1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Изучение безопасности UEFI BIOS на базе платформ Intel**

по дисциплине «Принципы построения, проектирования и эксплуатации автоматизированных информационных систем»

1. Выполнила
2. студентка гр. 4851003/90801 Кулеева А.Г.

1. Руководитель
2. старший преподаватель Чернов А.Ю.
3. Санкт-Петербург
4. 2023

# **Цель**

Ознакомление с базовой архитектурой UEFI BIOS и их механизмами защиты.

# Формулировка задания

В рамках лабораторной требуется выполнить следующие действия и описать в отчете подробно все дальнейшие выполняемые действия со скриншотами:

1. Изучить процесс загрузки ЭВМ на базе UEFI BIOS в соответствии со спецификацией <https://uefi.org/sites/default/files/resources/UEFI_Spec_2_8_final.pdf>
2. Получить образ UEFI BIOS вашего ПК/Ноутбука c официального сайта производителя.
3. С использованием утилиты UEFITool открыть образ UEFI BIOS <https://github.com/LongSoft/UEFITool>
4. Определить модульный состав и точку входа для всех фаз функционирования UEFI BIOS.
5. С помощью утилиты **CHIPSEC** <https://github.com/chipsec/chipsec> определить конфигурацию LPC/SPI-устройства
6. Определить поддержку технологий Intel Boot Guard, Intel BIOS Guard, Secure Boot и их конфигурацию. Сделать предположение о корректности заданной производителем конфигурации данных технологий.
7. В модульном составе UEFI BIOS выявить модуль Setup
8. С помощью утилиты **Universal-IFR-Extractor** <https://github.com/LongSoft/Universal-IFR-Extractor> извлечь IFR-информацию из Setup-модуля.
9. Выявить скрытые опции настройки UEFI BIOS через NVRAM-переменные
10. Сделать вывод о безопасности UEFI BIOS на исследуемой платформе.

Состав отчета

1. Электронный вид, печатаем только титул
2. Формулировка задания
3. Описание исследуемой платформы
4. Описание характеристик UEFI BIOS (версия, размер, производитель, состав и назначение регионов)
5. Подробно со скриншотами описание проделанной работы.

# Ход работы

Данная лабораторная работа выполнялась на ноутбуке со следующими характеристиками (Таблица 1):

Таблица 1 — Характеристики изучаемой системы

|  |  |
| --- | --- |
| Название модели | Acer Aspire A5 |
| Процессор | |
| Название модели | Intel(R) Core(TM) i3-6006U |
| Частота | 2.00GHz |
| Кол-во ядер | 2 |
| Кол-во логических процессоров | 4 |
| BIOS | |
| Режим | UEFI |
| Поставщик | Insyde Corp. |
| Версия | 1.15 (дата 01/08/2018) |
| Память | |
| ROM | 8192 KB |
| RAM | 8,00 ГБ |

## Изучение процесса загрузки ЭВМ на базе UEFI BIOS

После инициации включения платформы блок питания ждет, пока не завершатся переходные процессы, и после устанавливает сигнал на линию **Power\_Good**. И первым начинает работу не центральный процессор, а автономная подсистема **Intel Management Engine (ME)** или аналогичная ей **AMD Secure Technology (ST)**. Эта подсистема проводит собственные операции, а затем подготавливает и запускает первое ядро одного процессора, именуемое **Bootstrap Processor (BSP)**.

Как и в Legacy, процессор начинает выполнять первую инструкцию в конце адресного пространства по адресу **0xFFFFFFF0**. Эта инструкция — прыжок на первую фазу инициализации платформы — SEC.

### Фаза SEC (Security)

В данной фазе должны быть решены следующие задачи:

* обработка события включения;
* инициализация достаточного количества памяти для следующей фазы;
* становление корня доверия системы;
* передача необходимой информации и управления на следующую фазу.

Фаза SEC состоит из следующих этапов:

* Reset Vector: сброс кэша и переход на главную процедуру инициализации в ROM.
* Switch to protected mode: переключение в защищенный режим процессора с плоской памятью без подкачки. Процессоры x86\_64 запускаются в **16-битном реальном режиме**, и в процессе первичной инициализации BSP переводится в **32-битный защищенный режим**.
* Initialize MTRRs for BSP: запись в кэш известных значений для различных областей памяти.
* Microcode Patch Update: обновление микрокода всех доступных процессоров.
* Initialize NEM: свободный кэш помечается как несбрасываемый, после чего его можно использовать как временную RAM до инициализации основной. Такой режим называется **no-eviction mode (NEM)**. В выделенной памяти создается стек, что позволит модулям из следующих фаз использовать стековые языки программирования до инициализации основной оперативной памяти.
* Early BSP/AP interactions: далее происходит инициализация всех прикладных процессоров (Application Processor, AP) с отправкой им специальной последовательности межпроцессорных прерываний (Inter-Processor Interrupt, IPI). Последовательность Init IPI: Start-up IPI — пробуждает прикладной процессор и запускает на нем самотестирование — **Built-In Self-Test (BIST)**. Результаты тестирования записываются и передаются далее для анализа.
* Hand-off to PEI entry point: передача управления и данных в фазу PEI.

В конце фазы Security необходимо найти раздел **Boot Firmware Volume (BFV)**, на котором располагается исполняемый код следующей фазы, а также по возможности найти другие, неосновные, разделы с кодом (Firmware Volume, FV).

Чтобы оправдать название фазы Security и стать корнем доверия, во время выполнения этой фазы код, которому мы планируем передать управление, может быть проверен на отсутствие несанкционированных изменений и вредоносных частей программы.

В конце выполнения SEC собрана следующая информация:

* размер и адрес Boot Firmware Volume (BFV);
* размер и адреса других Firmware Volume (FV);
* размер и адрес временной оперативной памяти;
* размер и адрес стека.

После чего начинается следующий этап — Pre EFI Initialization.

### Фаза PEI (Pre EFI Initialization)

Задача фазы Pre EFI Initialization заключается в сборе информации о подключенных устройствах и подготовке минимально необходимого количества оборудования для запуска процесса полной инициализации.

По своей задумке фаза PEI должна быть легковесной, так как память процессорного кэша ограничена. Помимо этого, в фазе PEI может происходить восстановление после сбоя, поэтому есть потребность размещать код фазы PEI в более отказоустойчивом хранилище.

Данная фаза состоит из ядра, называемого PEI Foundation, и подключаемых модулей **PEI Module (PEIM)**. Центральной частью ядра является диспетчер модулей, **PEI Dispatcher**, который управляет порядком исполнения модулей, а также организует межмодульное взаимодействие (PEIM-to-PEIM Interface, PPI).

Фаза PEI происходит следующим образом:

* Establish use of «memory»: перенос данных из ROM в раннюю RAM. Отметим, что фаза SEC исполнялась с флэш-памяти на материнской плате, и только в начале PEI необходимый для этой фазы исполняемый код копируется во временную оперативную память.
* PEI Dispatcher: запуск модулей PEIM в порядке от не имеющих зависимостей до имеющих сложные зависимости. Архитектура фазы PEI позволяет разрабатывать собственные модули, которые могут передавать результаты своей деятельности в следующую фазу. Передача информации происходит через специальную структуру данных Hand-off Block (HOB).
* **CPU PEIM**: инициализация CPU.

