DOI: https://doi.org/10.36910/6775-2524-0560-2021-44-17
УДК 004.31
Костючко Сергій Миколайович, к. т .н., доцент
https://orcid.org/0000-0002-1262-6268
Ольшевський Олег Вадимович, магістр
Луцький національний технічний університет

ОГЛЯД АПАРАТНИХ ВРАЗЛИВОСТЕЙ ВБУДОВАНИХ СИСТЕМ НА ПРИКЛАДІ ROWHAMMER

Костючко С.М., Ольшевський О.В. Огляд апаратних вразливостей вбудованих систем на прикладі Rowhammer. У статті розглянуто вразливості вбудованих систем на прикладі Rowhammer. Розкрито суть вразливості та способи її використання для вчинення протиправних дій. Також описано особливості цієї вразливості на пристроях з архітектурою x86 та розглянуто декілька можливих способів атаки.
Ключові слова: вразливості вбудованих систем, rowhammer, dram пам'ять, архітектура x86.

Костючко С.Н., Ольшевский О.В. Обзор аппаратных уязвимостей встроенных систем на примере Rowhammer. В статье рассмотрены уязвимости встроенных систем на примере Rowhammer. Объяснено суть уязвимости и способы ее использования для совершения противоправных действий. Также описаны особенности этой уязвимости на устройствах с архитектурой x86 и рассмотрено несколько возможных способов атаки.
Ключевые слова: уязвимости встроенных систем, rowhammer, dram память, архитектура x86.

Kostiuchko S., Olshevskyi O. Overview of hardware vulnerabilities of embedded systems on the example of Rowhammer. The article examines the vulnerabilities of embedded systems on the example of Rowhammer. The essence of the vulnerability and ways to use it to commit illegal acts are explained. Features of this vulnerability on devices with x86 architecture are also described and several possible attack methods are considered.
Keywords: embedded system vulnerabilities, rowhammer, dram memory, x86 architecture.

Постановка проблеми.

Вбудовані системи займають значну частку ринку технологічних пристроїв та обчислювальних систем у різних областях. Деякі з цих областей, наприклад, середовища критичної інфраструктури, потребують від таких пристроїв значної швидкодії та захисту від кібератак.

До вбудованих систем можна застосувати широкий спектр таких атак. Проти більшості з них вже є засоби протидії, що включають програми проти зловмисного програмного забезпечення, брандмауери, системи виявлення вторгнень та засоби виявлення аномалій у роботі обладнання. Проте, зловмисники все ж знаходять способи «взламати» вбудовані системи, оскільки в цих системах не передбачався надійний захист від кібератак. Нещодавно, у проекті H2020 CIPSEC (Покращення захисту критичної інфраструктури за допомогою інноваційної програми безпеки) вищезазначені інструменти розглядались як єдине ціле, таким чином структуруючи захист від кібератак, які застосовуються до багатьох різних критичних інфраструктурних середовищ, куди входять вбудовані системи[1]. Основна ідея полягає в тому, що окрім критичної інфраструктури, кінцеві пристрої, які в основному є вбудованими системами, також ідентифікуються як можлива мета зловмисника, що проводить кібератаку. Таким чином, пропонується посилення безпеки вбудованих системних кінцевих пристроїв як апаратними, так і програмними засобами.

Як і більшість обчислювальних пристроїв, вбудовані системи не можна вважати надійними через багато відомих вразливостей. Як правило, програмні засоби (наприклад, антивірус, антивірусне програмне забезпечення чи брандмауери) можуть захистити систему від зловмисників, що використовують ці уразливості для впровадження деякого шкідливого програмного забезпечення. Однак існують прогалини в безпеці та вразливості, які неможливо відстежити вищезазначеними програмними засобами, оскільки вони не розміщені на рівні програм, драйверів чи операційної системи, а скоріше на рівні архітектури комп'ютера та в самій структурі обладнання. Програмне забезпечення, що використовує ці вразливості залишається прихованим для більшості програмних засобів протидії кібератакам, тому становить серйозний ризик для безпеки всіх пристроїв, що мають такі вразливості.

