эпохе бурного развития – компьютер с динамической архитектурой

в последние годы техника совершила огромный скачок вперед, но это почти никак не отразилось на архитектуре компьютеров

текст В. А. Конявский, доктор технических наук

Наиболее заметное здесь изменение – опережающий рост решений на базе гарвардской архитектуры. Если несколько лет назад объемы продаваемых процессоров типов x86 и ARM соотносились как 80:20, то уже сегодня 50:50, и эта тенденция усиливается[1]. В то же время доказано, что именно архитектура универсальных компьютеров, которые моделируют машину Тьюринга (а это все современные компьютеры), является предпосылкой эффективности хакерских атак. Сложились условия для размышлений над усовершенствованием архитектуры.

При разработке компьютера главное понять, какая часть функций должна быть реализована аппаратно, а какая – программно. Правильный выбор этого соотношения позволил ПЭВМ с архитектурой фон Неймана многие годы занимать лидирующее положение. Однако сейчас практически достигнут предел эффективности данного технического решения, развитие по экстенсивному пути (увеличение производительности, памяти, снижение габаритов и т.п.) уже не удовлетворяет потребности общества. Драйвером развития может стать компьютер, лишенный тех уязвимостей, которые тормозят развитие информационного взаимодействия. При создании нового компьютера в аппаратную часть нужно включать то, что снижает стоимость, редко изменяется, расширяет возможности и используется постоянно. Кроме этого, сейчас совсем не выглядит экзотической мысль о том, что в процессе работы структура компьютера может динамически изменяться. Например, структура вначале может быть такой, как конечный автомат, а потом, на следующем этапе, стать «универсальным исполнителем» по Тьюрингу.

Отличительной особенностью архитектуры фон Неймана является то, что команды и данные не разделяются, они передаются по единому общему каналу. Гарвардская архитектура предполагает наличие разных каналов для команд и данных. Такая схема взаимодействия требует более сложной организации процессора, но обеспечивает большее быстродействие, так как потоки команд и данных становятся не последовательными, а параллельными, независимыми.

Однако и в случае компьютера фоннеймановского типа, и в случае компьютера с гарвардской архитектурой, организация потоков команд и данных таковы, что архитектурная уязвимость присуща каждому из них. Гибкость, универсальность и в одном, и в другом случае обеспечивается возможностью изменения последовательности команд и данных – независимо от того, в одной памяти они лежат или разделены. В свою очередь, возможность изменения последовательности команд и данных создает и возможность для несанкционированного вмешательства вредоносного программного обеспечения – это и есть основная архитектурная уязвимость.

На использовании этой уязвимости основаны практически все современные хакерские атаки, которые в основном сводятся к атаке на «перехват управления». Схема атаки обычно выглядит так:

- Внедряется и размещается в оперативной памяти вредоносное ПО (ВрПО).
- Внедряется и размещается в оперативной памяти вредоносный обработчик прерываний.
- Записывается в долговременную память ВрПО и обработчик прерываний.
- С помощью любого доступного механизма вызывается прерывание – например, с помощью DDoS-атаки.
- Внедренный ранее обработчик прерываний срабатывает и передает управление ВрПО.
- ВрПО выполняет свою функцию, например, реализует разрушающее программное воздействие (РПВ).

Здесь 1–3 – это шаги по подготовке атаки, 4 – инициирование атаки, 5 и 6 – собственно использование архитектурной уязвимости.

Чтобы обезвредить шаги 1 и 2, обычно используются антивирусные программы. Иногда это бывает полезным, но только иногда: невозможно с помощью антивирусных программ выявить все ВрПО. Более того – специалистам известны конструкции ВрПО, которые точно нельзя обнаружить. Можно даже сказать, что компьютерные вирусы и в целом ВрПО удается обнаружить только в силу их несовершенства. В общем случае всегда можно разработать такое ВрПО, которое не может быть обнаружено с помощью антивирусных программ сигнатурного поиска, эвристических анализаторов и поведенческих блокираторов.

[1] *Оценка дана академиком Б.А. Бабаяном (INTEL) в беседе с автором в июне 2015 года.*

В стандартных архитектурах выполнение шага 3 нельзя предотвратить. Можно только блокировать последствия. Блокирование последствий выполнения шага 3 выполняется при последующей загрузке с помощью механизмов контроля целостности – по сути, ревизоров, определяющих, есть ли изменения в составе данных. Иногда эта проверка выполняется с помощью тех же наборов антивирусных программ – но это слабое решение, так как проверка должна выполняться до загрузки ОС, а программы, в том числе и антивирусные, работают под управлением ОС.

