ДОВЕРЕННАЯ СРЕДА

УДК 004.056.53
DOI: 10.52190/2073-2600_2021_4_13

Технология SMM и ее применение в компьютерной безопасности

М. В. Пахомов
Московский физико-технический институт (национальный исследовательский университет), г. Долгопрудный, Московская обл., Россия

В статье рассмотрен один из наиболее привилегированных режимов выполнения кода на компьютерах архитектуры x86 System Management Mode (SMM). Описана корректная конфигурация режима SMM, определены возможные функции безопасности с использованием SMM и указаны плюсы и минусы использования этой технологии для обеспечения безопасности. Проведен анализ и дана оценка целесообразности применения данной технологии в компьютерной безопасности.

Ключевые слова: System Management Mode, System Management Interruption, средство защиты информации, компьютерная безопасность, регистр SMI_EN, кольца защиты, режим исполнения кода, архитектура x86.

Архитектура x86 подразумевает наличие как минимум трех различных уровней привилегий для выполнения инструкций процессором [1]. Эти уровни привилегий называются кольцами защиты (англ. *protection rings*), Код, исполняемый в центральном кольце («ring 0»), обладает наибольшим доступом в операционной системе (ОС), а во внешнем кольце («ring 3») — наименьшим [1]. Эти уровни привилегий предназначены для реализации аппаратного разграничения доступа процесса к ресурсам ЭВМ (например, к портам ввода-вывода) и реализованы в ЭВМ в виде различных режимов работы центрального процессора (ЦП).

Однако кроме стандартных уровней привилегий для ОС, существуют также более привилегированные уровни (обладающие большим доступом, чем «ring 0»), которые нумеруются в отрицательную область чисел, с соответствующими режимами работы ЦП: «ring -1» — режим гипервизора и «ring -2» — System Management Mode (SMM) [1]. Также существует режим «ring -3», основанный на технологиях Intel Management Engine (ME) для процессоров Intel и AMD Secure Technology для процессоров AMD [2, 3]. В свою очередь, режимы гипервизора и SMM подразумевают использование технологий гипервизора и SMM, в которых используются выделенная (недоступная из менее привилегированных режимов) память и независимое от ОС программное обеспечение (ПО): ядро, приложения и драйвера.

При этом уже в режиме супервизора (т. е. на уровне «ring 0») исполняемые процессором инструкции обладают практически полным доступом к ресурсам ЭВМ и этот доступ может контролироваться только более привилегированными режимами [1]. Вместе с тем выполнение инструкций в режимах «rings -1, -2, -3» для ОС прозрачно и не отслеживаемо напрямую [4, 5]. Поэтому с одной стороны, такие технологии являются ключевыми для атак низкого уровня [6, 7] и могут представлять угрозы для средств защиты информации (СЗИ), функционирующих с привилегиями не ниже «ring 0», а с другой стороны, с помощью таких технологий можно расширить функциональность существующих СЗИ либо попытаться реализовать полнофункциональное СЗИ на основе таких технологий.

Используемые методы исследования

В данной статье из ранее упомянутых режимов будет рассматриваться только SMM. Цель работы — оценка перспективы использования технологии SMM в разработке СЗИ. В работе используются такие методы исследования, как анализ, абстрагирование, сравнение и эксперимент. Объектом исследования является SMM. Предмет исследования — возможность использования механизмов SMM для разработки СЗИ.

Обзор литературы

1. Описание SMM

SMM является привилегированным режимом выполнения кода у x86 совместимых процессоров (Intel, AMD), впервые реализованным компанией Intel в своих процессорах в середине 90-х годов [8]. С момента создания и до сих пор этот режим используют для совершения действий, незаметных и практически не отслеживаемых для ОС, но при этом выполняемый код обладает полным досту-

Пахомов Михаил Вадимович, студент.
E-mail: pahomov.mv@phystech.edu

Статья поступила в редакцию 18 октября 2021 г.

пом к памяти и всем подключенным устройствам [5].

Изначально SMM применяли в области управления питанием компонентов ЭВМ: обработчики событий в SMM собирали статистику по использованию устройств и в случае их долговременного простоя отключали [8]. То есть первоначально в задачи SMM не входило решение проблем обеспечения безопасности, а привилегированный режим был обусловлен необходимостью в постоянном контроле всех устройств ЭВМ. На данный момент задача управления питанием компонентов ЭВМ выполняется не с помощью SMM, а при помощи ОС [8].