Одним з найпотужніших засобів підвищення комп'ютерної безпеки без компромісів є міграція апаратних функцій безпеки. З цієї причини існує багато елементів апаратної безпеки разом із відповідними бібліотеками програмного забезпечення, що підвищують безпеку. Модулі платформи, що створюють надійне середовище виконання, - це дуже надійні рішення для встановлення довіри, які забезпечують безпечне, ізольоване від потенційних атак виконання критичних додатків [2–4]. Це досягається шляхом створення

ізольованих областей пам'яті для виконання коду та зберігання конфіденційних даних, як це пропонується та реалізується технологією процесора ARM TrustZone [4]. Однак ці рішення, які поступово впроваджуються в критичних для безпеки областях (наприклад, у системах критичної інфраструктури), не можуть бути повністю захищені від апаратних вразливостей. Типові комп'ютерні апаратні компоненти, такі як модулі пам'яті та кеші, демонструють дивну поведінку за певних обставин, дозволяючи таким чином потенційним зловмисникам отримати доступ через бекдор. Помилки, що пов'язані з пам'яттю спостерігаються в мікросхемах DRAM, розміщених і використовуваних у всіх пристроях критичної інфраструктури. Було виявлено, що при повторному зверненні до певного рядка банку пам'яті DDR (подвійна швидкість передачі даних), тобто відкриття (активації) та закриття (деактивації) протягом інтервалу оновлення DRAM, один або кілька бітів змінюють своє значення у фізично суміжних рядках DRAM.

Ця вразливість DRAM стала відома як Rowhammer [5]. Кілька дослідників помітили, що Rowhammer може бути використаний для здійснення атаки та обходу більшості встановлених функцій безпеки та довіри програмного забезпечення, включаючи ізоляцію пам'яті, шляхом написання відповідного шкідливого програмного забезпечення, що використовує вразливість Rowhammer. Цю атаку можна використовувати для пошкодження системної пам'яті, аварійного завершення роботи системи, отримання та зміни секретних даних або захоплення всієї системи. Крім того, обчислення, пов'язані з безпекою, на вбудованому системному пристрої в реальному часі можуть передавати конфіденційну інформацію, що обробляється, до обізнаного зловмисника. Такі типи атак позначаються як атаки на бічні канали (SCA) та мають на меті збір даних про час роботи програми та за допомогою статистичних обчислень, можуть отримувати конфіденційну інформацію. SCA можуть бути дуже небезпечними для вбудованих системних пристроїв, які залишаються без спостереження протягом тривалого часу, бо зловмисник може мати фізичний доступ до них. Характерними прикладами таких пристроїв без нагляду є кінцеві вузли критичних систем інфраструктури (наприклад, датчики, в польових приводах, ПЛК (програмований логічний контролер) або інші пристрої моніторингу/управління). Однак, навіть якщо фізичний доступ до пристроїв неможливий, існують типи SCA, такі як мікроархітектурні/кешовані SCA, які можна реалізувати дистанційно.

Суть вразливості Row hammer

Прагнучи зменшити пам'ять у розмірах, зменшити ціну за біт та збільшити швидкодію, постачальники зменшили фізичну геометрію DRAM та збільшили щільність чіпу. Але менші комірки пам'яті можуть містити меншу, обмежену кількість заряду, що зменшує запас шуму і робить комірку більш уразливою до втрати даних. Крім того, висока щільність комірок вносить ефект електромагнітного зв'язку між ними. Крім того, багато варіацій технології виготовлення, збільшують кількість комірок пам'яті, сприйнятливих до перехресних перешкод. Тому нові технології DRAM частіше страждають від вразливостей та провокують помилки. Існування та поширеність таких помилок у реальних мікросхемах DRAM були вперше виявлені [5] у 2014 р. Тоді, корінь проблеми був виявлений у коливанні напруги на внутрішніх з'єднаннях. Кожен рядок комірок пам'яті має свою власну доріжку, по якій подається напруга. При виникненні багатьох звернень до одного рядка, ці звернення змушують лінію вмикатись та вимикатись, що неодноразово провокує коливання напруги, які створюють вплив на сусідні рядки. Заряд зі збуджених рядків витікає із прискореною швидкістю, і якщо дані не відновлюються досить швидко, деякі комірки змінюють своє початкове значення. Фактично, було виявлено, що ця вразливість існує у більшості останніх чіпів DRAM, оскільки вона більш поширена на технологіях пам'яті 40 нм.

