УДК 004.056.53
DOI: 10.15827/0236-235X.140.598-608
Дата подачи статьи: 13.07.22, после доработки: 21.07.22
2022. Т. 35. № 4. С. 598–608

Разработка программных моделей доверенного универсального микропроцессора и микропроцессорной системы на его основе

С.И. Аряшев [1], *к.т.н., зам. директора по микроэлектронике и вычислительным системам, aserg@cs.niisi.ras.ru*
Н.А. Гревцев [1], *аспирант, научный сотрудник, ngrevcev@cs.niisi.ras.ru*
П.С. Зубковский [1], *зав. ОАВМ, zubkovsky@cs.niisi.ras.ru*
П.А. Чибисов [1], *к.т.н., старший научный сотрудник, chibisov@cs.niisi.ras.ru*
А.С. Кулешов [1], *старший разработчик, rndfax@cs.niisi.ras.ru*
К.А. Петров [1], *к.т.н., зам. зав. ОРВС, petrovk@cs.niisi.ras.ru*

[1] *Федеральный научный центр Научно-исследовательский институт системных исследований РАН, г. Москва, 117218, Россия*

При разработке доверенного микропроцессора для цифровых систем управления (систем с критической миссией) требуется разработать программную модель (эмулятор) доверенного микропроцессора и эмулятор системы на его основе для утверждения архитектурной модели и изучения возможностей парирования угроз. Покомандный и поведенческий эмуляторы микропроцессора являются инструментами моделирования микропроцессорной архитектуры и системы в целом и играют фундаментальную роль в различных областях микроархитектурного проектирования.

В работе приведены критерии, необходимые для создания доверенных систем, разработаны покомандный эмулятор микроархитектуры доверенного микропроцессора (VMIPS), а также поведенческий эмулятор архитектуры микропроцессорной системы (QEMU) на основе доверенного микропроцессора для утверждения архитектурной модели и изучения возможностей парирования угроз.

Поскольку по отдельности эмуляторы QEMU и VMIPS в целом не являются доверенными системами из-за того, что в эмуляторе QEMU отсутствует поддержка виртуализации, а в эмуляторе VMIPS поддержка IOMMU, в работе предложено использовать связки обоих эмуляторов: эмулятор QEMU запускает эмулятор VMIPS для эмуляции процессорных ядер. Для создания полноценной доверенной среды эмулятор VMIPS был представлен в виде библиотеки с API по эмуляции процессорных ядер, которая динамически подгружается эмулятором QEMU.

На основе анализа трасс подтверждаются те или иные ситуации угроз и их парирование. Для апробации использовано ПО, проверяющее функции эмулятора по обеспечению выполнения критериев доверенности системы путем парирования угроз из банка данных угроз безопасности информации ФСТЭК, а также произведен запуск демонстрационной задачи в виртуальной среде на виртуальном программируемом логическом контроллере с доверенным микропроцессором с применением SCADA для мониторинга и контроля.

Использование виртуального ПЛК с виртуальной средой позволяет проводить тестирование и отладку, исследования безопасности, строить модели существующих и будущих узлов, отрабатывать различные сценарии, получать полную информацию о ходе работы. Предварительное тестирование в виртуальной среде также позволяет снизить риски ввода в эксплуатацию и отработать различные модели угроз и их парирования до разработки микропроцессора. На основе результатов выполненной работы будет произведена разработка MIPS-подобного доверенного микропроцессора для цифровых систем управления СКМ.

Ключевые слова: *покомандный эмулятор, поведенческий эмулятор, программируемые логические контроллеры, виртуализация, MIPS, QEMU.*

Обеспечение информационной безопасности *автоматических систем управления производственными и технологическими процессами* (АСУ ТП) приобретает все большую актуальность. Это особенно важно в отношении систем с критической миссией и прежде всего в таких отраслях, как атомная и тепловая энергетика, железнодорожный и воздушный транспорт, химические производства, где удачно проведенная кибератака может иметь разрушительные последствия. Ключом к решению проблемы является применение доверенных отече-

ственных изделий электроники и сертифицированных программных платформ, специально созданных для ответственных применений.

Существующие в настоящее время сетевые вычислительные платформы не в состоянии в полной мере удовлетворить многочисленные требования в отношении безопасности обрабатываемых данных, предъявляемые со стороны возможных участников информационного взаимодействия: компаний-владельцев инфраструктуры, конечных пользователей, контент-провайдеров и особенно для систем с критической миссией предприятий атомной и тепловой энергетики.

Поскольку современные системы становятся все более сложными, открытыми и взаимосвязанными, традиционные технологии безопасности больше не могут соответствовать требованиям безопасности таких архитектур. Это объясняет тенденцию к интеграции концепций доверенных вычислений в различные системы, такие как встроенные системы АСУ ТП.

На современных промышленных предприятиях технологические задачи автоматизируются при помощи *программируемых логических контроллеров* (ПЛК) – микропроцессорных систем специального назначения с проблемно-ориентированным ПО для реализации замкнутых систем автоматического управления [1]. Благодаря универсальной структуре и гибкости ПЛК заменили собой ранее применяемые блоки релейной автоматики и устройства жесткой логики на интегральных микросхемах.

В отличие от блока релейной автоматики, логика которого определяется непосредственными физическими соединениями различных физических элементов, алгоритм логического управления в ПЛК задается программой. Это дает возможность создавать и изменять алгоритмы управления в ПЛК без прямого физического взаимодействия с ним.

