УДК 621.384.3

**В. С. Соколовський, В. В. Карпінець,
Ю. Є. Яремчук, Д. П. Присяжний, А. В. Приймак**
Вінницький національний технічний університет
Хмельницьке шосе, 95, 21021 Вінниця, Україна

Захист віртуальних машин на основі інструкцій нового покоління процесорів AMD Zen

Розглянуто можливість використання спеціалізованого криптографічного ядра процесора AMD Zen Pro для підвищення захисту віртуальних машин від несанкціонованого доступу, зокрема, через мережу Інтернет. Запропонований метод дозволяє виконувати шифрування та дешифрування захищених сторінок оперативної пам'яті на апаратному рівні, що забезпечує стійкість віртуальних машин до багатьох активних атак, а також дозволяє підвищити їхню швидкодію в цілому. Наведено приклад реалізації запропонованого методу для операційної системи Windows 10.

Ключові слова: *захист інформації, віртуалізація, гіпервізор, криптографічний процесор, шифрування пам'яті, технологія Secure Encrypted Virtualization, Secure Memory Encryption, мікроархітектура AMD Zen.*

Вступ

Інформаційні технології дали можливість виконувати велику кількість користувацьких задач швидко та безпомилково. Сьогодні обчислювальні потужності комп'ютерних систем використовуються для бізнес-проектів і рішень будь-якого масштабу. Проте з прогресивним ходом технологій проблема інформаційної безпеки завжди залишається актуальною [1].

Традиційно в контексті захисту інформаційних технологій розуміють декілька аспектів, з точки зору апаратно-програмного забезпечення — фізичний захист серверних приміщень і мережевої апаратури, а також широкий спектр технологій для захисту програмного забезпечення від несанкціонованого доступу до його ресурсів, що можуть включати в себе як алгоритми роботи, так і конфіденційні дані користувача. Проте до 2017 року левова частка витрат на захист інформації спрямовувалася саме на захист програмного забезпечення, так як виконання машинного коду серверною частиною вважали достатньо безпечним, у той час як гнучкі та потужні технології віртуалізації з потенціалом для бізнес-рішень потребували постійного нагляду [2].

Багато IT-організацій вирішують проблеми безпеки, застосовуючи узагальнену політику безпеки: всі віртуальні машини повинні запускати перевірене та найбільш актуальне програмне забезпечення. Віртуальні машини можуть сприяти більш ефективним моделям використання, які можуть використовувати для роботи непідтверджені або застарілі версії програмного забезпечення. Це створює ряд проблем, оскільки треба створювати та підтримувати патчі або інший захист для широкого кола операційних систем, а також враховувати ризики, що пов'язані з наявністю в мережі безлічі комп'ютерів з потенційними вразливостями. Наприклад, сьогодні клієнти багатьох хостинг-сервісів просто користуються віртуальними машинами, що працюють над своїм новим операційним середовищем, і програми поступово переносяться до цього середовища або, навпаки, застарілі програми запускаються у віртуальній машині. Це може зменшити потребу в тривалих і складних циклах оновлення, але призводить до поширення різних версій операційних систем (ОС). Це, в свою чергу, ускладнює управління патчами, особливо за наявності застарілих версій операційних систем.

Руткіти для гіпервізора на прикладі Blue Pill дозволяють основній операційній системі продовжувати працювати безпосередньо з апаратним забезпеченням напряму, в той час мігруючи її до стану гостьової ОС, перезаписуючи основну ОС шкідливим кодом. Blue Pill може зробити інсталяцію «вручну» і просто мігрувати операційну систему з прямого управління фізичним комп'ютером у віртуалізований стан під контролем вищенаведеного руткіту. Даний руткіт виступає в ролі «надтонкого» гіпервізора, який в більшості випадків залишається деактивованим, використовуючи фактично нульові ресурси процесора та оперативної пам'яті та очікуючи задані зловмисником події, такі як введення даних з клавіатури.

Опубліковані компанією Google в 2017 році статті [3, 4] виявили низку критичних вразливостей процесорів, що дозволяли обійти майже будь-які програмні засоби захисту та за допомогою низькорівневого коду отримати доступ до пам'яті ядра операційної системи (що в нормальному стані є захищеною від користувача) та отримати конфіденційні дані або отримати доступ до машини. Найбільш вразливими виявилися процесори компанії Intel, інші процесори (AMD на базі x86-64, ARM на базі AArch64, MIPS, SPARC та IBM POWER) були компрометовані лише частково.