Platform PEIM: инициализация северного (в т.ч. Memory Controller Hub) и южного (I/O Controller Hub) мостов, встроенных интерфейсов платформы (SMBus, Reset, и т.п.). Определение режима загрузки (обычный, Recovery, S3 Resume), используя данные, полученные в фазе SEC.

* Memory Initialization PEIM: инициализация основной RAM и перенос в нее данных из кэша. Процесс зависит от определенного на предыдущем шаге состояния системы, например, при S3 Resume тестирование памяти не выполняется, что сокращает время загрузки.
* Если система находится в S3 Resume, выполняется CPU PEIM for S3 Boot Script, выполняющий возврат всех процессоров в их сохраненное состояние.
* S3 Boot Script Executor: восстановление состояния других устройств.
* OS Resume Vector: переход к ОС.

**Состояние S3 (Suspend to RAM)** — это состояние сна, при котором процессор и часть чипсета отключаются с потерей контекста. При пробуждении из такого состояния процессор начинает выполнение как при обычном включении. Но вместо полной инициализации и прохождения всех тестов система ограничивается восстановлением состояния всех устройств.

При запуске из любого другого состояния управление передается в фазу Driver Execution Environment.

### Фаза DXE (Driver eXecution Environment)

Задача фазы Driver Execution Environment (DXE) сводится к инициализации оставшихся устройств. К моменту старта фазы DXE процессор и основная память уже готовы к работе, а на драйверы DXE не накладываются строгие ограничения по потребляемым ресурсам.

Аналогично PEI Foundation в данной фазе есть собственное ядро **DXE Foundation**. Ядро создает необходимые интерфейсы и загружает три вида DXE сервисов:

**UEFI Boot Services** — сервисы времени загрузки;

**UEFI Runtime Services** — сервисы времени исполнения;

**DXE Services** — специальные сервисы, необходимые ядру DXE.

После инициализации сервисов начинает работу **DXE Dispatcher**. Он находит и загружает DXE драйверы, которые, в свою очередь, завершают инициализацию оборудования.

Среди множества загружаемых драйверов на процессорах x86\_64 стоит уделить внимание драйверу System Management Mode Init (SMM Init). Этот драйвер подготавливает все для работы **режима системного управления (System Management Mode, SMM)**. SMM — это особый привилегированный режим, который позволяет приостановить выполнение текущего кода (в т.ч. операционную систему) и выполнить программу из защищенной области памяти **SMRAM** в собственном контексте.

Дополнительно отметим модуль **Compatibility Support Module (CSM)**, который обеспечивает совместимость с Legacy и позволяет загружать ОС без поддержки UEFI.

После инициализации всего оборудования наступает время выбора загрузочного устройства.

### Фаза BDS (Boot Device Select)

В фазе Boot Device Select реализуется политика загрузки приложений UEFI. Несмотря на то, что это отдельная фаза, все сервисы, включая диспетчера, созданные на фазе DXE, остаются доступны.

Цель фазы BDS сводится к выполнению следующих задач:

* инициализация консольных устройств;
* поиск устройств, с которых можно загрузиться;
* попытка загрузиться с найденных устройств в порядке приоритета.

Поиском загружаемых областей на устройствах занимается **Boot Manager**. На некоторых картах расширения, например, на сетевых картах и RAID-контроллерах, может находиться собственный «BIOS», называемый **Option ROM**, или **OpROM**. Содержимое OpROM устройств запускаются сразу после обнаружения, а после выполнения управление возвращается в Boot Manager.

Все разделы, на которых находятся загружаемые области, сохраняются в памяти менеджера загрузки и упорядочиваются в соответствии с порядком загрузки. Если ни одного приложения не нашлось, Boot Manager может вызвать диспетчера DXE, на случай если за время поисков диспетчер загрузил дополнительные драйвера и менеджеру загрузки могут «открыться» новые устройства.

## Анализ с помощью UEFI Tool

С помощью утилиты Chipsec был извлечен образ прошивки BIOS. Использованные команды:

python chipsec\_util.py spi dump rom.bin

python chipsec\_main.py -k chipsec.md

После чего указанный образ был загружен в UEFITool для первичного анализа и ознакомления со структурой прошивки. Из Рисунка 1 можно сделать вывод, что UEFI BIOS состоит из трех регионов: Descriptor; ME; BIOS.

Descriptor регион содержит в себе информацию о смещении на остальные регионы, информацию о моделях чипсетов поддерживаемых устройством и прав на чтения и запись в другие регионы и таблицы. ME регион содержит Intel Management Engine Firmware и ее настройки. ME — встроенная в компьютерные платформы подсистема, обеспечивающая аппаратно-программную поддержку различных технологий Intel. BIOS регион состоит из одного или нескольких EFI Firmware Volume. Firmware Volume — это логическое представление содержимого flash. В данном регионе хранятся модули и точки входа для фаз загрузки UEFI BIOS.

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок — Структура BIOS

Были определены следующие модули: SEC (Рисунок 2), PEI (Рисунок 3), DXE и SMM (Рисунок 4).

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок — SEC

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок — Часть найденных PEIM

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок — Часть найденных DXE и SMM

## Реверс инжиниринг UEFI BIOS

Далее был проведен анализ образа bios.bin с помощью утилиты IDAPro. В самом конце кода находится reset vector, который является начальной точкой выполнения программы (Рисунок 5).

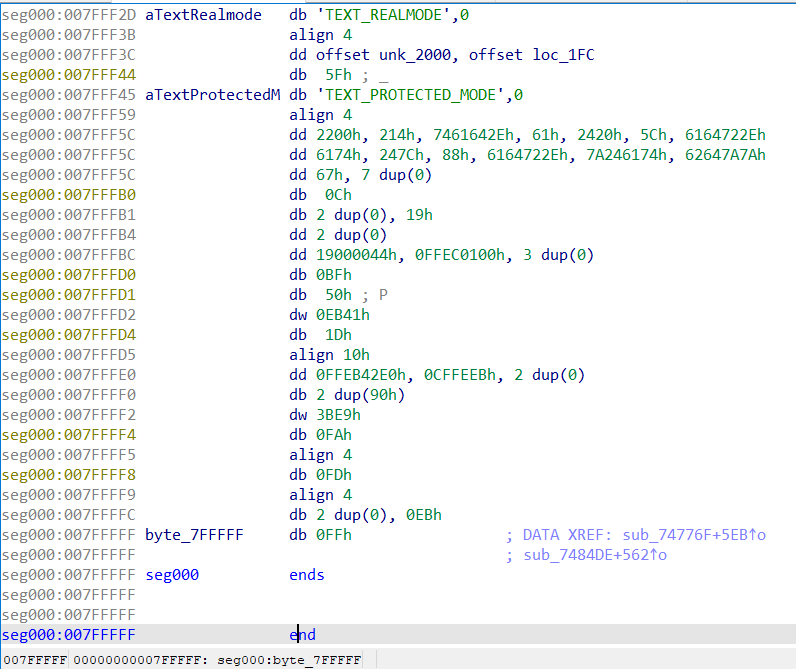


Рисунок — Reset vector

После прыжка по вектору происходит переход из реального режима, в защищённый (Рисунок 6).