У роботі Університету Карнегі -Меллона [5] було виявлено, що це явище помилки існує в 139 модулях DRAM на процесорах x86, для всіх модулів, що виготовлені з 2013 по 2014 рік. Лантейн здійснив аналіз пам'яті DDR4 [6], та довів, що безумовно, це явище можна відтворити за певною технологією. З 12 тестуваних модулів, 8 показали зміну бітів під час експериментів з частотою оновлення. Наслідки вразливості Rowhammer можна простежити у різноманітних аспектах роботи обладнання. Зміна напруги лінії пам'яті може спричинити шум у сусідній лінії за допомогою електромагнітного зв'язку та частково дозволити це витік напруги у осередках свого рядка. Перемикання лінії може прискорити потік заряду між двома осередками моста. Мости - це клас несправностей DRAM, при якому провідні канали утворюються між не пов'язаними проводами або конденсаторами. Ефект ін'єкції гарячого носія може назавжди пошкодити канал передачі заряду та може бути досягнутий шляхом перемикання протягом 100 годин тієї самої лінії, як доведено в [5].

Як скористатись вразливістю Rowhammer

Неодноразово звертаючись, використовуючи один і той же рядок пам'яті (рядок агресора), зловмисник може викликати достатню кількість збурень у сусідньому рядку (рядок жертви), щоб викликати

незначну зміну бітів. Це може бути програмне забезпечення, запущене за допомогою коду з циклом, який генерує мільйони читань до двох різних рядків DRAM одного банку в кожній ітерації. Можливість зробити активацію цього рядка достатньо швидкою - перша передумова для запуску атаки. Контролер пам'яті повинен дозволяти цей швидкий доступ, але, як правило, найбільшою проблемою є перевищення кількох шарів кешу, які маскують всю пам'ять ЦП (центрального процесора). Доступ до пам'яті поділяється на етапи. На активній стадії спочатку активується рядок, щоб перенести рядок даних у буфер рядків банку пам'яті, увімкнувши його відповідну лінію слів. Після цього, конкретний стовпець із рядка зчитується/записується (етап READ/WRITE) з або в буфер рядків. Нарешті, рядок закривається, попередньо заряджаючи (етап PRECHARGE) конкретний банк пам'яті, записуючи значення в рядок і вимикаючи лінію. Помилка збурень виникає в рядку DRAM, коли напруга сусідньої лінії слова неодноразово перемикається, а це означає, що вона виникає при повторному АКТИВНОМУ/PRECHARGE рядках, а не на етапі стовпця ЧИТАННЯ/ЗАПИС. Коли програмний код запускає помилку Rowhammer шляхом повторного доступу до тієї ж адреси фізичної DRAM, необхідно бути обережним, щоб кожен доступ відповідав активації нового рядка. Якщо безперервно звертатися до однієї фізичної адреси, то відповідні дані вже знаходяться в буфері рядків, і нова активація не виробляється. Тому потрібно отримати доступ до двох фізичних адрес, які відповідають рядкам на одному банку, щоб переконатися, що буфер рядків очищається між доступом до пам'яті. Другий виклик, який слід розглянути, – це необхідність пошуку фізичних адрес, які відображаються у рядках з того самого банку.