Вразливість даних процесорів пов'язана, в першу чергу, з їхньою технологією спекулятивного виконання, що значно підвищує їхню швидкодію [4]. Суть технології полягає в тому, що під час виконання інструкції переходу (якщо потребується операнд умовного оператора, що обчислюється в іншому потоці) конвеєр може не дочікуватися результату, а виконувати обидва розгалуження коду спекулятивно (наперед), допоки результат для умовного операнду не буде отриманий. У такому випадку конвеєр відкидає непотрібні результати та продовжує виконання правильної гілки. Також процесори використовують модуль передбачення розгалужень коду, що намагається наперед визначити гілку виконання на основі попередніх результатів обчислень, при цьому при невірному прогнозі конвеєр очищається. Проблема полягає в тому, що деякий час перед очищенням відкинуті дані можуть зберігатися в кеші процесора. І хоча доступ до них напряму отримати неможливо, зловмисник за допомогою маніпуляції кодом може передбачити, які саме дані надійш-

ли до кешу, так як запит до оперативної пам'яті на процесорах Intel отримує дані в кеш ще до того, як перевіряє можливість доступу до даної сторінки пам'яті.

Якщо зловмисник може ідентифікувати відповідний пакет даних на більш привілейованому рівні, він може використати його, щоб вивести вміст пам'яті, доступний для операційної системи, але не користувачу чи атакуючому.

Націлена ін'єкція інструкцій розгалуження використовує переваги непрямих передбачень всередині процесора, які використовуються для керування спекулятивним виконанням операцій. Впливаючи на те, як працюють непередбачувані, проте прогностичні фактори, зловмисник може призвести до спекуляції виконання шкідливого коду, а потім використовувати привілеї, які використовує код для кеш-пам'яті, для виявлення значень даних.

Для умовних інструкцій існує лише два варіанти того, який код спекулятивно виконує процесор — або виконання переходить до вказівника на комірку пам'яті, де знаходиться інструкція, що буде виконана при виконанні умови, або до інструкції безпосередньо після розгалуження. Зловмисник не може примусово виконувати код за межами цих місцеположень. Однак непрямі розгалуження можуть викликати спекулятивне виконання коду на більш широкому наборі задач.

Можливість втручатись у прогностичні модулі процесора та використовувати ці вразливості залежить від мікроархітектурної реалізації, тому точні методи можуть варіюватись у різних сімействах процесорів від покоління до покоління. Наприклад, непрямі прогнозні фактори деяких процесорних реалізацій можуть використовувати лише підмножину загальної адреси для індексування в модуль-предиктор. Якщо зловмисник може розпізнати, яка підмножина бітів використовується, він може використовувати цю інформацію для створення перешкод через накладання на вказівник.

Метод сторонніх даних кешу процесора — завантаження сторонніх даних у кеш процесора. Цей метод передбачає застосування безпосередньо зондування ядра (супервізора) пам'яті. Така операція, як правило, призводить до помилки програми (помилка сторінки через дозволи сторінок таблиці, яких програма зловмисника не має і не повинна мати). Проте така операція може бути спекулятивно виконана за певних умов у певних реалізаціях архітектури мікропроцесорів. Наприклад, у деяких реалізаціях така спекулятивна операція передає дані лише для наступних операцій, якщо дані знаходяться в кеш-пам'яті даних найнижчого рівня (L1). Це може дозволити короткочасний доступ до значень даних, які запитує програма, і призводить до побічного каналу витоку, який відображає дані в пам'яті операційної системи. Цей метод застосовується тільки до регіонів пам'яті, які призначені супервізором.

Захист програмного забезпечення, який рекомендує корпорація Intel, полягає в тому, щоб установити бар'єр для зупинення спекулятивного виконання у відповідних місцях. Зокрема, для цієї мети рекомендується використовувати інструкцію LFENCE [5]. Інструкції щодо серіалізації, а також інструкція LFENCE зупиняють молодші інструкції перед їхнім виконанням, навіть спекулятивно, до того, як попередні інструкції будуть опрацьовані, але LFENCE пропонує кращу продуктивність, ніж інші інструкції для серіалізації. Інструкція LFENCE, яка вставлена після перевірки меж масиву, перешкоджає виконанню нових операцій, перш ніж закінчиться перевірка відповідності. Потрібно зауважити, що вставка LFENCE повинна бути

здійснена розумно; якщо воно використовується занадто часто — продуктивність може бути значно скомпрометованою.