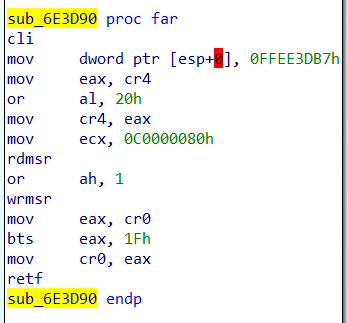


Рисунок — Переход в защищённый режим

Далее необходимо проинициализировать GDT (Рисунок 7).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок — Инициализация GDT

Чтобы найти точку входа в фазу SEC необходимо найти команду «call», вызывающую «SecStartup». Практическим путём от функции start был осуществлён поиск первых команд «call», одна из которых вызывала функцию, содержащую точку входа в фазу SEC (Рисунок 8).

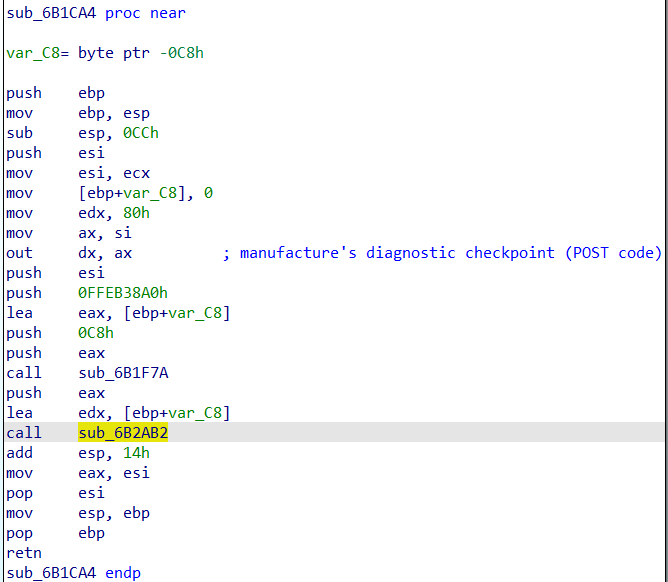


Рисунок — Функция, содержащая точку входа в фазу SEC

На основе найденной функции был получен её псевдокод на языке C (Рисунок 9). Функция принимает на вход 3 аргумента:

* SizeOfRam — размер RAM, доступной для использования;
* TempRamBase — адрес RAM;
* BootFirmwareVolume — адрес Boot Firmware Volume.



Рисунок — Код функции SecStartup

SecStartup возвращает результат функции SecStartupPhase2 (Рисунок 10), которая в свою очередь необходима для поиска точки входа в фазу PEI и перехода в данную фазу. Для поиска точки входа в PEI используется функция FindAndReportEntryPoints (Рисунок 11), PeiCoreEntryPoint значение представляет собой 64-разрядный виртуальный адрес (на Рисунке 10 обозначена v2).

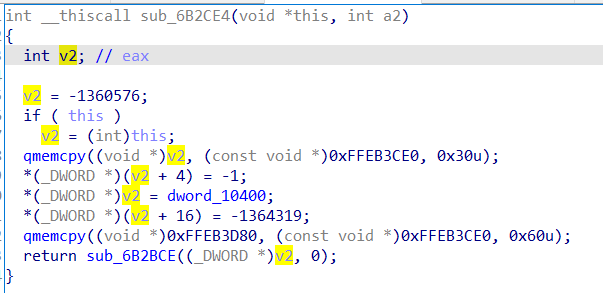


Рисунок — SecStartupPhase2

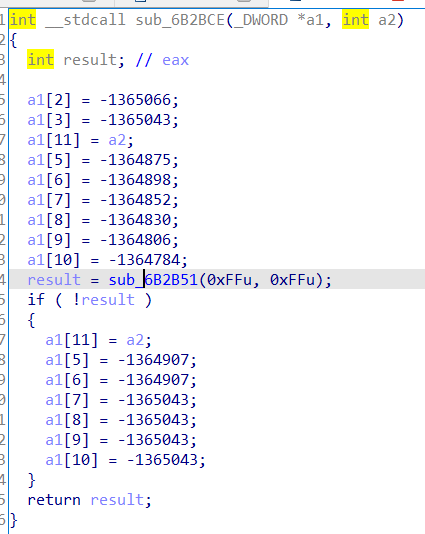


Рисунок — FindAndReportEntryPoints

Фаза PEI передаёт управление фазе DXE. Чтобы это осуществить, ядро — PEI Foundation использует специальный модуль DXE Initial Program Load (IPL) PPI для обнаружения и отправки DXE Foundation компонентов, необходимых для работы. Передача информации происходит через специальную структуру данных Hand-off Block (HOB). PEI Foundation находит и передаёт управление DXE IPL PPI, сканируя все PPI по GUID (Global Unic Identificator), чтобы найти тот GUID, который соответствует EFI\_DXE\_IPL\_PPI.

## Конфигурация LPC/SPI-устройств

Для анализа содержимого и конфигурации микросхемы BIOS была использована ОС Kali Linux LiveCD на флеш-накопителе. На данной системе была собрана по инструкции утилита из исходных файлов, также собран драйвер ядра, необходимый для работы chipsec.

Две основные утилиты в CHIPSEC — chipsec\_main и chipsec\_util. Утилита chipsec\_util содержит набор инструментов для взаимодействия с аппаратной платформой. Утилита chipsec\_main запускает тесты. Чтобы запустить стандартный набор тестов для оценки защищенности аппаратной платформы.

С помощью утилиты *chipsec\_util* (команда *spi info*) была получена информация о конфигурации SPI (Приложение 1). SPI — последовательный периферийный интерфейс для передачи данных в режиме полного дуплекса, предназначенный для обеспечения высокоскоростного сопряжения микроконтроллеров и периферии, в данном случае через него происходит взаимодействие с микросхемой BIOS. В полученном отчете о конфигурации есть сведения об архитектуре системы, о регионах флеш-памяти (SPI Flash Map, адреса), дескрипторах, связанных с компонентами (SPI Flash Descriptor), информация об опкодах (SPI Opcode Info), настройках доступа для BIOS (SPI Flash Protection).

С помощью утилиты chipsec\_main можно провести тестирование прошивки BIOS на безопасность. Каждый из отдельных тестов CHIPSEC создан на основании реальных атак или рекомендаций по безопасной разработке прошивки. Полный результат тестирования представлен в Приложении 2.

Ниже приведены основные тесты на безопасность, касающиеся SPI:

* common.spi\_access проверяет права доступа к регионам SPI Flash,
* common.spi\_desc cчитывает SPI Flash Descriptor, проверяет программную возможность его модификации,
* common.spi\_fdopss проверяет значение бита FDOPSS регистра HSFS контроллера SPI, сигнализирующего о перезаписи SPI Flash Descriptor,
* common.spi\_lock проверяет значение бита FLOCKDN регистра HSFS контроллера SPI. Если FLOCKDN не установлен, конфигурация SPI-контроллера, включая PR-регистры, может быть модифицирована злоумышленником.