Ранні проекти протистояли цим викликам, вибираючи випадкові адреси, дотримуючись ймовірнісних підходів [5,9–11]. В останніх роботах вказано, що слід знати точне відображення фізичних адрес усіх банків, щоб отримати доступ до обох рядків, розташованих безпосередньо над ними а під жертвою – для двосторонньої атаки Row hammer. Така атака є набагато ефективнішою. Можливість викликати помилки в певних системах залежить від частоти перемикання, що в свою чергу, залежить від інтервалу оновлення. Це відноситься до того, скільки часу певний рядок залишається активним між командами оновлення, які поповнюють дані кожної комірки. Як правило, чим більше разів рядок відкривається на інтервалі оновлення, тим більша ймовірність зміни бітів. Тим не менше, коли швидкість доступу до пам'яті перевищує обмеження, менша кількість бітів змінюється, ніж у звичайному випадку [7]. При достатньо високому інтервалі оновлення помилки можна було б повністю усунути, але з відповідним наслідком як для споживання електроенергії, так і для продуктивності [5,7,13]. Залежно від реалізації DRAM, деякі комірки пам'яті представляють логічне значення "1", використовуючи стан заряду, тоді як інші комірки представляють логічне значення "1", використовуючи розряджений стан. Бувають випадки, коли обидві орієнтації використовуються в окремих частинах одного модуля DRAM. Для запуску процесу зміни бітів, комірки повинні бути заряджені, щоб збільшити швидкість їх розряду та викликати row hammer рядок. Доведено, що ймовірність перевертання бітів у певному рядку залежить від даних, які зберігаються у цьому рядку, а також від орієнтації конкретних комірок рядка [5]. Якщо орієнтація модулів не відома до атаки, необхідно провести інший тест шаблону даних, щоб перевірити вразливість Rowhammer на конкретній системі [5,6,14].

Особливості Rowhammer на архітектурі x86

З моменту виявлення, були реалізовані різні методи експлуатації, що відтворюють Rowhammer з метою обходу всіх поточних засобів захисту систем на базі архітектури x86.

Архітектура набору інструкцій x86 була реалізована в процесорах Intel, Cirix, AMD, VIA та багатьох інших компаній і використовується в декількох нових поколіннях вбудованих систем. Атаки Rowhammer проти таких пристроїв, спрямовані безпосередньо на доступ до пам'яті, а не на саму ОС, важко виявити за допомогою традиційних програмних засобів, таких як антивірусне програмне забезпечення або продукти брандмауера. Потенційна різноманітність атаки може призвести до широкого кола можливих проблем, включаючи несанкціонований доступ, відмову в наданні послуг або крадіжку ключів шифрування. Атаки Rowhammer також можуть використовуватися як бекдор для більш складних атак, які в іншому випадку були б виявлені і не виконані в архітектурах x86.

Кілька атак Rowhammer, на архітектурі x86, що спричиняють серйозні порушення безпеки, були описані в [12]. В одній з таких атак, через активацію уразливості Rowhammer, у програмі Native Client (NaCl) викликаються зміни бітів, через що досягається ескалація привілеїв, вихід з x86 середовтща NaCl та отримання можливості здійснювати системні виклики ОС.

В іншій атаці [10] фізична пам'ять заповнюється таблицями сторінок для одного процесу (функція nmap () використовується для багаторазового заповнення даними файлу). Такі дії заповнюють пам'ять записами таблиці сторінок (PTE), які використовуються для перекладу нещодавно створених віртуальних

адрес mmap() [12]. Використовуючи rowhammer в пам'яті, що містить таблиці сторінок, існує нетривіальна ймовірність того, що деяка PTE через зміну бітів буде змінена так, що вкаже на фізичну сторінку, яка містить сторінку таблиць. Це дасть застосунку, який ми використовуємо, доступ до власних таблиць сторінок, таким чином надаючи користувачеві програми (тобто зловмиснику) повний дозвіл на читання/запис на запис таблиці сторінок, дію, яка врешті-решт дає доступ всієї фізичної пам'яті [10,12].

Експлуатація крос-віртуальних атак rowhammer виявилася успішною на машинах, які знаходяться на стороні сервера, завдяки можливості вилучення приватного ключа з веб-сервера HTTPS та ін'єкції коду, що може обійти автентифікацію паролем на сервері OpenSSH (SSH: Secure Shell) [16]. Це призводить до серйозної загрози для хмарних сервісів, оскільки там користувачі можуть розміщувати свої віртуальні машини на одному фізичному сервері, та використовувати апаратні ресурси, включаючи DRAMS. Атаки використовують віртуальну машину з паравіртуалізацією, щоб порушити ізоляцію віртуальної машини та поставити під загрозу цілісність та конфіденційність спільно розташованих віртуальних машин. Вразливість ґрунтується на тому факті, що зіставлення між сторінкою псевдофізичної пам'яті та машинною пам'яттю для простору віртуальної машини користувача може бути обманутим, щоб вказати на сторінки пам'яті іншої віртуальної машини або навіть гіпервізора. Представлена методика, яка називається атакою заміни таблиці сторінок, проводить атаки rowhammer, щоб детерміновано змінити біти у записі каталогу сторінок (PDE), таким чином, зробивши покажчик на іншу таблицю сторінок.