Можна також створити набір правил статичного аналізу, щоб допомогти знайти місця в програмному забезпеченні, де може знадобитися бар'єр для спекулятивного виконання. Аналіз ядра Linux, який виконала компанія Intel, наприклад, виявив лише кілька місць, де потрібно вставити LFENCE, що призвело до мінімальної утрати ефективності. Як і у випадку з усіма інструментами статичного аналізу, в результатах, ймовірно, будуть помилкові позитивні результати, і рекомендується проведення інспекції операторами вручну.

Вищезгадані технології віртуалізації повсякденно використовуються сьогодні для імплементації хмарних технологій, хостингу веб-додатків і розміщення продуктів. Багато потужних IT-компаній, такі як Microsoft Azure, Oracle Zen, VMWare (KVM) пропонують широкий спектр технологій для безпечного розгортання бізнес-проектів [6]. Для комерціалізації безпека інформації має ключовий характер, тому захист клієнтських даних має розпочинатися саме з машинного рівня.

Постановка задачі та методика дослідження

Розглянути можливість використання криптографічного ядра процесора AMD Zen Pro для підвищення захисту віртуальних машин від несанкціонованого доступу (НСД) та інших активних зловмисних атак і запропонувати відповідний метод. Крім того проаналізувати можливість підвищення швидкодії роботи віртуальних машин за рахунок апаратного шифрування захищених сторінок оперативної пам'яті.

Використання технології Secure Encrypted Virtualization процесорів AMD Zen PRO для захисту віртуальних машин

У 2018 році компанія AMD запропонувала своє рішення щодо забезпечення захищеності даних в оперативній пам'яті гіпервізора. Технологія є доступною для нового покоління процесорів мікроархітектури Zen PRO (Ryzen PRO та серверних EPYC) [7]. Технологія базується на використанні криптографічного модуля, що використовує спеціалізоване ядро ARM Cortex для забезпечення швидкої, безперебійної і безпечної роботи з фізичною оперативною пам'яттю машини, а також для управління розподіленням ключів віртуальних машин.

Secure Encrypted Virtualization (SEV) [8] інтегрує основні можливості шифрування пам'яті з існуючою архітектурою віртуалізації AMD-V для підтримки зашифрованих віртуальних машин. Шифрування віртуальних машин може захистити їх не тільки від фізичних загроз, але й від інших віртуальних машин або навіть самого гіпервізора. Таким чином, SEV являє собою нову парадигму безпеки віртуалізації, яка особливо застосовна до хмарних обчислень, коли віртуальні машини не повинні повністю довіряти гіпервізору та адміністратору їхньої хостової системи. Як і для малого та середнього бізнесу, для підтримки SEV не потрібні жодні зміни програмного забезпечення.

Основне шифрування пам'яті виконується через виділене апаратне забезпечення в контролерах пам'яті на вході. Кожен контролер включає високопродук-

тивне стандартне шифрування (AES), яке шифрує дані, коли вони записуються на DRAM, і розшифровують його при зчитуванні.

Шифрування даних виконується за допомогою 128-бітного ключа в режимі, який використовує додаткові фізичні налаштування на основі адреси для захисту від переміщення шифр-текстового блоку атаки. Ключ шифрування, який використовує алгоритм AES з SEV, послідовно створюється при кожному скиданні системи і не впливає на будь-яке програмне забезпечення, що працює на серверах ЦП. Цей ключ керується цілком захищеним процесором AMD (що використовує технологію AMD SME — Secure Memory Encryption) [9], 32-розрядним мікроконтролером (ARM® Cortex®-A5), який функціонує як спеціальна підсистема безпеки, що інтегрована в AMD SoC на базі AMD Ryzen PRO, AMD Ryzen Threadripper, а також серверних процесорів AMD EPYC. Ключ створюється за допомогою бортового NIST SP 800-90 сумісного апаратного генератора випадкових чисел і зберігається в спеціальних апаратних регістрах, де він ніколи не виявляється за межами SoC. SEV не вимагає, щоб програмне забезпечення, яке працює на ядрі ЦП, брало участь у керуванні ключами.