Генерация события на шаге 4 частично блокируется с помощью специальных средств анализа трафика, устанавливаемых как в сети, так и на клиентских компьютерах. Важно, что пока нет средств, позволяющих гарантированно блокировать эту уязвимость.

Негативные последствия шагов 5 и 6 блокируются с помощью механизмов контроля запуска задач (процессов, потоков). Это очень эффективные механизмы, но реализующие их средства довольно дорогие и для их настройки нужно быть специалистом в компьютерных технологиях и информационной безопасности.

Поскольку некоторые из перечисленных функций безопасности должны выполняться до загрузки операционной системы, то реализовать их нельзя программно, а можно только с помощью сложного устройства, такого как СЗИ НСД «Аккорд», например.

Несмотря на большую распространенность, цена СЗИ НСД «Аккорд» довольно высока, и его настройка – дело для профессионалов. Конечно, это лучшее решение для корпоративных применений, но, видимо, оно слишком сложно для частного применения, например клиентами ДБО. Сложность его связана именно с фон-неймановской архитектурой защищаемого компьютера: нужно добавить неизменяемую память, разделить потоки команд и данных, исполнить контрольные процедуры в доверенной среде до запуска ОС и многое другое.

Однако в компьютерах, использующих гарвардскую архитектуру, потоки команд и данных уже разделены. Если это так, нельзя ли это обстоятельство использовать для упрощения и удешевления защитных механизмов? Нужно только сделать память неизменяемой (тогда нет необходимости использовать сложные механизмы контроля целостности программ и данных до старта ОС), а контрольные процедуры в этом случае можно исполнять под управлением проверенной и неизменяемой ОС.

Эти функции легко реализовать, если обеспечить движение команд и данных только в одном направлении – из памяти в процессор.

Очевидно, что такая архитектура обеспечит неизменность ОС, программ и данных.

Если вернуться при этом к схеме атаки, описанной выше, то видно, что шаг 3 не может быть выполнен, поэтому и сама атака (шаги 5 и 6) тоже не исполнится. Такой компьютер приобретет значительный «вирусный иммунитет», так как вредоносное ПО не будет фиксироваться на компьютере.

Недостатком при этом будет то, что придется дорабатывать практически все программное обеспечение, так как разработчики существующего ПО не ограничивают себя в использовании операций записи в память. Для работы практически всех программ необходима возможность записи.

Чтобы можно было использовать без доработок все ранее разработанное ПО, необходимо предложенную архитектуру дополнить блоками сеансовой памяти [2], в которой и будут исполняться программы.

Таким образом, архитектура компьютера будет отличаться на разных этапах – от этапа начальной загрузки к этапу функционирования.

Предложенная нами архитектура получила название «новая гарвардская архитектура». Она отличается тем, что в ней используется память, для которой установлен режим «только чтение». При загрузке команды и данные размещаются в сеансовой памяти, в которой и исполняются. Начальная загрузка и копирование кодов в сеансовую память могут выполняться как последовательно, так и параллельно – суть разделения этапов от этого не меняется.

Конечно, эта схема описана условно, и в реальных компьютерах все немного сложнее. Однако можно уверенно сказать, что владельцы таких компьютеров чувствуют себя намного защищеннее от атак хакеров.

Новая архитектура характеризуется динамической изменяемостью, что сохраняет эффективность применения, и при этом обеспечивает достаточную защищенность, неизменность операционной системы, «вирусный иммунитет», применение адаптированных стандартных ОС и всего программного обеспечения, написанного для них.

В архитектуре этого компьютера мы нарушили несколько принципов фон Неймана и в первую очередь – *размещение всей памяти компьютера в едином адресном пространстве* и *обеспечение условного перехода к любому участку кода в процессе выполнения программы*, несмотря на то, что команды выполняются последовательно. Действительно, память команд и память данных недоступны на запись, да и нумерацию ячеек этой памяти и сеансовой памяти нельзя считать последовательной. Ну и, естественно, условный переход возможен в пределах сеансовой памяти и невозможен в защищенной памяти. И что же мы получили взамен?

Основных преимуществ два:

- высокий уровень «вирусного иммунитета» и возможность создания и поддержки доверенной среды;
- возможность использовать все ранее наработанное программное обеспечение.

Важно то, что на основе описанной архитектуры можно создавать компьютеры для всех видов информационного взаимодействия, при которых доверенность и защищенность взаимодействия важна – от ДБО и защищенных «облаков» до «интернета вещей». NBJ

[2] *Название «сеансовая память» предложено А.В. Бабаниным при обсуждении этой архитектуры с автором, март 2015 года.*