Разработка ПО для ПЛК подчиняется стандартному жизненному циклу разработки и включает в себя итерационный процесс отладки и тестирования. Для программы ПЛК, основная функция которого заключается во взаимодействии с внешней средой (физическими узлами), выделяют четыре варианта сценария тестирования и отладки:

– на реальном ПЛК и в реальной среде;
– на виртуальном ПЛК, но в реальной среде;
– на реальном ПЛК, но в виртуальной среде;
– на виртуальном ПЛК и в виртуальной среде.

Задействование реальной среды при тестировании и отладке программы ПЛК подразумевает высокую цену и риски: от стоимости разработки тестовых узлов до рисков повреждения реальных узлов на предприятиях. Перед вводом ПЛК в эксплуатацию задействование реальной среды неизбежно, поэтому для снижения рисков предпочтительно предварительно проводить тестирование и отладку с использованием виртуальной среды. Виртуальная среда в связке с виртуальным ПЛК дает возможность получить полностью программное решение по тестированию и отладке реальных технологических задач.

Использование виртуального ПЛК с виртуальной средой позволяет проводить тестирование и отладку новых ПЛК, исследования безопасности, строить модели существующих и будущих узлов, отрабатывать различные сценарии функционирования, получая при этом полную информацию о ходе работы. Предварительное тестирование в виртуальной среде также позволяет снизить риски при вводе в эксплуатацию реального ПЛК, отработать модели угроз и их парирование до разработки архитектурной модели микропроцессора. В качестве виртуального ПЛК может быть применена связка покомандного и поведенческого эмуляторов микропроцессора, которые являются инструментами моделирования микропроцессорной архитектуры и системы, реализованной на ней, а в целом играют фундаментальную роль в различных областях микроархитектурного проектирования [2]. Эмуляторы используются в качестве эталонной модели для функциональной верификации и для оценки влияния новых идей, вносимых разработчиками на микроархитектурном уровне, на производительность системы в целом, а также для понимания поведения пользовательских программ и выявления аппаратных элементов, ограничивающих эффективность системы. При разработке микропроцессора (или микропроцессорной системы) все научные подходы и методы необходимо апробировать в сборе перед реализацией в микросхеме. На настоящий момент такие инструменты отсутствуют.

В данной работе представлен метод, в отличие от известных решений позволяющий анализировать эффективность проектируемой системы защиты информации с применением эмуляторов ядра и микропроцессорной системы, а также обеспечивать контроль степени доверия на заданном уровне на раннем этапе

жизненного цикла разработки изделия. Практическая значимость представленного метода заключается в том, что моделирование задач на эмуляторах разрабатываемой архитектуры в отличие от стандартных решений, моделирующих класс архитектур, позволит проводить отладку и моделировать угрозы применительно к конкретному изделию.

В данном исследовании доработан для соответствия критериям доверенного микропроцессора покомандный эмулятор VMIPS разработки НИИСИ РАН и продемонстрировано использование связки поведенческого эмулятора QEMU и покомандного эмулятора VMIPS. Связка эмуляторов выступает в качестве виртуального ПЛК, для которого была поставлена тестовая задача, разработано ПО с использованием специализированного языка программирования для ПЛК из состава спецификации МЭК 61131-3. Данное ПО запущено в гостевом режиме гипервизором, и произведена отработка задачи с использованием SCADA ScadaLTS, где для связи с ПЛК использован промышленный протокол Modbus.

Одним из результатов работы является апробированный на эмуляторе перечень решений, которые необходимо внести в RTL-модель разрабатываемого микропроцессора для обеспечения соответствия критериям доверенности. Разработанный и апробированный инструмент позволяет подтвердить корректность применимости существующих научных методов до начала разработки RTL-модели микропроцессора. Иными словами, этот инструмент позволяет отрабатывать системные решения и методы для создания доверенных микропроцессорных систем, чтобы применять их в актуальных средствах вычислительной техники.

Использование виртуализации для создания доверенной системы

С развитием и распространением вычислительных распределенных систем, а также с объединением их в общие сети появилась потребность в доверенных системах, способных обеспечить определенные политики безопасности, включающие в себя механизмы изоляции, защиты данных, криптографии, защиты ПО [3]. Доверенная система должна иметь механизмы проверки целостности системы, создания доверительной цепочки загрузки, иметь защищенное хранилище, а также сервисные функции по генерации криптографических ключей и шифрованию. Все это должно быть защищено от несанкционированного влияния. Одним из способов организации защищенной среды с разделением уровней доступа является виртуализация [4].

Виртуализация вводит дополнительный уровень разграничения привилегий в виде гипервизора. Гипервизор обслуживает гостевые системы, назначает аппаратные ресурсы, контролирует к ним доступ, обрабатывает различные ситуации, возникшие в гостевых системах [5].

К примеру, виртуализация на архитектуре MIPS может быть осуществлена без специальной аппаратной поддержки со стороны процессора с помощью классической методики trap-and-emulate [6], где гостевая система запускается в низкопривилегированном режиме, а любая привилегированная инструкция, которая, например, изменяет режим работы процессора или делает что-либо с кэшами или TLB, выполненная в низкопривилегированном режиме, вызовет исключительную ситуацию в работе гостевой системы и произойдет переход на уровень гипервизора – в обработчик этого исключения. Гипервизору доступна вся необходимая информация о причинах возникновения исключения. Тем самым обработчик может выполнить требуемые действия по обработке исключительной ситуации от гостевой системы и вернуть управление в гостевую систему.