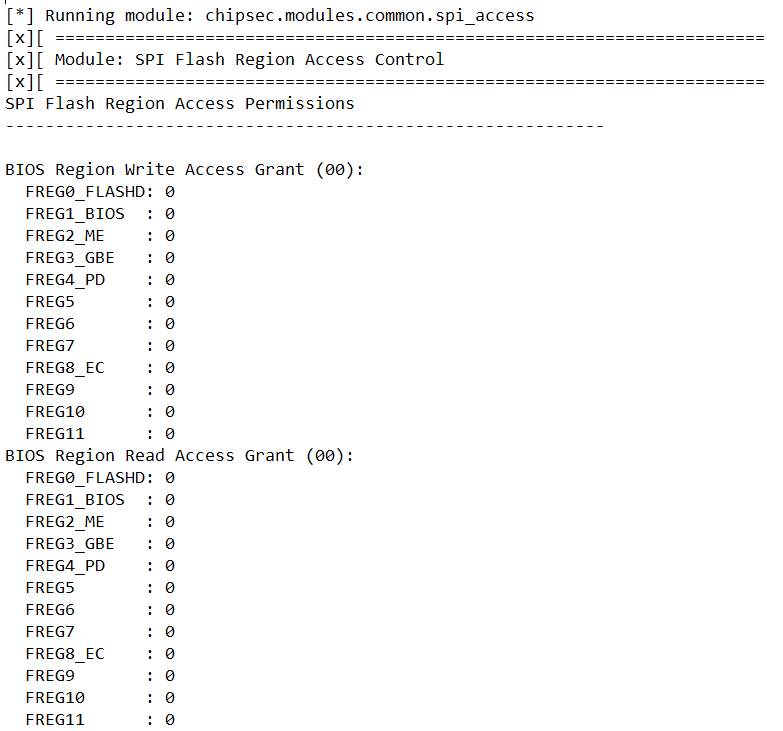
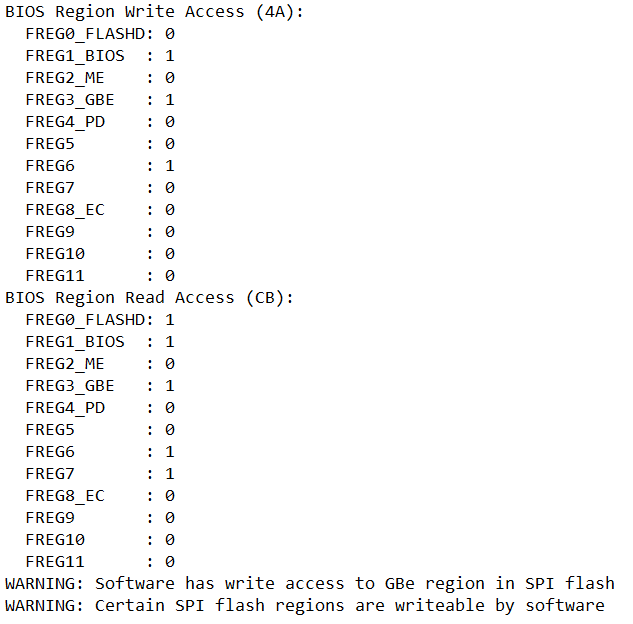
 

