышленником.

Перехват *VDSO* включает перенаправление вызовов к динамически подключаемым разделяемым библиотекам. Системный вызов *Ptrace*, являющийся одним из способов защиты памяти, может препятствовать записи кода в процесс. Нарушитель в таком случае может захватить заглушки кода интерфейса системных вызовов, сопоставленные с процессом из общего объекта *VDSO*, для выполнения сопоставления и открытия *VDSO*. В дальнейшем же с помощью перенаправления потока выполнения можно вызвать этот код.