Передача управления от гостевой системы к гипервизору и обратно, а также обработка исключительных ситуаций гостевых систем являются дорогостоящими с точки зрения потраченных инструкций операциями, требующими сохранения и восстановления состояний процессора. Для уменьшения вынужденного влияния гипервизора на процесс работы гостевых систем в процессоры добавляют аппаратную поддержку виртуализации, предоставляющую возможности по выполнению значительной части различных привилегированных инструкций гостевой системой самостоятельно без вовлечения гипервизора.

Аппаратная поддержка виртуализации в архитектуре MIPS

Архитектура MIPS имеет расширение поддержки аппаратного ускорения виртуализации, описанное спецификацией Virtualization Module of the MIPS64 Architecture. Данная спецификация вводит понятия двух контекстов – корневого и гостевого. Корневым контекстом

является обычный стандартный контекст архитектуры MIPS. Для поддержки виртуализации вводится гостевой контекст, который представляет собой новый отдельный контекст со своим управляющим сопроцессором CP0 и TLB. Большинство привилегированных инструкций может быть выполнено в рамках гостевого контекста, и лишь малая часть вызывает исключительную ситуацию, требующую обработки в корневом контексте гипервизором.

Спецификация задает обязательное наличие двойной трансляции адресов (рис. 1), где гостевой виртуальный адрес преобразуется в гостевой физический адрес с помощью гостевого TLB, а затем гостевой физический адрес трактуется как корневой виртуальный адрес и преобразуется в корневой физический адрес с помощью корневого TLB. С получившимся корневым физическим адресом происходит обращение к физической шине.

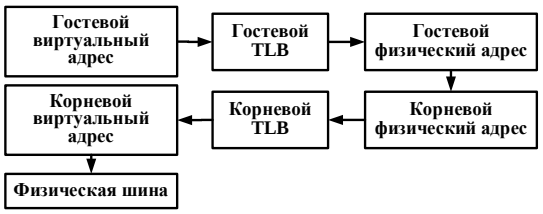

Рис. 1. Преобразование гостевого виртуального адреса в корневой физический

Fig. 1. Converting guest virtual address into a root physical one

Кэш-памяти в RTL-моделях микропроцессоров, разрабатываемых в НИИСИ, адресуются физическими адресами, поэтому в них будут содержаться все данные без привязки к конкретному контексту. В архитектуре MIPS с помощью специализированных инструкций можно манипулировать данными в кэшах. Например, освободить занимаемые строки кэша для соблюдения когерентности, запросить подгрузку данных, а также получить или записать произвольные данные в произвольные строки кэша. Спецификация на виртуализацию MIPS запрещает использование инструкций работы с кэшами в гостевом контексте, но вводит опциональную поддержку для разрешения части из них для аппаратного выполнения в гостевом контексте. Типы инструкций, позволяющие получить или модифицировать произвольным образом данные в кэшах, запрещены к выполнению в гостевом контексте безусловно. При этом типы инструкций, отвечающие за обеспечение когерентности кэшей и памяти, то есть те, что делают инвалидацию или запись строк кэшей в память, в гостевом контексте могут быть разрешены.

Корневой TLB разрешает прямой доступ гостевой системы к физической шине. С помощью корневого TLB гостевая система имеет прямой доступ к оперативной памяти и может иметь прямой доступ к периферийным устройствам. Периферийные устройства по различным событиям могут выставлять прерывания процессору. Корневой контекст получает прерывание и обрабатывает с помощью зарегистрированного обработчика. При работе гостевой системы с периферийными устройствами прерывания должны быть явным образом переброшены в гостевую систему в обработчике прерываний в гипервизоре. Спецификация MIPS на виртуализацию вводит опциональную поддержку аппаратной маршрутизации физических линий прерываний процессора напрямую в гостевой управляющий сопроцессор CP0, избавляя от дополнительной интервенции гипервизора.

Таким образом, аппаратная поддержка виртуализации в архитектуре MIPS уменьшает число исключительных ситуаций в гостевых системах при выполнении привилегированных инструкций и позволяет предоставить полный контроль над периферийным устройством, обеспечив доступ к нему по физической шине с помощью корневого TLB и обратную связь с гостевой системой с помощью механизма маршрутизации прерываний. Предоставляемые механизмом виртуализации необходимые аспекты по организации защищенной среды позволяют применять ее для создания доверенных систем.

Разработка прототипа покомандного эмулятора, реализующего доверенный микропроцессор

Эмулятор (эмулятор машины, или эмулятор микропроцессора) представляет собой программное решение, запускаемое на хостовой машине одной архитектуры, по выполнению программ этой же или другой процессорной архитектуры. Результаты работы программ, выполняемых в среде эмулятора, идентичны результатам работы такой же программы на аналогичной реальной аппаратуре. В эмуляторах допускаются упрощения в процессах получения результатов, что делает их более простыми в разработке и более быстрыми по сравнению с моделированием, например,

RTL-моделей. Эмуляторы используются для различных целей, для разработки и отладки как ПО, так и аппаратных блоков.

Архитектура эмулятора выстраивается в зависимости от его целей. Какие-то эмуляторы предназначены для быстрой эмуляции [7], в них сильно упрощается сам процесс эмуляции и основные усилия сосредотачиваются на скорости эмулирования инструкций. Так, например, из процесса эмуляции может быть отброшена эмуляция процессорных кэшей, различных буферов, протоколов когерентности, а процессы работы DMA могут быть выполнены моментально, без растягивания по времени, как это происходит на реальной аппаратуре. Другие эмуляторы предназначены для более точной эмуляции [8], и в них процессы эмуляции становятся все более похожими на реальные физические процессы в аппаратуре. Такие эмуляторы расширяют возможности по инспекции поведения программ с точки зрения глубинных процессов, что помогает в отладке программ и моделей RTL.