Контроль за сторінками пам'яті, які зашифровуються, здійснюється через ОС або HV у програмному забезпеченні. Біт 47 фізичної адреси (також називається C-біт, як і в enCrypted) використовується для позначки, чи захищена сторінка пам'яті. ОС або HV задає біт 47 фізичної адреси у таблицю сторінок (PTE, Page Table Extensions), щоб позначити, чи сторінка повинна бути зашифрована і розшифрована алгоритмом AES в контролері SME [9].

Безпечне шифрування сторінок визначає просту й ефективну архітектурну можливість шифрування основної пам'яті. У той час як технології шифрування пам'яті були використані раніше в різних спеціалізованих продуктах і галузях, SEV є механізмом загального призначення, який є гнучким, інтегрованим у архітектуру процесора, масштабовану від вбудованих до високопродуктивних робочих навантажень сервера і не вимагає модифікацій прикладних програм.

Основне шифрування пам'яті може бути використано для захисту системи від різних атак. Хоча дані зазвичай шифруються сьогодні, коли вони зберігаються на диску, вони зберігаються в DRAM.

Це може залишити дані вразливими для відстеження неавторизованими адміністраторами або програмним забезпеченням, або апаратним зондуванням. Нова технологія енергонезалежної пам'яті (NVDIMM) посилює цю проблему, оскільки чіп NVDIMM може бути фізично вилучений із системи з неповною інформацією, подібною до жорсткого диска. Без шифрування будь-яка збережена інформація, така як конфіденційна інформація, паролі або секретні ключі, може бути легко скомпрометована.

Традиційні обчислювальні системи функціонували за допомогою кільцевої моделі безпеки. У цій моделі високий код привілею має повний доступ до ресурсів на його рівні та всіх нижчих привілейованих рівнях. У моделі SEV код, що виконується на різних рівнях (а саме гіпервізор або гість), є ізольованим і не має доступу до ресурсів іншого. Хоча рівень гіпервізора є традиційно більш привілейований, ніж рівень гостя, SEV розділяє ці рівні через криптографічну ізоляцію.

Отже, SEV-технологія побудована на основі моделі загрози, де передбачається, що зловмисник матиме доступ не лише до привілейованого коду користувача на

цільовій машині, але й потенційно може виконувати шкідливе програмне забезпечення на рівні вищого привілейованого гіпервізора. Зловмисник також може мати фізичний доступ до машини, включаючи самі чіпи DRAM. У всіх цих випадках SEV надає додаткові гарантії, які допомагають захистити віртуальний машинний код гостя та дані від зловмисника. Зауважимо, що SEV не захищає від DDoS-атак в обслуговуванні гостя.

Побудована за концепцією апаратних віртуальних машин, SEV може бути використана для підвищення безпеки в різних задачах. Завдяки використанню шифрування основного пакета, SEV надає всі ті самі переваги захисту, що й SEV для захисту від фізичних атак, що описані раніше. Адміністратор зі шкідливими намірами в хмарному центрі обробки даних не зможе отримати доступу до даних у розміщеній VM.

Завдяки застосуванню конструктивних особливостей віртуальної машини SEV побудовано на основі ідеї захищених середовищ для пісочниці, де програмне забезпечення (ПЗ) може виконуватися незалежно та захищатися від усіх інших програм у системі.

Ці пісочниці можуть бути настільки ж великими, як і повноцінний віртуальний мультимедіа-сервер з власним диском та операційною системою, але вони також можуть бути невеликими і використовуватися для більш дрібної ізоляції. Наприклад, обладнання для SEV може використовуватися для криптографічного ізолювання контейнерів Docker від хост-системи, щоб краще захистити конфіденційні дані.

Шифрування та дешифрування пам’яті через модуль AES може внести невелику кількість додаткової затримки доступу до пам’яті DRAM. Вплив цієї затримки на програмне забезпечення дуже залежить від робочого навантаження системи, але за оцінками, це дуже мало впливає на продуктивність системи. Якщо лише підмножина пам’яті зашифрована, вплив на продуктивність буде невеликим, оскільки це не вимагає додаткової затримки.

