УДК 004.056:004.4

АНАЛІЗ МЕТОДІВ ЗАХИСТУ ВІД КОПІЮВАННЯ ПРОШИВКИ В СУЧАСНИХ 32-БІТНИХ МІКРОКОНТРОЛЕРАХ

Рябінін Д. В.
email: denys.riabinin@nure.ua
Науковий керівник – к.н.т., доцент кафедри автоматизації проектування обчислювальної техніки Філіппенко Інна Вікторівна
Харьківський Національний Університет Радіоелектроніки
м. Харків, Україна

This paper provides a brief analysis of modern firmware copy protection methods used in 32-bit microcontrollers. The study highlights that safeguarding intellectual property is a critical step in commercial development due to the high cost of software engineering. Basic security measures, such as hardware Readout Protection (RDP) and the deactivation of debug interfaces like JTAG or SWD, are examined alongside their potential vulnerabilities, including hardware exploits and service backdoors. Furthermore, the paper discusses advanced techniques like on-the-fly memory encryption and the associated risks of bootloader vulnerabilities. Ultimately, the analysis concludes that absolute security is unattainable. Instead of providing an unbreakable defense, the primary goal of these protection mechanisms is simply to make the reverse engineering process economically unviable for competitors.

У сучасному світі мікроконтролери використовуються майже скрізь, від найпростіших побутових приладів до складних систем промислової автоматизації. Оскільки розробка програмного забезпечення вимагає значних витрат часу та зусиль цілої команди інженерів, питання захисту інтелектуальної власності стає дуже гострим. Особливо це стосується 32-бітних мікроконтролерів, які зараз є стандартом де-факто в індустрії і мають достатньо об'єму пам'яті для зберігання складних і дорогих алгоритмів. Тому розробники мікросхем постійно змушені шукати нові способи, як ускладнити життя тим, хто намагається несанкціоновано вкрасти прошивку з готового пристрою. Захист прошивки є критично важливим етапом випуску будь-якого комерційного продукту на ринок.

Одним із найпоширеніших і базових методів є апаратний захист від зчитування пам'яті, який в технічній документації найчастіше називають RDP (Readout Protection). Суть цього методу полягає в тому, що після завантаження прошивки в контролер на заводі, спеціальний конфігураційний біт перемикається в певний стан, який на апаратному рівні забороняє зовнішній доступ до флеш-пам'яті. Якщо хтось спробує підключитися звичайним програматором і зчитати машинний код, він отримає або суцільні нулі, або випадкове сміття, або мікроконтролер взагалі зависне і перестане відповідати на команди. Існує кілька рівнів такого захисту, і перший рівень зазвичай дозволяє зняти захист для перепрошивки, але при цьому вся внутрішня пам'ять повністю стирається. Це робить неможливим копіювання чужого коду, хоча сам чип можна використовувати повторно в інших проєктах. Хоча насправді в інтернеті повно інформації про те, як існують методи обходу цього захисту через апаратні вразливості старих ревізій чипів, коли, наприклад, живлення дуже

різко переривається і мікроконтролер просто не встигає перевірити стан цього захисного біта і віддає шматок пам'яті.

Також дуже важливою частиною захисту є правильне поводження з інтерфейсами відладки під час переходу до серійного виробництва. Розробники завжди залишають апаратні інтерфейси типу JTAG або SWD увімкненими для того, щоб було зручно шукати помилки в коді і дивитися стан регістрів під час створення пристрою. Але в готовому виробі ці піни обов'язково треба програмно відключати ще на етапі ініціалізації периферії, або перепризначати на звичайні порти вводу-виводу для світлодіодів чи кнопок. Якщо програміст забуде це зробити, зловмисник може просто підключитися до працюючого пристрою на платі, зупинити виконання ядра процесора і спокійно викачати весь вміст оперативної пам'яті. З іншого боку, якщо повністю і назавжди заблокувати відладку на апаратному рівні, то унеможливлюється будь-яка глибока діагностика пристрою в сервісному центрі у разі серйозної поломки. Тому іноді інженери залишають спеціальні програмні бекдори для доступу сервісників, що насправді повністю множить на нуль усю безпеку системи, бо ці бекдори часто знаходять через реверс-інжиніринг супутніх програм на комп'ютері.

Більш сучасні та дорогі методи захисту включають шифрування самої прошивки прямо всередині мікроконтролера. Код зберігається у флеш-пам'яті у зашифрованому вигляді за стандартами типу AES, і розшифровується спеціальним апаратним криптографічним блоком буквально "на льоту" безпосередньо перед тим, як інструкція потрапить у конвеєр ядра для виконання. Це звучить як ідеальний захист, тому що навіть якщо хтось фізично розпиляє пластиковий корпус мікросхеми кислотою і спробує зчитати стан комірок транзисторів під електронним мікроскопом, він побачить лише випадковий набір даних. Ключ шифрування при цьому лежить в окремій дуже захищеній області пам'яті, яка не читається. Але тут виникає величезна проблема з оновленням прошивки пристрою по повітрю або через флешку, бо треба якось безпечно передавати нові ключі і саму зашифровану прошивку. Це вимагає написання дуже складного кастомного завантажувача (bootloader). А сам завантажувач зазвичай пишеться людьми і може мати дірки в безпеці або банальні баги переповнення буфера, через які хакери можуть змусити контролер виконати їхній власний код і просто вивести оригінальну розшифровану прошивку через будь-який доступний порт типу UART.

Список використаних джерел:

1. Коваленко О. О., Петренко І. В. Аналіз методів апаратного захисту інформації у вбудованих системах. Захист інформації. 2024. Т. 26, № 2. С. 45–52.

2. Харченко В. С., Скляр В. В. Надійність та безпека мікропроцесорних систем. Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «ХАІ», 2023. 320 с.

3. Yiu J. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. 3rd ed. Amsterdam : Newnes, 2014. 600 p.

4. STMicroelectronics. STM32F4xx advanced Arm-based 32-bit MCUs Reference manual (RM0090). 2025. URL: https://www.st.com/resource/en/reference_manual/rm0090.pdf (дата звернення: 09.03.2026).