Эмулятор VMIPS как эмулятор процессорных ядер QEMU

Покомандный эмулятор VMIPS, основанный на свободно распространяемом открытом одноименном проекте, представляет собой инструмент для разработки и отладки RTL-моделей, микросхем и ПО. VMIPS эмулирует процессор с MIPS-совместимой архитектурой, память и необходимый набор периферийных устройств для возможности запуска различного ПО от бареметальных тестов (baremetal – платформа без операционной системы, которая позволяет ПО получать прямой доступ к аппаратной части) до полноценного дистрибутива GNU/Linux в эмулируемой среде. VMIPS предоставляет средства гибкой настройки эмуляции, инспектирования и отладки.

По мере увеличения требований к разрабатываемым микросхемам, повышению точности соответствия эмулятора и RTL-модели или необходимости проведения каких-либо исследований [2] в эмулятор VMIPS вносятся необходимые преобразования. Так, в изначальный эмулятор VMIPS были внесены следующие доработки: реализация 64-битной архитектуры, сопроцессора FPU, векторного сопроцессора, набора периферийных устройств типа контроллеров DMA, прерываний, UART, Ethernet. Были также добавлены эмуляция кэш-памятей, многоядерного процессора с протоколами когерентности, задержек при обращении к памяти и выполнении кода, когерентность кэш-памяти для DMA, системного контроллера, а также эмуляция жесткого диска.

Поскольку в процессе разработки RTL-модели эмулятор VMIPS используется в качестве поведенческой эталонной модели, разработка поддержки аппаратной виртуализации MIPS в эмуляторе VMIPS опирается на технологические возможности реализации виртуализации в аппаратуре. Использование упомянутой двойной трансляции адреса с помощью каскадирования двух TLB повлечет за собой значительные изменения в архитектуре RTL. Был разработан альтернативный механизм второй трансляции Memory Protection Unit (MPU, рис. 2).

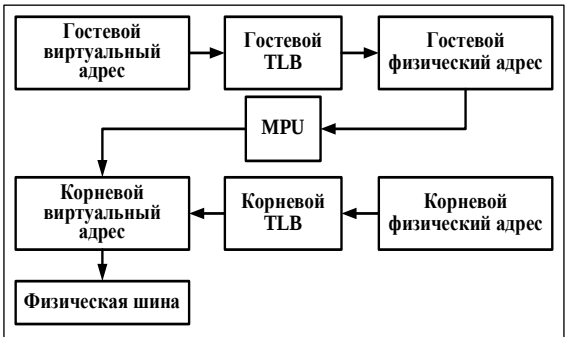

Рис. 2. Измененная схема преобразования адреса

Fig. 2. A changed address translation scheme

MPU имеет более простую структуру, чем корневой TLB, и специальные биты управления трансляцией, необходимые именно для трансляции гостевых физических адресов, которые отсутствуют в традиционном TLB. MPU работает постранично, преобразуя гостевой физический адрес в корневой.

MPU представляет собой таблицу из 8 строк. Каждая строка описывает окно доступа, которое содержит входящий гостевой физический адрес, исходящий корневой физический адрес и атрибуты окна, включающие разрешение работы окна, маску размера окна, политику преобразования атрибута кэширования и атрибут разрешения на запись. Настройка окон MPU производится через регистры корневого управляющего сопроцессора CP0 без возможности доступа к ним гостевым контекстом.

При каждом обращении из гостевого контекста к памяти определяется принадлежность гостевого физического адреса к одному из

8 окон. Если принадлежность обнаружена, то происходит преобразование старших битов гостевого физического адреса в корневой, заданный для данного окна. В противном случае генерируется исключение в корневом контексте для обработки ситуации в гипервизоре. Для минимизации отличия от спецификации на аппаратную виртуализацию MIPS исключения от MPU генерируются аналогичными исключениям корневого TLB в аналогичных ситуациях с выставлением необходимых кодов исключений и предоставлением необходимой информации в управляющих регистрах корневого сопроцессора CP0 для их обработки.

При реализации виртуализации в эмуляторе VMIPS были сделаны отступления от спецификации на аппаратную виртуализацию MIPS по части двойной трансляции гостевого адреса. Поэтому для работы модуля KVM (Kernel-based Virtual Machine) для архитектуры процессора, эмулируемой VMIPS, требуется добавление поддержки этой измененной реализации аппаратной виртуализации MIPS в модуль KVM. Отличие заключается в обработчике исключений TLB. Когда возникает исключительная ситуация во время второй трансляции в корневом TLB по причине отсутствия в нем нужной страницы, реализация KVM для MIPS осуществляет поиск требуемой страницы в гостевых таблицах трансляций и при нахождении прописывает ее в корневой TLB. Так как MPU генерирует аналогичные исключения, как и корневой TLB, требуются минимальные правки, которые отличаются только конечным этапом программирования таблицы трансляции.

Поведенческий эмулятор QEMU, разрабатываемый в НИИСИ и основанный на свободно распространяемом открытом одноименном проекте [7], представляет собой инструмент для разработки и отладки RTL-моделей, микросхем и ПО. QEMU эмулирует процессор, память и имеет богатый набор периферийных устройств для возможности запуска различного ПО от баремeтальных тестов до полноценного дистрибутива GNU/Linux в эмулируемой среде.

В эмуляторе QEMU уже была представлена поддержка архитектуры MIPS. Для поддержки архитектуры процессоров, разрабатываемых в НИИСИ, были проведены работы по добавлению поддержки векторных сопроцессоров, управляющего сопроцессора CP0 и эмуляции периферийных устройств, присутствующих в различных микросхемах разработки НИИСИ.