Дослідження [11] показало, що технологія дозволяє підтримувати ECC-пам’ять DDR3 та DDR4 з робочою частотою до 3200 МГц, з утратою швидкості при потоковому шифруванні пам’яті менше 10 %, що є набагато кращим результатом, ніж попередні аналогічні технології, і дозволяє підвищити швидкість процесу шифрування віртуальних машин. При шифруванні на програмному рівні втрата швидкості складає від 50 % до 200 %. Таким чином, вказана технологія дозволяє підвищити швидкість шифрування пам’яті порівняно із шифруванням на програмному рівні більш ніж на 40 %.

Розглянемо два різні приклади моделей, за допомогою яких програмне забезпечення може використовувати функцію SEV. У першій моделі всі DRAM зашифровані, а в другому — зашифрована лише вибрана область (відповідна гостьовим VM).

При повному шифруванні пам’яті весь вміст DRAM зашифровано за допомогою випадкового ключа, який забезпечує надійний захист від «холодного завантаження», віддаленого перегляду інтерфейсу DRAM і подібних типів атак. Для систем з NVDIMM повне шифрування пам’яті також забезпечує захист від зловмисника, видаляючи модуль пам’яті та намагаючись витягти його вміст.

Підтримка шифрування повного запам'ятовуючого пристрою з SEV здійснюється як ОС, так і HV, встановлюючи C-біт у всіх фізичних адресах DRAM у всіх таблицях сторінок. Це повинно включати як сторінки інструкцій, так і дані, а також сторінки, що відповідають самим таблицям сторінок. По суті, програмне забезпечення ОС або HV може розглядати систему як DRAM, що починається з адреси 0x8000_0000_0000. У випадку встановлення HV на C-біт для шифрування всієї фізичної пам'яті всі VM, що керуються HV, шифруються разом з HV і використовують один і той же ключ шифрування.

Використання C-біту в таблицях сторінок забезпечує гнучкість для ОС або HV, щоб вибірково шифрувати лише частину пам'яті, якщо це потрібно. Все це забезпечує не лише фізичний захист зашифрованої пам'яті, але також може використовуватися для підвищення продуктивності.

Крім того, вибір зашифрованої чи незашифрованої пам'яті може бути використаний для забезпечення шару ізоляції для критичних навантажень.

Одним із прикладів цієї ізоляції може бути система, яка лише позначає пам'ять, яку гостьові VM використовують як зашифровану. Це можна зробити без будь-яких модифікацій гостьових VM-кодів. Встановлюючи C-біт у всіх вкладках таблиці вкладок, що відповідають DRAM, гіпервізор дозволяє шифрувати тільки для пам'яті VM. Такий сценарій може бути використаний для захисту віртуальних машин від несанкціонованого адміністратора машини. Хоча адміністратор матиме апаратний доступ і доступ до хост-системи, але він не зможе перевіряти дані про гостьові VM, навіть за допомогою утиліт сканування пам'яті.

Приклад реалізації запропонованого методу захисту віртуальних машин для операційної системи Windows 10

Розглянемо абстрактну модель системи захисту, що базується на спеціалізованій службі Windows NT (Kernel version 10.X), яка може отримувати доступ до інтерфейсу даних інструкцій процесора та виконувати шифрування пам'яті на ходу. Служба напряму підключається до хоста Hyper-V, що є технологією віртуалізації на основі гіпервізору 1 типу, яку запропонувала Microsoft для своєї серверної платформи [10].