Зловживаючи системою дедуплікації пам'яті Linux (KSM) і використовуючи вразливість rowhammer, зловмисник може змінювати біти на будь-якій фізичній сторінці стека програмного забезпечення [20]. Віртуальна машина може зловживати дедуплікацією пам'яті, щоб розмістити на цільовій контрольній фізичній сторінці розміщену на спільній веб-сторінці сторінку віртуальної машини, а потім використати Rowhammer для зміни певного біта на сторінці жертви без дозволу на запис цього біту. Через високу популярність KSM у хмарних сервісах, цей метод може мати руйнівні наслідки.

На скриптових мовах було реалізовано віддалене програмно-індуковане виконання Rowhammer. Атака виконується у пісочниці JavaScript, яка за замовчуванням включена у кожному сучасному веб-переглядачі, тому її можна запускати з будь-якого веб-сайту. Конкретна реалізація не залежить від набору інструкцій центрального процесора, і її основна проблема полягає у виконанні оптимальної стратегії вилучення кешу.

Ще дві різні моделі потоків пропоновані [11] на основі методу тимчасового зберігання архітектури x86. Вони дуже зручні, оскільки їх можна використовувати для реалізації атак за допомогою широко використовуваних функцій memset та memcpy з бібліотеки libc, тим самим надаючи практично кожному коду потенціал Rowhammer. Одна з моделей може бути використана для реалізації коду з ненадійного джерела та масштабування пісочниці NaCl. Друга модель може бути використана для запуску Rowhammer з наявним доброякісним кодом, який не впливає на безпеку, наприклад, мультимедійні програвачі або зчитувачі PDF файлів шляхом компрометації вхідних даних.

Згадані вище підходи, як скриптові так і компільовані мови, орієнтовані на активацію Rowhammer у віддалених системах і дають можливість для отримання кореневих прав за допомогою віддаленої атаки та отримання доступу до всієї фізичної пам'яті цільової системи. Обидва підходи можуть обійти основні заходи боротьби з цією вразливістю, оскільки вони оминають інструкцію flush(заборонено в деяких останніх версіях ОС).

Висновки.

У даній статті було розглянуто вразливість Rowhammer, одну з багатьох, які можуть бути використані при атаці на вбудовані системи через апаратні уразливості. Ця атака використовує програмний код, який безпосередньо не заважає іншим програмним засобам (заражає програми, безпосередньо змінює їх функціональні можливості), а лише експлуатує внутрішні вразливості системи. Аналіз атаки, проведений у цій статті, показує, що вона здатна суттєво знизити рівень безпеки вбудованої системи, а також залишитись непоміченою для багатьох технологій, що підвищують безпеку, включаючи програмні засоби протидії кібератакам (антивірусне програмне забезпечення, виявлення аномалій в роботі, брандмауери), а також апаратні, такі як ізоляція пам'яті через віртуальні машини, надійні середовища виконання, ARM TrustZone або виділені елементи безпеки. Архітектор системи безпеки вбудованої системи в режимі реального часу повинен бути поінформований про можливість таких атак і повинен чинити допомагати виробникам вбудованих систем, щоб вони надавали патчі безпеки для своїх продуктів, які здатні перешкоджати атакам.

References.