Для переключения процессора в SMM используются System Management Interruptions (SMIs), которые генерируются компонентами материнской платы либо различными драйверами, приложениями (в том числе приложениями из ОС) или пользователями с административными правами в ОС [5]. К типовым системным SMI, которые поддерживаются в большинстве ЭВМ (в частности, согласно документации, у компьютеров с 8/9/200/300 сериями чипсетов Intel [9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4]), можно отнести следующие прерывания и соответствующие им события [5, 9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4]:

- GPIO Unlock SMI. Генерируется при снятии бита Lock (*GLE*) с регистров управления выводами GPIO. Обработчик проверяет ПО, снявшее бит, и если оно не авторизованное, выставляет бит обратно;
- TCO SMI. Генерируется при различных событиях. Обработчик прерывания выполняет действия согласно источнику прерывания:
  - Intel TCO watchdog-таймером обратного отсчета при опускании таймера до нуля. Этот таймер должен взводиться ОС каждые несколько секунд. Если таймер опустится до нуля, то произойдет прерывание, обработчик которого произведет перезагрузку системы;
  - при выставлении бита *BIOSWE* у регистра BIOS Control, отвечающего за возможность читать и писать в ПЗУ, где находится код BIOS'а. Если бит выставляется в SMM, то прерывание не будет сгенерировано, так как выполняемый код уже в SMM режиме; если бит выставляется в любом другом режиме, то прерывание будет сгенерировано, после чего будет вызван обработчик, который просто выставит бит обратно. Такой механизм реализован для защиты от перезаписи прошивки ЭВМ из любого режима, кроме SMM;
- APMC (APM Control) SMI. Генерируется при записи в APM_CNT I/O порт (почти всегда это порт 0xB2). Срабатывание такого прерывания может быть вызвано администратором ОС при помощи записи в ранее указанный порт. Количество обработчиков может быть 256. При этом при вызове можно указывать номер желаемого обработчика;
- IOTR (IO Trap) SMI. Генерируется при обращении к портам CPU I/O. Обработчик позволяет эмулировать Legacy-устройства (например, клавиатуру), которые раньше использовали I/O порты;
- xHCI (Extensible Host Controller Interface) SMI. Генерируется USB-контроллером при различных событиях;
- Periodic SMI. Генерируется чипсетом по таймеру с периодичностью 8/16/32/64 (период настраивается с помощью битов *PER_SMI_SEL*).