Рисунок — spi\_access

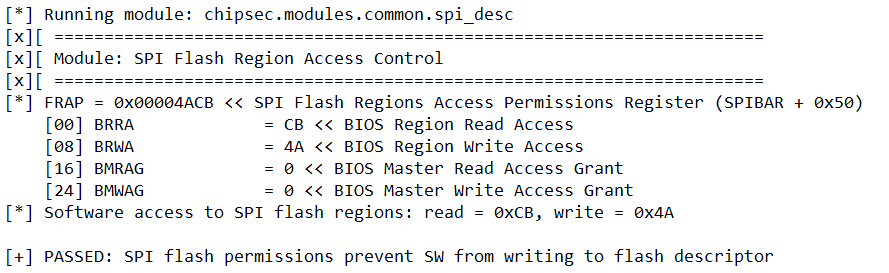


Рисунок — spi\_desc

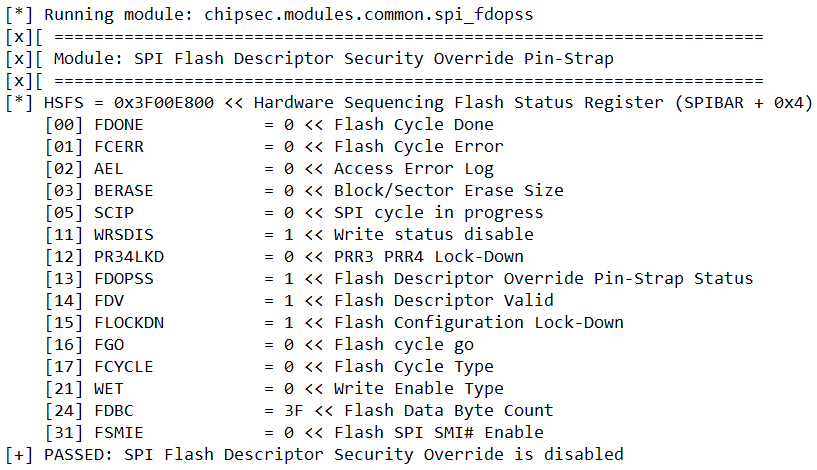


Рисунок — spi\_fdopss

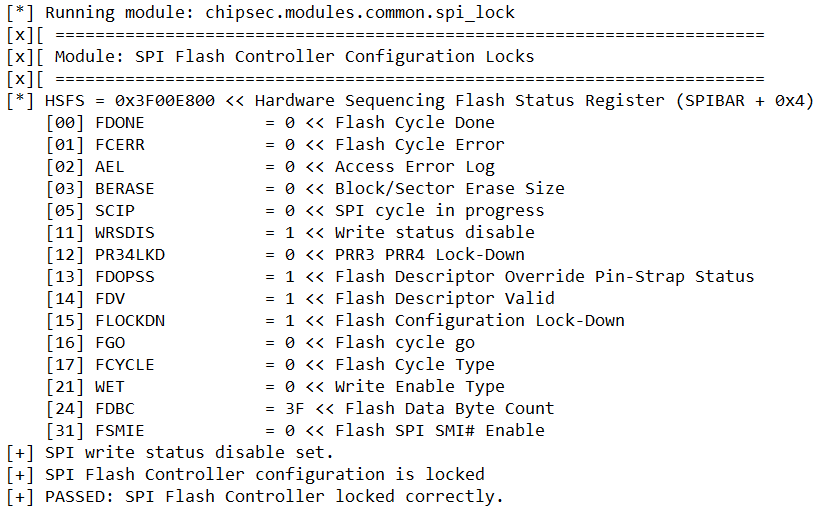


Рисунок — spi\_lock

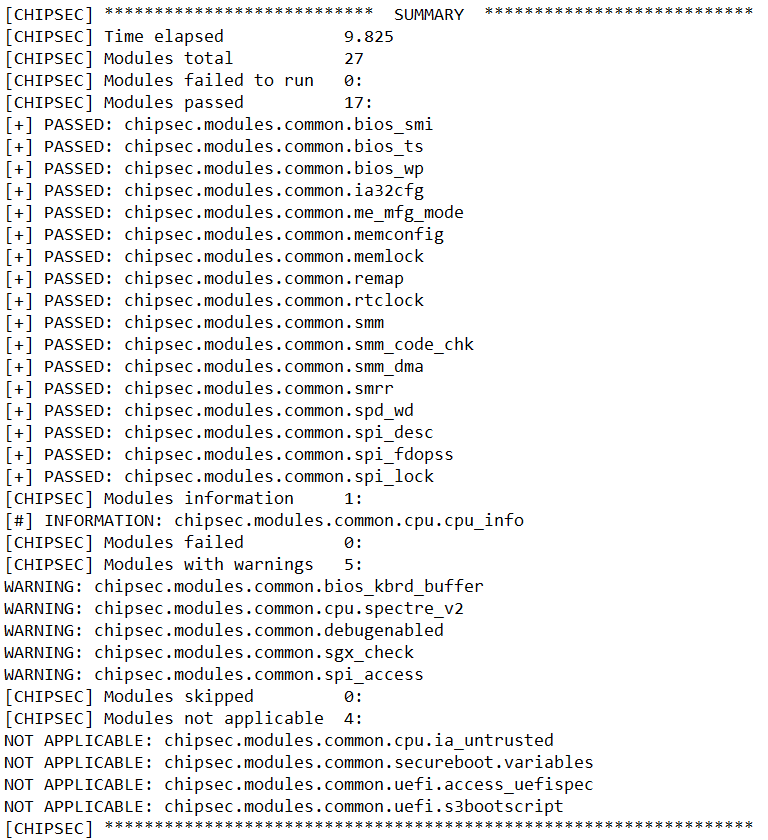


Рисунок — Общая оценка безопасности биос

## Intel Boot Guard, Intel BIOS Guard, Secure Boot

Информацию о Boot Guard можно получить из секции FIT UEFI tool. На Рисунке 17 отображено содержимое FIT таблицы. Видно, что отсутствуют локации ключей и политик Intel Boot Guard. С помощью Intel утилиты MEinfo была получена информация о конфигурации Intel Boot Guard. Утилита выводит информацию о ME регионе. В результате было выяснено, что Intel Boot Guard отключена (Рисунок 18).

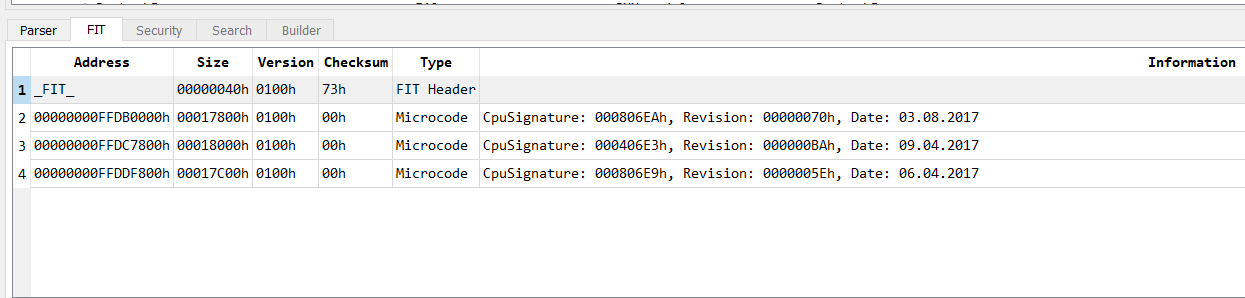


Рисунок 17 — Содержимое Firmware Interface Table



Рисунок — Данные MEInfo

С помощью функции поиска в утилите UEFI Tool определим наличие модуля Intel BIOS Guard Protection. На Рисунке 19 видно, что модуль присутствует в системе.

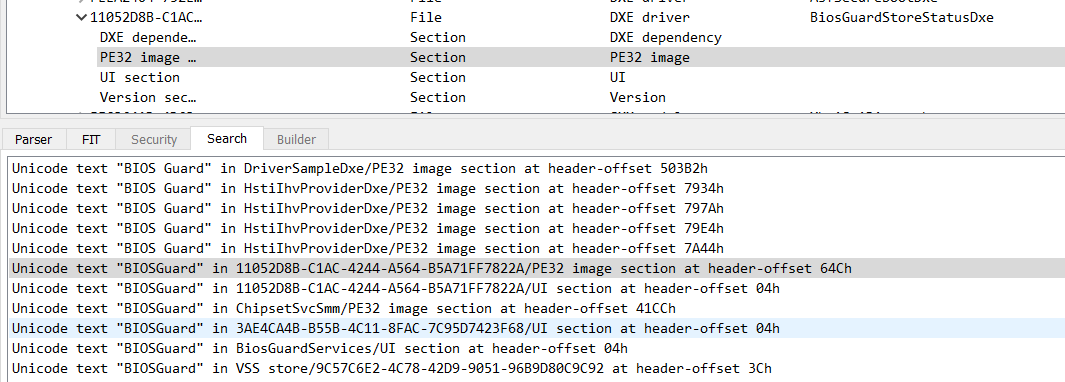


Рисунок — Поиск по ключевому слову BiosGuard в UEFITool

С помощью msinfo32 проверим состояние Secure Boot. На Рисунке 20 видно, что Secure Boot отключён. На самом деле это потому, что ноутбук работает в режиме Legacy, однако при настройке режима UEFI можно включить опцию защищенной загрузки.

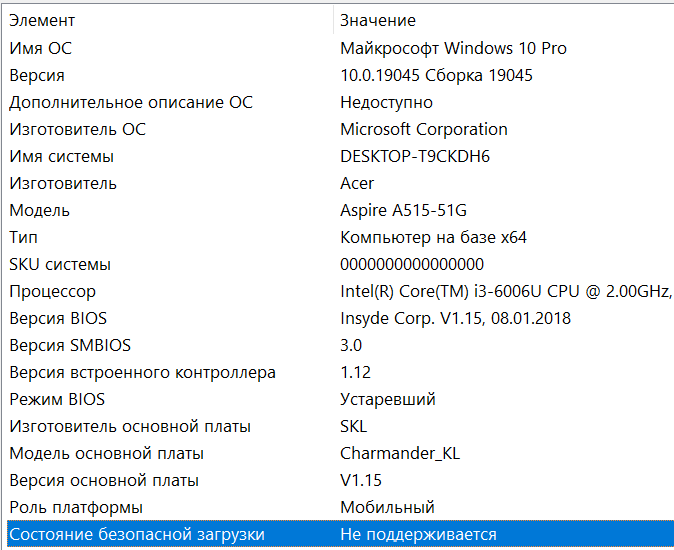


Рисунок — Данные msinfo32

## Модуль Setup

С помощью функции поиска в утилите UEFI Tool был найден модуль Setup (Рисунок 21).

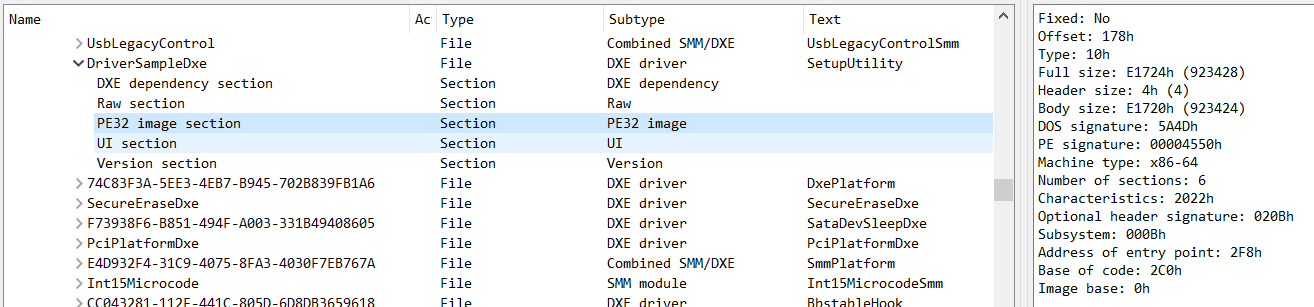


Рисунок — Модуль Setup

## Извлечение IFR-информации из Setup-модуля.

Internal Forms Representation (IFR) — это структуры данных, которые используются для описания моделей меню ввода. Также структуры данных определяют язык, который используется для описания разрешенного ввода пользователем. С использованием утилиты IFRExtractor было получено текстовое представление извлеченного бинарного файла модуля. Начало полученного файла, содержащее смещения NVRAM-переменных, представлено на Рисунке 22.