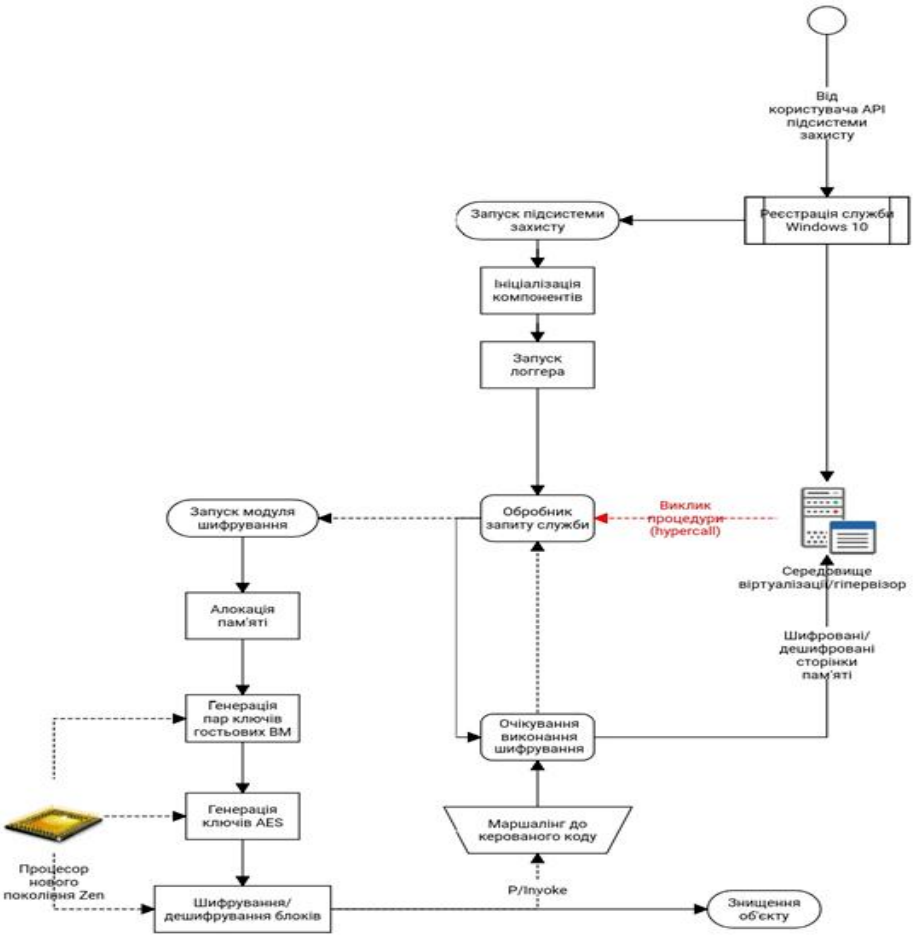
Адміністратори мають багато переваг у налаштуванні параметрів безпеки віртуальної машини в Hyper-V Manager, щоб захистити дані та стан віртуальної машини. Існують можливості захисту віртуальних машин від перевірки, крадіжки та втручання шкідливих програм, які можуть працювати на хості центрів обробки даних. Рівень безпеки, який ви отримуєте, залежить від деяких факторів — на якому комп'ютері ви хочете встановити віртуальну машину, а також від того, чи встановлюєте ви цю службу (Служба підтримки хоста), яка дозволяє хостам запускати захищені віртуальні машини.

Служба підтримки сторожової служби є новою роллю у Windows Server 2016. Він визначає довірені хости Hyper-V і дозволяє їм запускати певну віртуальну машину. Найчастіше служба охоронця хоста встановлюється для центру обробки даних. Також існує можливість створити екрановану віртуальну машину щоб запустити її локально, не встановлюючи дану службу. Адміністратор має змогу пізніше поширювати екрановану віртуальну машину у захищеному середовищі віртуалізації.

Якщо вищенаведена служба не встановлена або запускається в локальному режимі на хості Hyper-V — є можливість змінити параметри. Власник ключа охорони — це організація, яка створює приватні та відкриті ключі та надає їм доступ до всіх віртуальних машин, створених за допомогою цього ключа.

Налаштування безпечного завантаження в менеджері Hyper-V — це функція, що доступна для віртуальних машин 2 покоління, яка допомагає запобігти неавторизованому вбудованому програмному забезпеченню, операційним системам або драйверу UEFI. Захист завантаження ввімкнутий за замовчуванням. Ви можете використовувати безпечне завантаження з віртуальними машинами покоління 2, які працюють під управлінням операційної системи Windows або Linux.

На рисунку зображено алгоритм роботи запропонованого методу. Пунктирними лініями позначено асинхронні виклики процедур до некерованого коду бібліотек C++, а також виклики процедур гіпервізора (hypercalls) для отримання службою доступу до програмного інтерфейсу Hyper-V, які можуть виникати паралельно з виконанням основного алгоритму.