2. SMM memory (SMRAM)

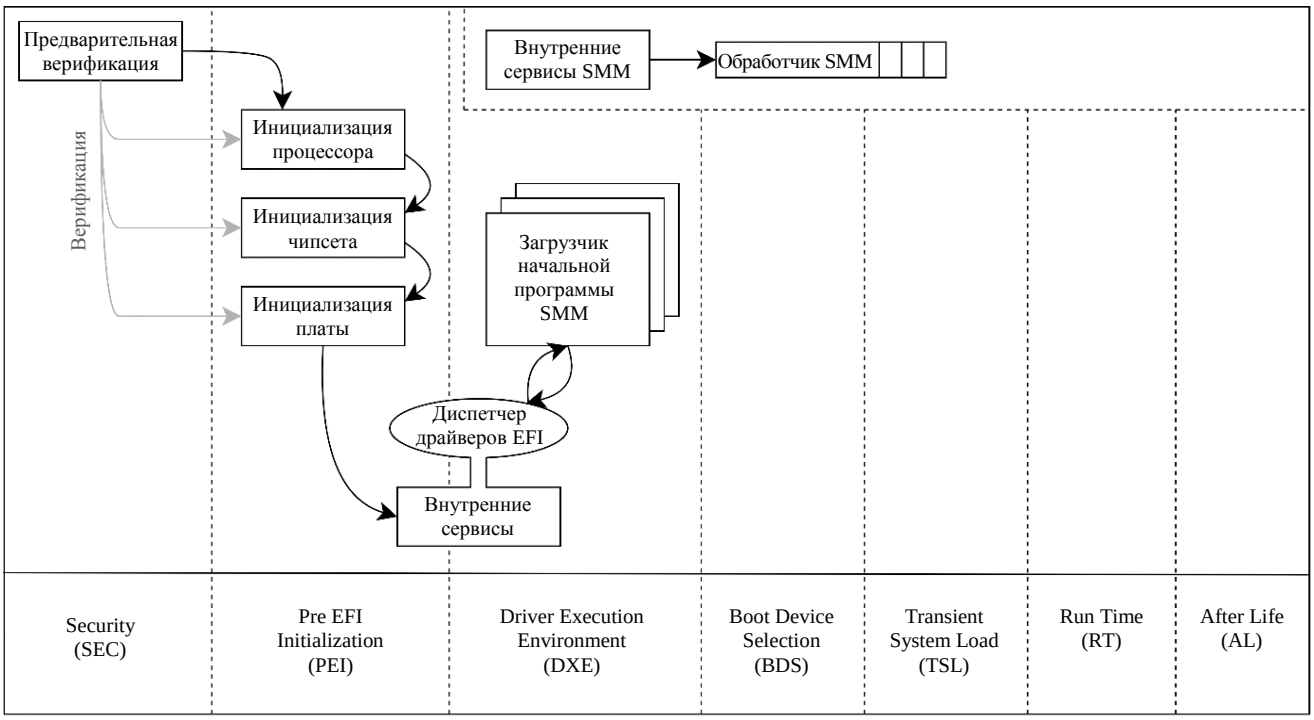
Для изолированности среды выполнения операций в режиме SMM используется память SMM memory (SMRAM), в которой хранятся все необходимые данные для работы SMM: код и обработчики прерываний, а также содержимое регистров (процессор сохраняет контекст при переключении в SMM). Если какая-либо операция выполняется не в режиме SMM, а в обычном режиме (менее привилегированном), то эта память недоступна, т. е. нельзя как прочитать данные из этого пространства, так и записать туда что-либо. SMRAM может состоять из следующих областей (вывод об использовании или не использовании сделан по отношению к современным ЭВМ со стандартной конфигурацией SMM) [5, 8]:

- ASEG [0x000A0000-0x000BFFFF] не используется;
- HSEG [0xFEDA0000-0xFEDBFFFF] не используется;
- TSEG (настраиваемый диапазон) используется.

3. Инициализация SMRAM

Целесообразно рассматривать инициализацию SMRAM и работу SMM с использованием UEFI BIOS, а не Legacy BIOS, так как большинство современных ЭВМ используют именно интерфейс UEFI, а Intel вовсе планирует исключить поддержку Legacy BIOS из своих устройств в ближайшие годы [13].

Весь код SMM (в том числе SMI-обработчики) хранится в коде UEFI BIOS, который находится в ПЗУ. Этот код выгружается в SMRAM и конфигурируется однократно при включении ЭВМ на стадии SMM, которая, в свою очередь, является частью фазы DXE (все стадии изображены на рисунке) [14]. После этого SMM-фаза активна в течение всего временного интервала активности ЭВМ параллельно другим Run Time (RT) фазам.



Фазы загрузки системы с UEFI

В процессе инициализации SMRAM выполняются инициализация физической памяти, настройка TSEG, копирование SMM-кода в физическую память, настройка таблицы дескрипторов. Одним из важнейших заключительных этапов является корректное выставление регистров, ответственных за корректную работу памяти и её защиту.

4. Регистры SMM

Для корректной работы SMRAM и настройки доступа к этой памяти используется несколько регистров. Основным регистром является System Management RAM Control (*SMRAMC*) регистр [15, ч. 3.29]. Значение *SMRAMC* выставляется при инициализации SMRAM, после чего регистр блокируется до следующего рестарта ЭВМ [16]. Среди битов этого регистра наиболее важны биты *D_OPEN* и *D_LCK*, которые должны быть установлены в 0 и 1 соответственно на любой корректно сконфигурированной системе, чтобы SMRAM-память была доступна только из кода, выполняемого в SMM.