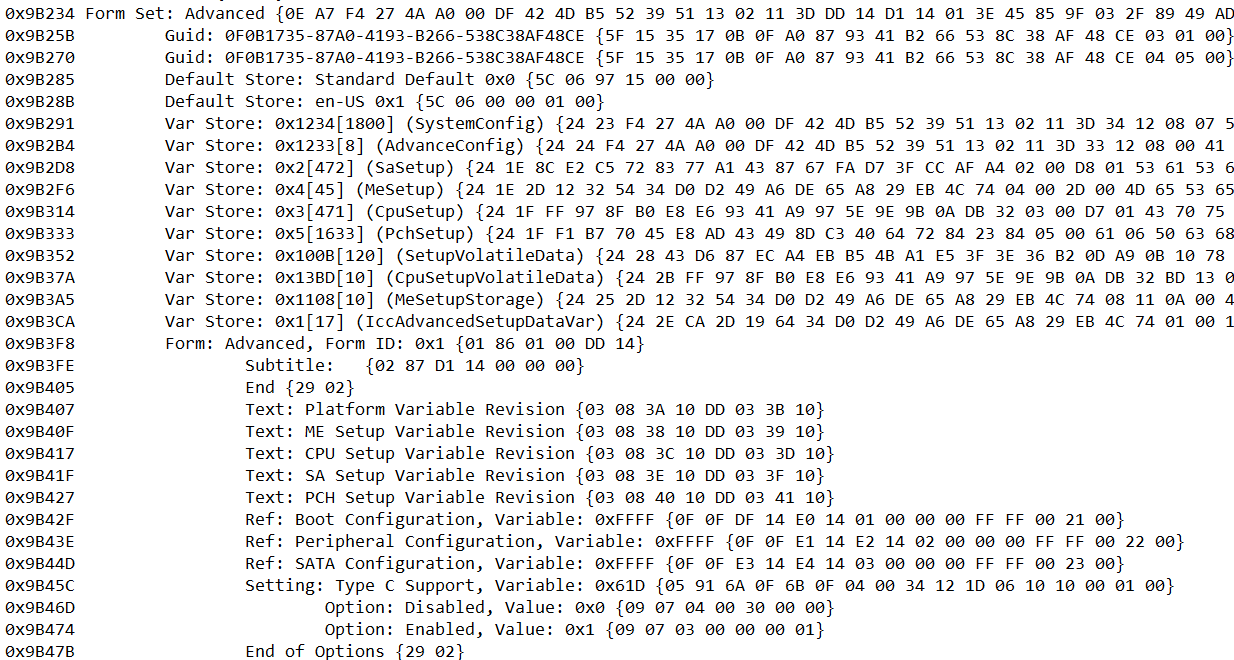


Рисунок — Часть переменных NVRAM

## Анализ UEFI BIOS через NVRAM-переменные

С помощью утилиты Chipsec были получены переменные NVRAM. С помощью них можно дополнительно проанализировать прошивку. На Рисунке 23 представлена переменная Setup.