1. Enhancing Critical Infrastructure Protection with Innovative SECurity Framework (CIPSEC). H2020 European Project. Available online: www.cipsec.eu (accessed on 28 March 2017).
2. Challener, D.; Yoder, K.; Catherman, R.; Safford, D.; Van Doorn, L. A Practical Guide to Trusted Computing; IBM Press: Indianapolis, IN, USA, 2007.
3. Trusted Computing Group. TCG TPM Specification Version 2.0; Trusted Computing Group: Beaverton, OR, USA, 2014.
4. ARM. ARMTrustZone. Available online: https://www.arm.com/products/security-on-arm/trustzone (accessed on 1 April 2017).
5. Kim, Y.R.; Daly, J.; Kim, C.; Fallin, J.; Lee, H.; Lee, D.; Wilkerson, C.; Lai, K.; Mutlu, O. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In Proceedings of the ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), Minneapolis, MN, USA, 14–18 June 2014
6. Lanteigne, M. How Rowhammer Could Be Used to Exploit Weakness in Computer Hardware. 2016. Available online: https://www.thirdio.com/rowhammer.pdf (accessed on 1 April 2017).
7. Van der Veen, V.; Fratantonio, Y.; Lindorfer, M.; Gruss, D.; Maurice, C.; Vigna, G.; Bos, H.; Razavi, K.; Giuffrida, C. Drammer: Deterministic rowhammer attacks on mobile platforms. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16), Vienna, Austria, 24–28 October 2016.
8. Pessl, P.; Gruss, D.; Maurice, C.; Schwarz, M.; Mangard, S. DRAMA: Exploiting DRAM addressing for Cross-CPU attacks. In Proceedings of the USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016.
9. Bosman, E.; Razavi, K.; Bos, H.; Giuffrida, C. Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector. In Proceedings of the 2016 IEEE Symposium on Security Privacy, SP 2016, San Jose, MA, USA, 23–25 May 2016; pp. 987–1004.
10. Seaborn, M.; Dullien, T. Exploting the DRAM rowhammer bug to gain kernel privileges. In Proceedings of the 2016 ACM SIGSAC Conference, Vienna, Austria, 24–28 October 2016.
11. Qiao, R.; Seaborn, M. A new approach for rowhammer attacks. In Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 3–5 May 2016.
12. Aweke, Z.B.; Yitbarek, S.F.; Qiao, R.; Das, R.; Hicks, M.; Oren, Y.; Austin, T. ANVIL: Software-based protection against next-generation rowhammer attacks. In Proceedings of the 21st ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Atlanta, GA, USA, 2–6 April 2016
13. Moinuddin, K.Q.; Dae-Hyun, K.; Samira, K.; Prashant, J.N.; Onur, M. AVATAR: A variable-retention-time (vrt) aware refresh for dram systems. In Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, 22–25 June 2015
14. Program for Testing for the DRAM Rowhammer Problem. 2015. Available online: https://github.com/ google/rowhammer-test (accessed on 15 March 2017).
15. Seaborns, M. How Physical Adressesses Map to Rows and Banks in DRAM. 2015. Available online: http: //lackingrhoticity.blogspot.com/2015/05/how-physical-adresses-map-to-rows-and-banks.html (accessed on 5 April 2017).
16. Xiao, Y.; Zhang, X.; Teodorescu, R. One bit flips, one cloud flops: Cross-VM row hammer attacks and privilage escalation. In Proceedings of the 25th USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016.
17. Gruss, D.; Maurice, C.; Mangard, S. Rowhammer.js: A remote software-induced fault attack in javascript. In Proceedings of the 13th Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA), Donostia-San Sebastián, Spain, 7–8 July 2016
18. Bhattacharya, S.; Mukhopadhyay, D. Curious Case of Rowhammer: Flipping Secret Exponent Bits Using Timing Analysis. Lect. Notes Comput. Sci. 2016, 9813, 602–624
19. Salyzyn, M. AOSP Commit 0549ddb9: UPSTREAM: Pagemap: Do Not Leak Physiscal Addresses to Non-Privilahe Userspace. 2015. Available online: http://goo.gl/Qye2MN (accessed on 1 May 2017).
20. Razai, K.; Gras, B.; Bosman, E.; Preneel, B.; Giuffrida, C.; Bos, H. Flip feng shui: Hammering a needle in the software stack. In Proceedings of the 25th USENIX Security Sympoisium, Austin, TX, USA, 10–12 August 2016.