Существуют производные регистры для обеспечения корректного доступа к памяти, которые созданы вследствие обнаружения различных векторов атак на SMM (эти регистры также должны быть корректно сконфигурированы и заблокированы аппаратным обеспечением):

- System Management Range Registers (*SMRR*) определяет области памяти, в которых запись не из SMM игнорируется, а тип памяти является некэшируемым. Должен быть корректно выставлен вендорами. Атака — SMM cache poisoning [17];
- *TSEGMB*-регистр у DMA-контроллеров дублирует информацию о местоположении TSEG, после чего запрещается писать в эту область с помощью DMA. Должен быть корректно выставлен вендорами. Атака — DMA [18];
- *SMI_LOCK* бит регистра General PM Configuration, *SMM_BWP* бит регистра BIOS Control отвечают за генерацию SMI при прошивании BIOS. Атака — отключение генерации SMI, после чего перепрошивка BIOS неавторизованным ПО [19].

Результаты

1. Функции безопасности с использованием SMM

SMM предназначен для обработки прерываний, которые сгенерированы либо системой (системные SMI), либо прошивкой/драйверами/приложениями, обладающими доступом с правами администратора в ОС (программные SMI) при помощи записи в APM_CNT I/O порт. Таким образом, возможно создание лишь двух типов функций (обработчиков SMI) на основе SMM:

- создание обработчика программных SMI. Вызывать такой обработчик можно с помощью ПО в ОС;

- расширение обработчика системных SMI. Вызываться такой обработчик будет автоматически при каких-то событиях (зависит от типа SMI).

При этом, учитывая особенности SMM (привилегированность, изолированность среды выполнения инструкций SMM, хранение кода SMM в BIOS), можно говорить о применении обработчиков SMI для создания функций, которым необходимо одно из свойств:

- выполнение привилегированных инструкций;
- выполнение инструкций в изолированной среде по отношению к программной среде в ОС;
- возможность хранения секретных данных малого размера (например, токены, сертификаты, криптографические ключи) в защищенном пространстве, т. е. использование части памяти ПЗУ, где хранится код BIOS, как небольшого хранилища информации, к которому можно получить доступ с помощью SMI-обработчиков.

Далее рассматриваются более подробно возможные функции безопасности, которые потенциально можно реализовать с помощью обработчиков SMI.

1.1. Создание обработчика программных SMI

Для вызова обработчика программных SMI необходим корректно выставленный бит *APMC_EN* регистра *SMI_EN* [9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4]. Этот бит является R/W и может быть перезаписан в любой момент, если не реализована функциональность блокировки регистра *SMI_EN*. Таким образом, не следует полагаться на гарантированное срабатывание прерывания из-за отсутствия встроенной функциональности блокировки.

Обработчик программных SMI подходит для реализации функций, которые требуется вызывать из ОС для выполнения заведомо определенных привилегированных инструкций. Примеры возможных функций безопасности:

- выполнение мгновенной перезагрузки/выключения системы при выявленных попытках несанкционированного доступа;
- проверка переданных учетных данных и расширение прав доступа пользователя;
- генерация дочерних сертификатов/ключей на основе корневого сертификата/ключа без возможности прочитать корневой сертификат/ключ;
- включение/отключение/проверка компонентов системы и периферийных устройств (например, проверка контрольной суммы кода BIOS или отключение одного из USB-устройств через интерфейс xHCI);
- настройка регистров, которые контролируют доступ к компонентам системы и периферийным устройствам (например, включение/выключение возможности записи на жесткий диск).

1.2. Расширение обработчика системных SMI

Для вызова обработчика системных SMI необходим корректно выставленный бит включения прерываний регистра *SMI_EN* для требуемого типа прерываний [9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4]. Большинство таких битов являются R/W и аналогично биту *APMC_EN* могут быть перезаписаны в любой момент, если отсутствует функциональность блокировки регистра *SMI_EN*. Таким образом, подобно программным SMI, нельзя полагаться на гарантированное срабатывание большинства системных SMI при отсутствии функциональности блокировки. Вместе с тем для некоторых прерываний предусмотрена встроенная функциональность блокировки соответствующего бита. Так, например, для всех современных ЭВМ архитектуры x86 такими битами являются *GPIO_UNLOCK_SMI_EN* и *TCO_EN*.

