Надежность в MINIX 3

Одна из главных целей MINIX 3 — обеспечение надежности. Ниже мы обсудим некоторые из наиболее важных принципов, повышающих надежность MINIX 3. Эти принципы также повышают безопасность, поскольку большинство уязвимостей в системе безопасности возникают из-за того, что злоумышленники используют ошибки в коде, поэтому повышение надежности также улучшит безопасность. Некоторые из обсуждаемых идей уже реализованы в текущей версии, но некоторые запланированы на следующий релиз. Поскольку это исследовательский проект, мы часто вносим изменения по мере того, как находим новые способы повышения надежности.

Уменьшить размер ядра

Монолитные операционные системы (например, Windows, Linux, BSD) содержат миллионы строк кода ядра. Невозможно сделать корректно реализованным такое количество кода. В отличие от них, MINIX 3 содержит около 4000 строк исполняемого кода ядра. Мы считаем, что этот код в конечном итоге можно сделать практически безошибочным.

Запереть насекомых в клетке

В монолитных операционных системах драйверы устройств находятся в ядре. Это означает, что при установке нового периферийного устройства в ядро вставляется неизвестный, ненадежный код. Одна-единственная некорректная строка кода в драйвере может привести к сбою системы. Такая конструкция принципиально ошибочна. В MINIX 3 каждый драйвер устройства представляет собой отдельный процесс пользовательского режима. Драйверы не могут выполнять привилегированные инструкции, изменять таблицы страниц, осуществлять ввод-вывод или записывать данные в абсолютную память. Они должны обращаться к ядру для получения доступа к этим службам, и ядро проверяет каждый вызов на наличие необходимых полномочий.

Ограничьте доступ драйверов к памяти.

В монолитных операционных системах драйвер может записывать данные в любое слово памяти и, таким образом, случайно повредить пользовательские программы. В MINIX 3, когда пользователь ожидает данных, например, из файловой системы, он создает дескриптор, указывающий, кто имеет доступ и по каким адресам. Затем он передает индекс этого дескриптора файловой системе, которая может передать его драйверу. После этого файловая система или драйвер запрашивают у ядра запись через дескриптор, что делает невозможным запись по адресам за пределами буфера.

Пережить плохие советы

Разыменование некорректного указателя внутри драйвера приведет к сбою процесса драйвера, но не окажет влияния на систему в целом. Сервер восстановления автоматически перезапустит поврежденный драйвер. Для некоторых драйверов (например, дискового и сетевого) восстановление происходит незаметно для пользовательских процессов. Для других (например, аудио и принтерного) пользователь может это заметить. В монолитных системах разыменование некорректного указателя в драйвере (ядра) обычно приводит к сбою системы.

Укротить бесконечные циклы

Если драйвер попадает в бесконечный цикл, планировщик постепенно понижает его приоритет, пока он не перейдет в состояние ожидания. В конце концов, сервер перерождения обнаружит, что он не отвечает на запросы состояния, поэтому он завершит работу и перезапустит зацикливающийся драйвер. В монолитной системе зацикливающийся драйвер приводит к зависанию системы.

Ограничьте ущерб от переполнения буфера.

MINIX 3 использует сообщения фиксированной длины для внутренней связи, что исключает некоторые проблемы, связанные с переполнением буфера и управлением буфером. Кроме того, многие эксплойты работают за счет переполнения буфера, чтобы обманом заставить программу вернуться из вызова функции, используя перезаписанный адрес возврата в стеке, указывающий на переполненный буфер. В MINIX 3 эта атака не работает, поскольку пространство инструкций и данных разделено, и может быть выполнен только код из (только для чтения) пространства инструкций.

Ограничить доступ к функциям ядра.

Драйверы устройств получают доступ к службам ядра (например, копируют данные в адресное пространство пользователей) путем выполнения вызовов ядра. Ядро MINIX 3 имеет битовую карту для каждого драйвера, указывающую, какие вызовы ему разрешено выполнять. В монолитных системах каждый драйвер может вызывать любую функцию ядра, независимо от наличия разрешения.

Ограничить доступ к портам ввода-вывода

Ядро также поддерживает таблицу, указывающую, к каким портам ввода-вывода может обращаться каждый драйвер. В результате драйвер может обращаться только к своим собственным портам ввода-вывода. В монолитных системах неисправный драйвер может получить доступ к портам ввода-вывода, принадлежащим другому устройству.

Ограничить взаимодействие с компонентами операционной системы.

Не каждому драйверу и серверу необходимо взаимодействовать с каждым другим драйвером и сервером. Соответственно, битовая карта для каждого процесса определяет, в какие пункты назначения может отправлять данные каждый процесс.

Реинкарнация умерших или больных водителей

Специальный процесс, называемый сервером реинкарнации, периодически отправляет пинг каждому драйверу устройства. Если драйвер выходит из строя или перестает корректно отвечать на пинги, сервер реинкарнации автоматически заменяет его новой копией. Обнаружение и замена неработающих драйверов происходит автоматически, без каких-либо действий со стороны пользователя. В настоящее время эта функция не работает для дисковых драйверов, но в следующем релизе система сможет восстанавливать даже дисковые драйверы, которые будут скопированы в оперативную память. Восстановление драйверов не влияет на запущенные процессы.

Интегрируйте прерывания и сообщения.

При возникновении прерывания оно на низком уровне преобразуется в уведомление, отправляемое соответствующему драйверу. Если драйвер ожидает сообщения, он получает прерывание немедленно; в противном случае он получает уведомление при следующем выполнении команды RECEIVE для получения сообщения. Эта схема исключает вложенные прерывания и упрощает программирование драйверов.