В эмуляторе QEMU эмуляция ядра процессора является отдельной независимой частью. В базовом коде эмулятора QEMU за эмуляцию ядра процессора могут отвечать интерпретатор, динамический транслятор или подсистема гипервизора типа KVM. Выбор ответственного за эмуляцию процессорного ядра происходит через опции командной строки эмулятора QEMU. Эмуляция ядра процессора представляет собой обособленный цикл, который должен вычитывать и исполнять инструкции. При возникновении определенных ситуаций этот цикл должен прерываться для взаимодействия с другими частями эмулятора QEMU. Для создания полноценной доверенной среды было решено в качестве эмуляции ядра процессора использовать эмулятор VMIPS. Для этого VMIPS был предоставлен в виде библиотеки с API по эмуляции процессорных ядер, которая динамически подгружается эмулятором QEMU. После загрузки библиотеки VMIPS устанавливается прослушивающий обработчик изменений системной области памяти. События добавления или удаления областей ОЗУ или ПЗУ передаются в библиотеку эмулятора VMIPS. Обращения эмулируемого процессора в области, не попадающие в зарегистрированные области ОЗУ или ПЗУ в эмуляторе VMIPS, возвращаются на обработку в эмулятор QEMU, где посредством уже API данного эмулятора происходит нужное обращение.

В процессе работы эмулятор QEMU может устанавливать или считывать значения различных регистров процессора, запрашивать прочистку кэшей из-за обращений периферийных устройств к памяти, управлять линиями прерывания процессора, запрашивать остановку эмуляции.

Схема работы эмуляторов QEMU и VMIPS представлена на рисунке 3.

Демонстрация парирования угрозы микропроцессора и микропроцессорной системы

Увеличение вычислительной мощности элементов промышленных систем приводит к тому, что АСУ ТП с большей вероятностью будет подвергаться как вредоносным атакам, так и нецелевому использованию злоумышленниками. Атаки на ПЛК включают изменение его логики, активацию новой программы, тестирование нового кода, загрузку нового рецепта процесса, вставку вспомогательной логики для отправки сообщений или активации какой-

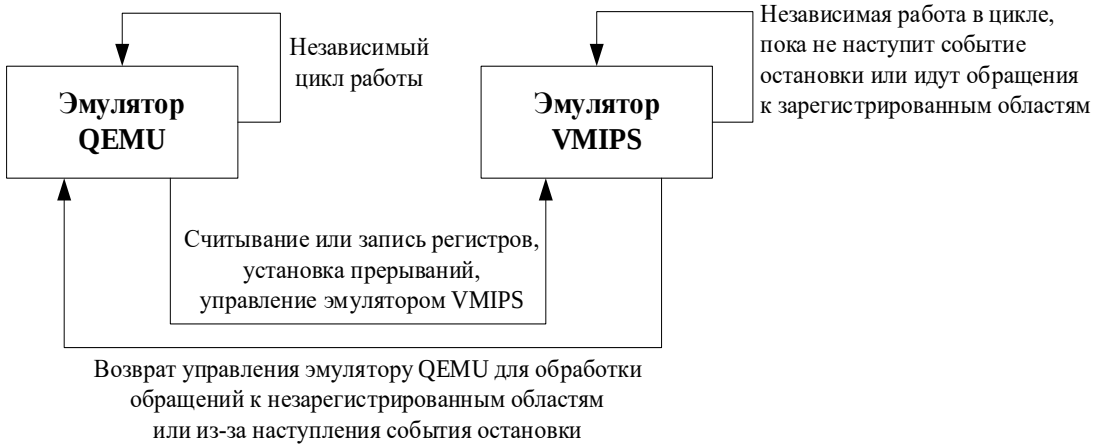

Рис. 3. Схема работы эмуляторов QEMU и VMIPS

Fig. 3. An operation scheme of QEMU and VMIPS emulators

либо функции. Для большинства ПЛК традиционные проверки криптографической целостности невозможны. Поскольку использование виртуального ПЛК позволяет отрабатывать различные сценарии, получать полную информацию о ходе работы, а также проводить исследования безопасности, он может быть использован для демонстрации парирования угроз.

Для демонстрации парирования угрозы будет представлена выдача сообщений исполнения эмуляторов QEMU и VMIPS. Так как по отдельности эмуляторы QEMU и VMIPS не представляют собой доверенную систему в целом из-за того, что в QEMU отсутствует поддержка виртуализации, а в VMIPS отсутствует поддержка IOMMU (Input/Output Memory Management Unit), для демонстрации будут использованы оба эмулятора, работающие в связке, где QEMU запускает VMIPS для эмуляции процессорных ядер. Наиболее информированные трассы может сгенерировать VMIPS, поэтому ему передаются опции для включения генерации трасс. На основе анализа трасс будут подтверждаться те или иные ситуации угрозы и их парирование. В данном случае не рассматриваются угрозы из-за ошибок, неправильной настройки или взлома ПО, из-за которых может реализоваться угроза. Рассматриваются частные ситуации, которые могут привести к реализации угрозы. Парирования первопричин угроз в данной работе не рассматриваются. Рассматриваются только парирования непреднамеренных результатов.

Трассы генерируются в текстовый файл. В трассе представлены адреса инструкции, дизассемблер инструкции (мнемокод языка ассемблера) и результаты работы инструкции. Для визуального прослеживания прогресса выполнения инструкций в трассах все инструкции пронумерованы. Трасса сопровождается отладочной выдачей о возникших исключительных ситуациях надписей типа Exception с пояснением типа исключения.