На базе таких обработчиков можно реализовать более специфичные функции с автоматически генерируемыми прерываниями различными компонентами системы при наступлении конкретных событий (зависит от компонента и типа SMI). Примеры возможных функций безопасности (обработчики):

- TCO SMI можно модифицировать для обработки запросов на перезапись прошивки;
- GPIO Unlock SMI можно модифицировать либо заменить своим обработчиком для контроля доступа ОС к GPIO-контактам;
- xHCI SMI (*xHCI_SMI_EN* бит является R/W) можно модифицировать либо дополнить необходимыми функциями, чтобы обрабатывать различные события, связанные с USB-устройствами [20, ч. 4.22.1];
- Periodic SMI (*PERIODIC_EN* бит является R/W) можно модифицировать либо дополнить необходимыми функциями, чтобы выполнять указанный код многократно с установленным периодом.

2. Плюсы и минусы SMM для СЗИ

Плюсы и минусы SMM разделены на фундаментальные и технические. Первые особо важны, так как они являются архитектурной особенностью режима, а следовательно, никаким образом

не могут быть значительно изменены в последующих обновлениях прошивки в отличие от технических атрибутов SMM.

2.1. Плюсы SMM

Фундаментальные:

- привилегированный доступ. В базовом сценарии SMM предоставляет максимальный доступ к системе среди прочих встроенных в систему технологий (за исключением технологий, которые базируются на РКБ);
- дешевизна и высокая бесперебойность. Решение на базе SMM не нуждается в дополнительном аппаратном оборудовании, а реализуется в уже существующем окружении, что положительно сказывается на бесперебойности СЗИ и количестве возможных аппаратных неисправностей, а также на стоимости самого решения.

Технические:

- ранняя стадия старта функционирования. SMM-код загружается и запускается в DXE-фазе загрузки UEFI. Эта фаза идет после фаз SEC и PEI и до фазы Boot Device Selection (BDS) (см. рисунок) [21]. С одной стороны, наличие двух фаз перед запуском SMM является, определенно, минусом, но с другой стороны, это позволяет коду SMM работать с памятью (которая инициализируется в PEI) и работать с устройствами системы. При этом полноценно функционировать SMM начинает после блокирования SMRAM до окончания фазы DXE, т. е. в конце фазы DXE SMRAM уже находится в заблокированном состоянии, и SMM может обрабатывать приходящие запросы. Поэтому в фазе BDS можно говорить о полном функционировании технологии SMM;
- встроенная защита кода SMM от перезаписи. Одной из основных задач SMM на текущий момент является обработка запросов системы на прошивание BIOS'а, где и хранится код SMM. Таким образом, при правильной конфигурации SMM можно защитить SMM-код от несанкционированных воздействий (не рассматривается прошивание памяти с помощью аппаратного воздействия).

2.2. Минусы SMM

Фундаментальные:

- доверие к вендору. При использовании SMM необходим РКБ для построения доверенной системы (в том числе для того, чтобы контролировать недоверенный процессор [22]) либо необходимо доверять процессору (который в таком случае будет выполнять некоторые задачи РКБ) и, как следствие, доверять вендору. Также, поскольку системные SMI генерируются различными компонентами системы (например, xHCI-контроллером, отвечающим за взаимодействие с USB), необходимо доверять этим контроллерам в части срабатывания прерываний при наступлении конкретных событий.

Технические:

- платформозависимость. Технология SMM сильно платформозависима и разработана только для x86-архитектуры. Также не менее важным является факт, что вендоры (в том числе Intel, AMD) не гарантируют наличие тех или иных системных SMI в системе по умолчанию (это можно проследить в документации [9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4]);
- ограничения по памяти. Согласно документации под сегмент TSEG SMRAM может быть выделено 1, 2 или 8 MB (мегабайт) [16, ч. 3.37], что ставит ограничения на разрабатываемый код;
- большинство битов в регистре *SMI_EN* являются R/W, поэтому нельзя полагаться на срабатывание конкретных SMI. Для устранения этого недостатка требуется разработать собственную логику в SMM, которая сможет блокировать необходимый бит.

Обсуждение

Таким образом, наиболее целесообразным и простым использованием SMM является применение этой технологии для реализации небольшого хранилища данных либо функций, к которым будет ограничен доступ на чтение и изменение посредством SMI обработчиков. При этом обеспечить взаимодействие с данной частью памяти и получить доступ к ней можно напрямую из ОС. Примеров реализации коммуникации кода ОС и SMM-кода в свободном доступе достаточно много [14, 23, 24].