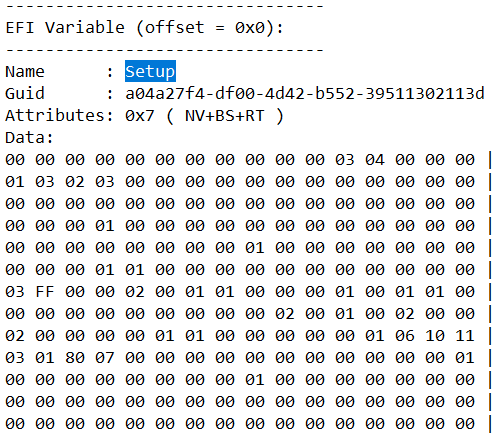


Рисунок — Setup переменная

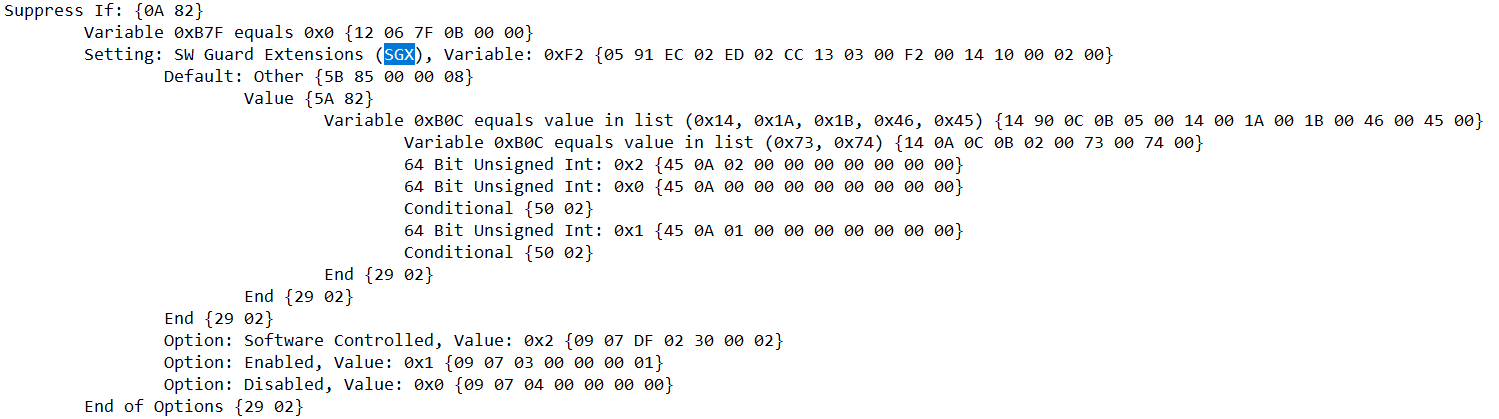


Рисунок 24 — Software Guard Extentions в IFR

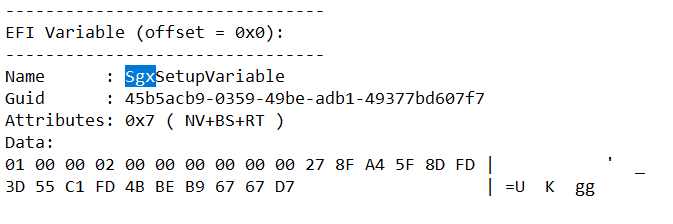


Рисунок 25 — Software Guard Extentions в Efi Variables

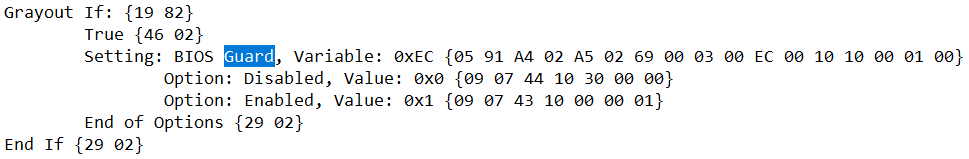


Рисунок — Bios Guard в IFR

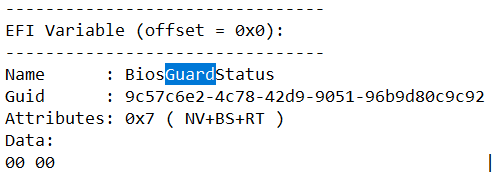


Рисунок 27 — Bios Guard в Efi Variables

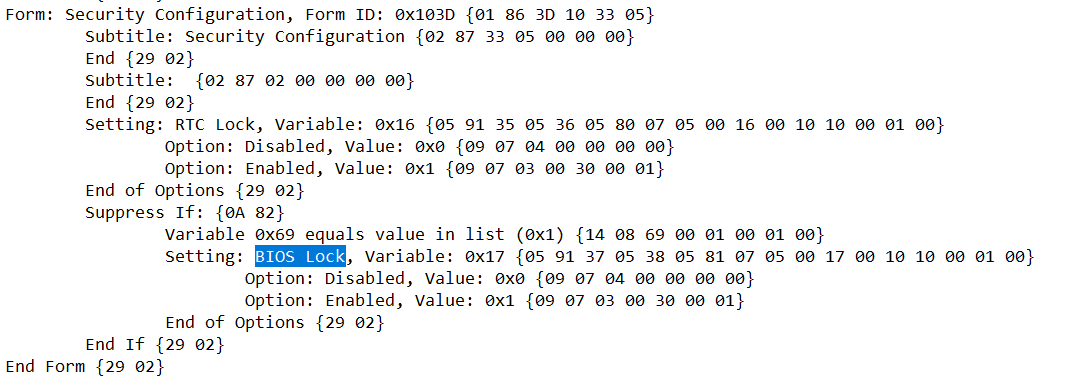


Рисунок — Bios Lock в IFR

# Вывод

В ходе выполнения лабораторной работы был изучен образ файла UEFI BIOS изучаемого устройства. С помощью утилит UEFITool и IDA PRO был произведен анализ исполняемого кода, в результате которого были изучены фазы процесса загрузки UEFI BIOS, а также найдены точки перехода между ними.

Кроме того, при помощи утилит CHIPSEC, MEInfo и Flash Image Tools был произведен анализ настроек безопасности изучаемого устройства. В результате проделанной работы был сделан вывод, что устройство защищено плохо, так как несколько технологий безопасности, такие как Secure Boot, Intel BIOS Guard и Intel Boot Guard не используются. Однако при этом используется средства аппаратной защиты такие, как регистры биоса.

Также с помощью утилиты IFR Extractor была получена IFR информация, проанализированная в дальнейшем относительно меню UEFI BIOS, выводимого пользователю. Значение всех NVRAM переменных, которые отвечают за настройку технологий безопасности, возможно изменить из операционной системы, что не является безопасным.

Полный анализ безопасности UEFI BIOS представлен в Таблице 1.

Таблица 1 — Результаты анализа безопасности UEFI BIOS

|  |  |
| --- | --- |
| Средство защиты | Реализация |
| SPI Protected Ranges | Защита регионов настроена |
| Secure boot | Технология реализована, но не настроена |
| Intel SGX | Технология реализована, но не настроена |
| Регистры BRRA, BRWA | Защита от записи в биос включена |
| Регистр FDOPSS | Бит установлен |
| Регистр FLOCKDN | Бит установлен |
| Boot Guard | Технология реализована, но не настроена |
| BIOS Guard | Технология реализована, но не настроена |
| Тесты CHIPSEC | Пройдено 17 из 27 |

# Приложение 1

################################################################

## ##

## CHIPSEC: Platform Hardware Security Assessment Framework ##

## ##

################################################################

[CHIPSEC] Version : 1.12.3

[CHIPSEC] Arguments: -l spi\_info.txt spi info

[CHIPSEC] OS : Linux 5.18.0-kali5-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 5.18.5-1kali6 (2022-07-07) x86\_64

[CHIPSEC] Python : 3.11.5 (64-bit)

[CHIPSEC] Helper : LinuxHelper (/home/kali/Desktop/chipsec/chipsec/helper/linux/chipsec.ko)

[CHIPSEC] Platform: Mobile 6th Generation Core Processor (Skylake U)

[CHIPSEC] CPUID: 406E3

[CHIPSEC] VID: 8086

[CHIPSEC] DID: 1904

[CHIPSEC] RID: 08

[CHIPSEC] PCH : PCH-U Premium

[CHIPSEC] VID: 8086

[CHIPSEC] DID: 9D48

[CHIPSEC] RID: 21

[CHIPSEC] Executing command 'spi' with args ['info']

[CHIPSEC] SPI flash memory information

============================================================

SPI Flash Map

------------------------------------------------------------

BIOS Flash Primary Region

------------------------------------------------------------

BFPREG = 07FF0200:

Base : 00200000

Limit : 007FFFFF

------------------------------------------------------------

Flash Region | FREGx Reg | Base | Limit

------------------------------------------------------------

0 Flash Descriptor | 00000000 | 00000000 | 00000FFF

1 BIOS | 07FF0200 | 00200000 | 007FFFFF

2 Intel ME | 01FF0001 | 00001000 | 001FFFFF

3 GBe | 00007FFF | 07FFF000 | 00000FFF

4 Platform Data | 00007FFF | 07FFF000 | 00000FFF

5 Flash Region 5 | 00007FFF | 07FFF000 | 00000FFF

6 Flash Region 6 | 00000000 | 00000000 | 00000000

7 Flash Region 7 | 00000000 | 00000000 | 00000000

8 Embedded Controller | 00000000 | 00000000 | 00000000

9 Flash Region 9 | 00000000 | 00000000 | 00000000

10 Flash Region 10 | 00000000 | 00000000 | 00000000

11 Flash Region 11 | 00000000 | 00000000 | 00000000

============================================================

SPI Flash Descriptor

------------------------------------------------------------

Flash Signature and Descriptor Map:

0FF0A55A

00040003

42100208

00310330

0FF0A55A

Components:

125C00F4

AD604221

C7C4B9B7

Regions:

00000000

07FF0200

01FF0001

00007FFF

00007FFF

Masters:

00A00B00

00C00D00

00800900

============================================================

SPI Opcode Info

------------------------------------------------------------

PREOP : 0x0000

OPTYPE: 0x0000

OPMENU: 0x0000000000000000

Prefix Opcode 0 = 0x00

Prefix Opcode 1 = 0x00

------------------------------------------------------------

Opcode # | Opcode | Optype | Description

------------------------------------------------------------

Opcode0 | 0x00 | 0 | SPI read cycle without address

Opcode1 | 0x00 | 0 | SPI read cycle without address

Opcode2 | 0x00 | 0 | SPI read cycle without address

Opcode3 | 0x00 | 0 | SPI read cycle without address

Opcode4 | 0x00 | 0 | SPI read cycle without address

Opcode5 | 0x00 | 0 | SPI read cycle without address

Opcode6 | 0x00 | 0 | SPI read cycle without address

Opcode7 | 0x00 | 0 | SPI read cycle without address

============================================================

SPI Flash Protection

------------------------------------------------------------

SPI Flash Region Access Permissions

------------------------------------------------------------

BIOS Region Write Access Grant (00):

FREG0\_FLASHD: 0

FREG1\_BIOS : 0

FREG2\_ME : 0

FREG3\_GBE : 0

FREG4\_PD : 0

FREG5 : 0

FREG6 : 0

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Read Access Grant (00):

FREG0\_FLASHD: 0

FREG1\_BIOS : 0

FREG2\_ME : 0

FREG3\_GBE : 0

FREG4\_PD : 0

FREG5 : 0

FREG6 : 0

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Write Access (4A):