В трассе введены обозначения текущего контекста. Если действие выполняется в гостевом режиме, то строки трассы помечаются префиксом [VZ], если в корневом контексте – без [VZ]. Для демонстрации парирования угрозы микропроцессора и микропроцессорной системы были отобраны самые значимые угрозы из банка данных угроз безопасности информации ФСТЭК. В репозитории НИИСИ РАН (https://github.com/SRISA-RAS/EMULATOR) представлены трассы парирования следующих угроз:

– вызов в отказе обслуживания всей системы по причине зависания процессора;

– несанкционированный доступ к защищенным данным;

– манипуляции настройками процессора;

– запуск измененного или непроверенного ПО;

– несанкционированный доступ к данным посредством DMA;

– внедрение в канал связи.

Демонстрация работы тестовой задачи на эмуляторе доверенного микропроцессора

Определим, разработаем и запустим демонстрационную задачу ПЛК в виртуальной среде

на виртуальном ПЛК с доверенным микропроцессором с применением SCADA для мониторинга и контроля.

Сформулируем условие демонстрационной задачи. Пусть есть датчик температуры, световой индикатор и кнопка сброса. При превышении температурой заданного порогового значения загорается световой индикатор, который продолжает гореть, пока не будет сброшена ниже порогового значения температура и не нажата кнопка сброса. Необходимо реализовать данное управление на ПЛК. Мониторинг состояния температуры, светового индикатора и управление кнопкой сброса необходимо обеспечить с помощью SCADA.

В качестве виртуального ПЛК и виртуальной среды будут выступать эмуляторы QEMU и VMIPS разработки НИИСИ. В связке данные эмуляторы эмулируют доверенный микропроцессор с интерфейсами ввода-вывода и коммуникации. Внутри эмулируемого микропроцессора есть блок АЦП с функцией измерения температуры. Через управляющую консоль эмулятора QEMU можно изменять значения, регистрируемые датчиком температуры в виде напряжения. Из коммуникационных контроллеров для простоты были выбраны контроллер UART и протокол Modbus. В качестве среды разработано миниатюрное окружение, выполняющее простые функции по синхронизации входов и выходов и по коммуникации через UART. Среда выполнения запускается в гостевом режиме простым гипервизором.

В соответствии с МЭК 61131-3 существуют пять различных взаимозаменяемых языков программирования. Для демонстрации был выбран язык Structured Text. Обозначим вход %IW0 значением температуры, а вход %I0.0 состоянием нажатия кнопки сброса. Выход %Q0.0 обозначим как индикатор превышения заданного порогового значения температуры. Зададим пороговое значение числом 1 000, которое находится в пределах возможных значений эмулируемого контроллера АЦП. (Текст программы ПЛК на языке программирования Structured Text представлен в приложении https://github.com/SRISA-RAS/EMULATOR.)

Для запуска программы на языке программирования Structured Text на эмулируемом процессоре был использован компилятор MatIEC. Он представляет собой свободно распространяемый компилятор, который транслирует текстовые языки программирования из спецификации МЭК 61131-3 в язык программирования C. Получившийся код программы на языке программирования C представляет собой обособленную логическую часть программы, соответствующую шагу 2 из цикла сканирования. Инициализация, синхронизация входов и выходов и выполнение одной итерации цикла получившейся программы происходят в среде выполнения.

Среда выполнения создана на языке программирования C. Она производит инициализацию программы выполнения, АЦП и UART и запускает цикличное сканирование. При выполнении 1-го шага сканирования – считывания входов – создается образ состояний входов в оперативной памяти. Среда выполнения заносит в %IW0 считанное через АЦП значение температуры, которое также заносится в регистр ввода № 0 протокола Modbus. Вход %I0.0 синхронизируется с регистром флагов № 1 протокола Modbus. При выполнении 3-го шага сканирования – синхронизация выходов – выход %Q0.0 синхронизируется с регистром флагов № 0 протокола Modbus.

В качестве SCADA был выбран программный пакет ScadaLTS (http://scada-lts.org), представляющий собой web-сервер, доступ к которому осуществляется через интернет-браузер. ScadaLTS производит мониторинг источников данных и точек данных и управление ими. На периодической основе ScadaLTS производит опрос точек данных. В ScadaLTS был добавлен источник данных с протоколом Modbus TCP/IP и IP-адресом программного моста. Были добавлены три точки данных:

– мониторинга температуры;
– мониторинга световой индикации;
– управления (кнопка сброса).

Точка мониторинга температуры запрашивает температуру, полученную из АЦП, по протоколу Modbus из регистра ввода № 0, который также продублирован на вход %IW0.0 программы ПЛК. Точка мониторинга световой индикации запрашивает состояние выхода %Q0.0 программы ПЛК через регистр флагов № 0. Точка управления сброса запрашивает установку значения через регистр флагов № 1, который задает значение входу %I0.0 программы ПЛК.

Для каждой точки был создан свой графический элемент. Температура отображается в виде столбика шкалы в пределах от 0 до 1 000 и текстового табло. Данные по температуре поступают в виде измерений АЦП в пределах от 0 до 4 095 включительно. Световая индикация отображается в виде лампочки, кнопка сброса – в виде кнопки с текстом, которую можно

нажать и отжать. Текстовая индикация на кнопке показывает действие, которое произойдет, если на нее нажать. Первоначальные данные показывают температуру со значением 163 (рис. 4), что соответствует значению 0.1 регистрируемой датчиком температуры, лампочка не горит, кнопка сброса не нажата и показывается текст RESET.