С другой стороны, можно использовать автоматически генерируемые SMI, создаваемые Platform Controller Hub'ом (PCH). Однако этот способ является эффективным только в случае блокировки битов, отвечающих за срабатывание прерываний. Поэтому необходимо также реализовать дополнительную функциональность, которая будет отвечать за неизменность этих битов. Документация Intel не декларирует возможностей по блокировке таких битов [9, 10, ч. 12.8.3.7; 11, 12, ч. 5.2.4], а примеры атак на ЭВМ [25] с изменением таких битов еще раз подтверждают наличие векторов

атак в случае реализации СЗИ через SMI. Существует патент по реализации блокировки регистра *SMI_EN* [26], но он скорее описывает архитектурную возможность по аппаратному улучшению чипсета (в частности, южного моста чипсета), нежели возможность по реализации блокировки регистра с помощью программных средств. Таким образом, предлагаемый вариант блокировки в патенте может быть целесообразен для вендоров компьютерной техники, поскольку они могут вносить существенные изменения в структуру элементов ЭВМ, но не является целесообразным для разработчиков СЗИ.

Заключение

Из изложенного можно сделать вывод, что поскольку выполняемый в SMM код обладает достаточно привилегированным доступом в ЭВМ, этот режим остается крайне интересным для исследования на уязвимости в целях выработки рекомендаций по корректному конфигурированию. При этом из-за архитектурных особенностей данной технологии реализация каких-либо новых инструментов безопасности с использованием SMM представляет собой сложную и нецелесообразную задачу. Поэтому в современных ЭВМ необходимо правильно конфигурировать SMM и не следует полагаться на эту технологию при разработке СЗИ.

Литература

1. *Domas C.* The Memory Sinkhole [Электронный ресурс]. URL: https://www.blackhat.com/docs/us-15/materials/us-15-Domas-The-Memory-Sinkhole-Unleashing-An-x86-Design-Flaw-Allowing-Universal-Privilege-Escalation-wp.pdf (дата обращения: 14.11.2020).

2. *Oster J. E.* Getting Started with Intel® Active Management Technology (Intel® AMT) [Электронный ресурс]. URL: https://software.intel.com/content/www/us/en/develop/articles/getting-started-with-intel-active-management-technology-amt.html (дата обращения: 18.11.2020).

3. О безопасности UEFI, часть заключительная [Электронный ресурс]. URL: https://habr.com/ru/post/268423/ (дата обращения: 18.11.2020).

4. *Jiewen Y., Zimmer J. V.* A Tour Beyond BIOS Launching STM to Monitor SMM in EDK II [Электронный ресурс]. URL: https://software.intel.com/content/dam/develop/external/us/en/documents/a-tour-beyond-bios-launching-stm-to-monitor-smm-in-efi-developer-kit-ii-819978.pdf (дата обращения: 18.11.2020).

5. О безопасности UEFI, часть вторая [Электронный ресурс]. URL: https://habr.com/ru/post/267197/ (дата обращения: 18.11.2020).

6. *Rauchberger J., Luh R., Schrittwieser S.* LONGKIT – A Universal Framework for BIOS/UEFI Rootkits in System Management Mode // Conference: 3rd International Conference on Information Systems Security and Privacy. P. 346—353.

7. Банк данных угроз безопасности информации. SMM [Электронный ресурс]. URL: https://bdu.fstec.ru/search?q=SMM (дата обращения: 18.11.2020).

8. SMM и SMRAM или 128 Кб потусторонней памяти: исследовательская работа № 5 [Электронный ресурс]. URL: https://xakep.ru/2008/07/29/44663/ (дата обращения: 18.11.2020).

9. Intel® 8 Series/C220 Series Chipset Family Platform Controller Hub (PCH) [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/8-series-chipset-pch-datasheet.pdf (дата обращения: 18.11.2020).

10. Intel® 9 Series Chipset Family Platform Controller Hub (PCH) [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/9-series-chipset-pch-datasheet.pdf (дата обращения: 18.11.2020).