Алгоритм роботи запропонованого методу захисту віртуальних машин від НСД

Перед запуском підсистеми користувач або розробник реєструє її як службу Windows. За допомогою гнучкого налаштування прив’язки служб підсистема

встановлюється таким чином, щоб вона могла приймати запити від сервера віртуалізації Hyper-V. Одразу після запуску операційної системи служба завантажує файли, ініціалізує об'єкти в оперативній пам'яті і стає готова приймати запити. Як тільки запит надходить від гіпервізора до обробника події — модуль шифрування викликається за допомогою маршалінга та вступає в дію, ініціалізуючи свої ресурси (ассети).

Після ініціалізації класів модуля шифрування викликаються функції malloc і calloc (для виділення фіксованого буфера пам'яті та подрібненого буфера пам'яті відповідно), причому викликаються вони залежно від конкретної потреби ПЗ. Функція malloc виділяє необхідну кількість оперативної пам'яті та повертає вказівник на нульову комірку пам'яті, в той час як calloc — вказівник на нульовий елемент масиву заданої довжини. Пам'ять буде очищена після завершення циклу виконання модуля.

Знімок оперативної пам'яті (її частина — таблиця, або уся пам'ять гостьової ОС гіпервізора) зчитується процедурами асемблера, які викликаються модулем, і вказівник до нього повертається модулю.

Модуль починає процедуру шифрування (або дешифрування), використовуючи нові технології AMD SEV, AMD SME та Intel AES-NI.

Після завершення роботи модуль передає контроль до керованого коду C# за допомогою P/Invoke, де гіпервізор отримує потрібні йому зашифровані або дешифровані дані, а також метадані про події. Одразу після цього викликається деструктор і процедура free для очищення оперативної пам'яті та вивільнення ресурсів.

Служба записує (виконує логгінг) усіх подій, та записує їх за потреби до системного журналу подій Windows. Обробник подій починає асинхронно слухати наступні запити прив'язаних компонентів системи.

Висновки

Розглянуто можливість використання криптографічного ядра процесора мікроархітектури AMD Zen Pro та технології Secure Encrypted Virtualization для посилення безпеки віртуальних машин від НСД за допомогою безпечного керування ключами асиметричних алгоритмів і підвищення швидкодії шифрування та дешифрування захищених сторінок оперативної пам'яті за допомогою підтримки апаратного прискорення більш ніж на 40 %.

Запропоновано метод захисту віртуальних машин з використанням інструкцій нового покоління процесорів AMD Zen Pro для забезпечення безпеки серверів і середовищ віртуалізації для бізнесу різних масштабів. Запропонований метод дозволяє значно знизити ризики компрометації серверів віртуалізації через мережу Інтернет, а також від більшості руткітів гіпервізорів, що дає можливість захистити гостьові операційні системи від витоку даних.

Наведено опис методу роботи криптографічного модуля, а також методу взаємодії з компонентами Windows на основі P/Invoke та InteropServices.

1. Stallings W., Brown L. Computer security: principles and practice 3d ed. 2015. 842 c.

2. Lars Kurth. Введение в виртуализацию Xen Project. 2017. URL: https://dou.ua/lenta/articles/an-introduction-to-xen-project-virtualisation/

3. Xen Project Software Overview. 2018. URL: https://wiki.xenproject.org/wiki/Xen_Project_Software_ Overview/

4. Гипервизор виртуализации Xen достиг 4 уровня. 2018. URL: http://linuxsam.org.ua/gipervizor-virtualizatsii-xen-dostig-4-urovnya/

5. Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4, 2018. 4844 c.

6. Виртуализация Intel VT-d на марше. URL: http://composter.com.ua/content/virtualizaciya-intel-vt-d-na-marshe

7. Технология AMD-V для виртуализации клиентов. URL: https://www.amd.com/ru/technologies/virtualization

8. An Introduction to Qubes OS. 2017. URL: https://www.qubes-os.org/intro/#arent-antivirus-programs-and-firewalls-enough

9. Bhanu P Tholeti. Learn about hypervisors, system virtualization, and how it works in a cloud environment. 2011. URL: https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/index.html.

10. Пастухов Д.А., Юрчик П.Ф., Остроух А.В. Сравнительный анализ гипервизоров. *Международный журнал экспериментального образования*. 2015. № 3. С. 346–350.

11. ARM CoreLink Memory Controllers. ARM Cortex TrustZone Ex-tensions. 2017. URL: https://developer.arm.com/products/system-ip/memory-controllers?ga=2.89282392.235924928.1538121279-2131351941.1538121279

Надійшла до редакції 07.09.2018