Проведем тестовое воздействие на датчик температуры с помощью управляющей консоли эмулятора QEMU. Последовательно увеличим значение, регистрируемое датчиком температуры до 0.4, 0.6 и 0.7:

```
(qemu) qom-set /machine/soc/adc temp 0.4
(qemu) qom-set /machine/soc/adc temp 0.6
(qemu) qom-set /machine/soc/adc temp 0.7
```

Рассмотрим изменение графических элементов в ScadaLTS на рисунке 5.

Рис. 4. Графическое отображение точек данных в ScadaLTS

Fig. 4. Graphical display of data points in ScadaLTS

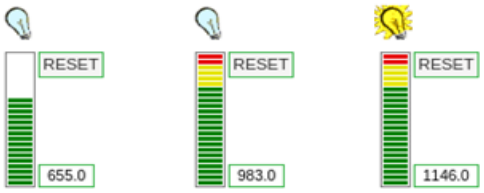

Рис. 5. Реакция графических элементов на изменение показателей от источника данных

Fig. 5. Changing graphical elements into changes in data source indicators

При увеличении температуры до 0.4 поднялась шкала температуры и изменилось соответствующее отображение на текстовом табло, показывающее значение 655. При установке температуры в 0.6 столбик шкалы поднялся до самого верха и градиентным переходом от зеленым к красным полоскам показывает приближение к критическому значению температуры. Табло показывает значение 983, что немного ниже порогового значения 1 000. После увеличения температуры до 0.7 табло показывает значение 1 146. Загорелась лампочка, показывающая превышение порогового значения температуры в 1 000.

Проверим сценарий погашения лампочки (рис. 6, слева направо).

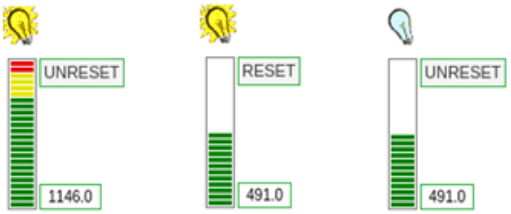

Рис. 6. Сценарий гашения лампочки

Fig. 6. A scenario of turning off a light bulb

Воздействие на кнопку сброса с текстом RESET производит нажатие кнопки и смену текста на UNRESET. Лампочка все еще горит, что соответствует условию задачи, которое гласит, что световая индикация должна погаснуть только при нижепороговом значении температуры и нажатой кнопке сброса. Снижение температуры до 0.3 через управляющую консоль эмулятора QEMU сбрасывает показатели температуры до 491, столбик температуры теперь находится в зеленой области, но лампочка продолжает гореть, так как условие ее погашения еще не выполнено полностью. После нажатия на кнопку сброса происходит гашение лампочки.

Заключение

Благодаря развитию ПЛК стало возможным создавать целые сети из ПЛК, централизованно управлять множеством узлов, проводить мониторинг и обслуживание. Специфика критических задач ПЛК и объединение их в сети поднимает вопрос безопасности. Для предотвращения несанкционированного доступа, который может повлечь за собой непредсказуемые последствия, в сетях с ПЛК, а также в самих ПЛК требуется использовать максимальные уровни защиты – доверенные микропроцессоры и сетевые технологии, способные обеспечить безопасные каналы связи. В статье показано, что использование виртуального ПЛК с виртуальной средой позволяет проводить исследования безопасности на ранней стадии проектирования архитектуры модели доверенного микропроцессора. Предварительное тестирование в виртуальной среде способствует снижению рисков ввода в эксплуатацию и отработке различных моделей угроз и их па-

рированию. В статье демонстрируются парирование угроз из списка ФСТЭК, а также пример работы тестовой задачи ПЛК с применением SCADA для мониторинга и контроля на связке двух эмуляторов – ядра микропроцессора и системы на его основе.

Литература

1. Yusuf S.A. Development of PLC and SCADA based integrated thermal control system with self/auto-tuning feature. Proc. CMD, 2018, pp. 1–6. DOI: 10.1109/CMD.2018.8535698.

2. Chibisov P., Grevtsev N., Kuleshov A., Zubkovsky P. Using architecture simulation tool for memory subsystem evaluation in multi-core systems. Proc. EWDTS, 2021, pp. 1–7. DOI: 10.1109/EWDTS52692.2021.9580989.

3. Sabt M., Achemlal M., Bouabdallah A. Trusted execution environment: What it is, and what it is not. Proc. IEEE Trustcom/BigDataSE/ISPA, 2015, vol. 1, pp. 57–64. DOI: 10.1109/Trustcom.2015.357.

4. Li W., Xia Y., Lu L., Chen H., Zang B. TEEv: Virtualizing trusted execution environments on mobile platforms. Proc. XV ACM SIGPLAN/SIGOPS Int. Conf. VEE, 2019, pp. 2–16. DOI: 10.1145/3313808.3313810.

5. Moratelli C., Johann S., Hessel F. Exploring embedded systems virtualization using MIPS virtualization module. Proc. ACM Int. Conf. CF, 2016 pp. 214–221. DOI: 10.1145/2903150.2903179.

6. Popek G.J., Goldberg R.P. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 1974, vol. 17, no. 7, pp. 412–421. DOI: 10.1145/361011.361073.

7. Bellard F. QEMU, a fast and portable dynamic translator. Proc. USENIX ATEC, 2005, pp. 41–46.

8. Abudaqa A.A., Al-Kharoubi T.M., Mudawar M.F., Kobilica A. Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with Gem5 simulator. Proc. V ICEEE, 2018, pp. 317–322. DOI: 10.1109/ICEEE2.2018.8391354.