11. Intel® 200 (Including X299) and Intel® Z370 Series Chipset Families Platform Controller Hub (PCH). Volume 2 [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/200-series-chipset-pch-datasheet-vol-2.pdf (дата обращения: 18.11.2020).

12. Intel® 300 Series and Intel® C240 Series Chipset Families Platform Controller Hub (PCH). Volume 2 [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/300-series-chipset-pch-datasheet-vol-2.pdf (дата обращения: 18.11.2020).

13. Intel to Remove Legacy BIOS Support from UEFI by 2020 [Электронный ресурс]. URL: https://www.anandtech.com/show/12068/intel-to-remove-bios-support-from-uefi-by-2020 (дата обращения: 09.05.2021).

14. Building reliable SMM backdoor for UEFI based platforms [Электронный ресурс]. URL: http://blog.cr4.sh/2015/07/building-reliable-smm-backdoor-for-uefi.html (дата обращения: 18.11.2020).

15. 8th and 9th Generation Intel® Core™ Processor Families and Intel® Xeon E Processor Family [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/8th-gen-core-family-datasheet-vol-2.pdf (дата обращения: 18.11.2020).

16. Intel® Platform Innovation Framework for EFI System Management Mode Core Interface Specification (SMM CIS) [Электронный ресурс]. URL: https://www.intel.ru/content/dam/doc/reference-guide/efi-smm-cis-v091.pdf (дата обращения: 18.11.2020).

17. Attacking SMM Memory via Intel® CPU Cache Poisoning [Электронный ресурс]. URL: https://invisiblethingslab.com/resources/misc09/smm_cache_fun.pdf (дата обращения: 18.11.2020).

18. Attacking UEFI Boot Script [Электронный ресурс]. URL: https://bromiumlabs.files.wordpress.com/2015/01/venamis_whitepaper.pdf (дата обращения: 18.11.2020).

19. О безопасности UEFI, части нулевая и первая [Электронный ресурс]. URL: https://habr.com/ru/post/266935/ (дата обращения: 18.11.2020).

20. eXtensible Host Controller Interface for Universal Serial Bus (xHCI) [Электронный ресурс]. URL: https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/extensible-host-controler-interface-usb-xhci.pdf (дата обращения: 18.11.2020).

21. Устройство файла UEFI BIOS, часть полуторная: UEFI Platform Initialization [Электронный ресурс]. URL: https://habr.com/ru/post/185764/ (дата обращения: 18.11.2020).

22. *Елькин В. М.* Контроль недоверенного процессора: Мат. XXIII Научно-практической конференции "Комплексная защита информации". Суздаль, 22—24 мая 2018 г. — М.: Медиа Групп "Авангард", 2018. С. 209—211.

23. System Management Mode Hacks [Электронный ресурс]. URL: http://phrack.org/issues/65/7.html (дата обращения: 18.11.2020).

24. Использование Intel Processor Trace для трассировки кода System Management Mode [Электронный ресурс]. URL: https://habr.com/ru/company/dsec/blog/481692/ (дата обращения: 18.11.2020).

25. Advanced x86: Introduction to BIOS & SMM. SMI Suppression [Электронный ресурс]. URL: https://opensecuritytraining.info/IntroBIOS.html (дата обращения: 18.11.2020).

26. *Ziarnik G. P., Durham M. R., Piwonka M. A.* Pattern № US9483426B2, United States (US). Locking a system management interrupt (SMI) enable register of a chipset. Application № US14/364,706. PCT Filed 31.01.2012. PCT № PCT/US2012/023225. PCT Date 12.06.2014; Publication Date 01.11.2016. Assignee: Hewlett-Packard Development Company, L.P.

SMM technology and its application in computer security

M. V. Pakhomov
Moscow Institute of Physics and Technology (National Research University),
Dolgoprudny, Moscow region, Russia

System Management Mode (SMM) is a highly privileged mode on most x86 computers, and there are many ways to exploit this mode to attack devices. In this article, the correct configuration of the mode is described, possible security functions using SMM are identified, and the pros and cons of using the technology for security are indicated. The result of the work was the analysis and assessment of the feasibility of using the technology in computer security.

Keywords: System Management Mode, System Management Interruption, information security tool, computer security, SMI_EN register, protection rings, code execution mode, x86 architecture.

Bibliography — 26 references. *Received October 18, 2021*