Software & Systems
DOI: 10.15827/0236-235X.140.598-608

Received 13.07.22, Revised 21.07.22
2022, vol. 35, no. 4, pp. 598–608

Development of trusted microprocessor software models and a microprocessor system

S.I. Aryashev [1], *Ph.D. (Engineering), Principal Director of Microelectronics and Computing Systems, aserg@cs.niisi.ras.ru*
N.A. Grevtsev [1], *Postgraduate Student, Research Associate, ngrevcev@cs.niisi.ras.ru*
P.S. Zubkovsky [1], *Head of the Architecture Department of High-performance Microprocessors Systems, zubkovsky@cs.niisi.ras.ru*
P.A. Chibisov [1], *Ph.D. (Engineering), Senior Researcher, chibisov@cs.niisi.ras.ru*
A.S. Kuleshov [1], *Senior Developer, rndfax@cs.niisi.ras.ru*
K.A. Petrov [1], *Ph.D. (Engineering), Deputy Head of Department OAVM, petrovk@cs.niisi.ras.ru*

[1] *Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA RAS), Moscow, 117218, Russian Federation*

Abstract. When developing a trusted microprocessor for digital SCM control systems (systems with a critical mission), it is necessary to develop a software model (emulator) of a trusted microprocessor and a system emulator based on it to approve the architectural model and to study the possibilities of parrying threats. Instruction-based and behavioral microprocessor emulators are tools for modeling the microprocessor architecture and the system as a whole. They play a fundamental role in various areas of microarchitecture design. Emulators are used as a reference model for functional verification and for assessing the contribution of new ideas introduced by developers at the microarchitecture level to the performance of the system as a whole, as well as for understanding the behavior of user programs and identifying hardware elements that limit the system effectiveness.

The paper presents the criteria necessary for creating trusted systems, a developed instruction-based emulator of the trusted microprocessor microarchitecture (vmips), as well as a behavioral emulator of the microprocessor system architecture (QEMU) based on a trusted microprocessor to approve the architectural model and study the possibilities of parrying threats.

There is a demonstration of software that tests the functions of the emulator to ensure the fulfillment of the system's trusted execution environment criteria by parrying threats from the FSTEC information security threats data bank. The paper also describes launching a demonstration task in a virtual environment on a virtual programmable logic controller with a trusted microprocessor using SCADA for monitoring and control. Using a virtual PLC with a virtual environment allows testing and debugging, conducting security studies, building models of existing and future nodes, working out various scenarios, and getting complete information about the work progress. Preliminary testing in a virtual environment also allows reducing the risks of commissioning and working out various threat models and their parrying before developing a microprocessor. Based on the results of the work performed, the development of a trusted microprocessor with a MIPS-like architecture for digital control systems of the SCM is planned in the future.

Keywords: trusted execution environment, PLC, ISS, behavioral emulator, virtualization, MIPS, QEMU.

References

1. Yusuf S.A. Development of PLC and SCADA based integrated thermal control system with self/auto-tuning feature. *Proc. CMD*, 2018, pp. 1–6. DOI: 10.1109/CMD.2018.8535698.

2. Chibisov P., Grevtsev N., Kuleshov A., Zubkovsky P. Using architecture simulation tool for memory subsystem evaluation in multi-core systems. *Proc. EWDTS*, 2021, pp. 1–7. DOI: 10.1109/EWDTS52692.2021.9580989.

3. Sabt M., Achemlal M., Bouabdallah A. Trusted execution environment: What it is, and what it is not. *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2015, vol. 1, pp. 57–64. DOI: 10.1109/Trustcom.2015.357.

4. Li W., Xia Y., Lu L., Chen H., Zang B. TEEv: Virtualizing trusted execution environments on mobile platforms. *Proc. XV ACM SIGPLAN/SIGOPS Int. Conf. VEE*, 2019, pp. 2–16. DOI: 10.1145/3313808.3313810.

5. Moratelli C., Johann S., Hessel F. Exploring embedded systems virtualization using MIPS virtualization module. *Proc. ACM Int. Conf. CF*, 2016 pp. 214–221. DOI: 10.1145/2903150.2903179.

6. Popek G.J., Goldberg R.P. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 1974, vol. 17, no. 7, pp. 412–421. DOI: 10.1145/361011.361073.

7. Bellard F. QEMU, a fast and portable dynamic translator. *Proc. USENIX ATEC*, 2005, pp. 41–46.

8. Abudaqa A.A., Al-Kharoubi T.M., Mudawar M.F., Kobilica A. Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with Gem5 simulator. *Proc. V ICEEE*, 2018, pp. 317–322. DOI: 10.1109/ICEEE2.2018.8391354.

Для цитирования

Аряшев С.И., Гревцев Н.А., Зубковский П.С., Чибисов П.А., Кулешов А.С., Петров К.А. Разработка программных моделей доверенного универсального микропроцессора и микропроцессорной системы на его основе // Программные продукты и системы. 2022. Т. 35. № 4. С. 598–608. DOI: 10.15827/0236-235X.140.598-608.

For citation

Aryashev S.I., Grevtsev N.A., Zubkovsky P.S., Chibisov P.A., Kuleshov A.S., Petrov K.A. Development of trusted microprocessor software models and a microprocessor system. *Software & Systems*, 2022, vol. 35, no. 4, pp. 598–608 (in Russ.). DOI: 10.15827/0236-235X.140.598-608.