FREG0\_FLASHD: 0

FREG1\_BIOS : 1

FREG2\_ME : 0

FREG3\_GBE : 1

FREG4\_PD : 0

FREG5 : 0

FREG6 : 1

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Read Access (CB):

FREG0\_FLASHD: 1

FREG1\_BIOS : 1

FREG2\_ME : 0

FREG3\_GBE : 1

FREG4\_PD : 0

FREG5 : 0

FREG6 : 1

FREG7 : 1

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Write Protection

------------------------------------------------------------

[\*] BC = 0x00000AAA << BIOS Control (b:d.f 00:31.5 + 0xDC)

[00] BIOSWE = 0 << BIOS Write Enable

[01] BLE = 1 << BIOS Lock Enable

[02] SRC = 2 << SPI Read Configuration

[04] TSS = 0 << Top Swap Status

[05] SMM\_BWP = 1 << SMM BIOS Write Protection

[06] BBS = 0 << Boot BIOS Strap

[07] BILD = 1 << BIOS Interface Lock Down

SPI Protected Ranges

------------------------------------------------------------

PRx (offset) | Value | Base | Limit | WP? | RP?

------------------------------------------------------------

PR0 (84) | 87FF06B0 | 006B0000 | 007FFFFF | 1 | 0

PR1 (88) | 85400252 | 00252000 | 00540FFF | 1 | 0

PR2 (8C) | 00000000 | 00000000 | 00000000 | 0 | 0

PR3 (90) | 00000000 | 00000000 | 00000000 | 0 | 0

PR4 (94) | 00000000 | 00000000 | 00000000 | 0 | 0

[CHIPSEC] Time elapsed 0.009

# Приложение 2

################################################################

## ##

## CHIPSEC: Platform Hardware Security Assessment Framework ##

## ##

################################################################

[CHIPSEC] Version : 1.12.3

[CHIPSEC] Arguments: -l tests.txt

[CHIPSEC] OS : Linux 5.18.0-kali5-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 5.18.5-1kali6 (2022-07-07) x86\_64

[CHIPSEC] Python : 3.11.5 (64-bit)

[CHIPSEC] Helper : LinuxHelper (/home/kali/Desktop/chipsec/chipsec/helper/linux/chipsec.ko)

[CHIPSEC] Platform: Mobile 6th Generation Core Processor (Skylake U)

[CHIPSEC] CPUID: 406E3

[CHIPSEC] VID: 8086

[CHIPSEC] DID: 1904

[CHIPSEC] RID: 08

[CHIPSEC] PCH : PCH-U Premium

[CHIPSEC] VID: 8086

[CHIPSEC] DID: 9D48

[CHIPSEC] RID: 21

[\*] loading common modules from "./chipsec/modules/common" ..

[\*] No platform specific modules to load

[\*] loading modules from "./chipsec/modules" ..

[+] loaded chipsec.modules.common.bios\_kbrd\_buffer

[+] loaded chipsec.modules.common.bios\_smi

[+] loaded chipsec.modules.common.bios\_ts

[+] loaded chipsec.modules.common.bios\_wp

[+] loaded chipsec.modules.common.cpu.cpu\_info

[+] loaded chipsec.modules.common.cpu.ia\_untrusted

[+] loaded chipsec.modules.common.cpu.spectre\_v2

[+] loaded chipsec.modules.common.debugenabled

[+] loaded chipsec.modules.common.ia32cfg

[+] loaded chipsec.modules.common.me\_mfg\_mode

[+] loaded chipsec.modules.common.memconfig

[+] loaded chipsec.modules.common.memlock

[+] loaded chipsec.modules.common.remap

[+] loaded chipsec.modules.common.rtclock

[+] loaded chipsec.modules.common.secureboot.variables

[+] loaded chipsec.modules.common.sgx\_check

[+] loaded chipsec.modules.common.smm

[+] loaded chipsec.modules.common.smm\_code\_chk

[+] loaded chipsec.modules.common.smm\_dma

[+] loaded chipsec.modules.common.smrr

[+] loaded chipsec.modules.common.spd\_wd

[+] loaded chipsec.modules.common.spi\_access

[+] loaded chipsec.modules.common.spi\_desc

[+] loaded chipsec.modules.common.spi\_fdopss

[+] loaded chipsec.modules.common.spi\_lock

[+] loaded chipsec.modules.common.uefi.access\_uefispec

[+] loaded chipsec.modules.common.uefi.s3bootscript

[\*] running loaded modules ..

[\*] Running module: chipsec.modules.common.bios\_kbrd\_buffer

[x][ =======================================================================

[x][ Module: Pre-boot Passwords in the BIOS Keyboard Buffer

[x][ =======================================================================

[\*] Keyboard buffer head pointer = 0x26 (at 0x41A), tail pointer = 0x26 (at 0x41C)

[\*] Keyboard buffer contents (at 0x41E):

E0 50 E0 50 E0 50 0D 1C 00 00 00 00 00 00 00 00 | P P P

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

[-] Keyboard buffer tail points inside the buffer (= 0x26)

It may potentially expose lengths of pre-boot passwords. Was your password 5 characters long?

[\*] Checking contents of the keyboard buffer..

WARNING: Keyboard buffer is not empty. The test cannot determine conclusively if it contains pre-boot passwords.

- The contents might have not been cleared by pre-boot firmware or overwritten with garbage.

- Visually inspect the contents of keyboard buffer for pre-boot passwords (BIOS, HDD, full-disk encryption).

[\*] Running module: chipsec.modules.common.bios\_smi

[x][ =======================================================================

[x][ Module: SMI Events Configuration

[x][ =======================================================================

[+] SMM BIOS region write protection is enabled (SMM\_BWP is used)

[\*] Checking SMI enables..

Global SMI enable: 1

TCO SMI enable : 1

[+] All required SMI events are enabled

[\*] Checking SMI configuration locks..

[+] TCO SMI configuration is locked (TCO SMI Lock)

[+] SMI events global configuration is locked (SMI Lock)

[+] PASSED: All required SMI sources seem to be enabled and locked

[\*] Running module: chipsec.modules.common.bios\_ts

[x][ =======================================================================

[x][ Module: BIOS Interface Lock (including Top Swap Mode)

[x][ =======================================================================

[\*] BiosInterfaceLockDown (BILD) control = 1

[\*] BIOS Top Swap mode is disabled (TSS = 0)

[\*] RTC TopSwap control (TS) = 0

[+] PASSED: BIOS Interface is locked (including Top Swap Mode)

[\*] Running module: chipsec.modules.common.bios\_wp

[x][ =======================================================================

[x][ Module: BIOS Region Write Protection

[x][ =======================================================================

[\*] BC = 0x00000AAA << BIOS Control (b:d.f 00:31.5 + 0xDC)

[00] BIOSWE = 0 << BIOS Write Enable

[01] BLE = 1 << BIOS Lock Enable

[02] SRC = 2 << SPI Read Configuration

[04] TSS = 0 << Top Swap Status

[05] SMM\_BWP = 1 << SMM BIOS Write Protection

[06] BBS = 0 << Boot BIOS Strap

[07] BILD = 1 << BIOS Interface Lock Down

[+] BIOS region write protection is enabled (writes restricted to SMM)

[\*] BIOS Region: Base = 0x00200000, Limit = 0x007FFFFF

SPI Protected Ranges

------------------------------------------------------------

PRx (offset) | Value | Base | Limit | WP? | RP?

------------------------------------------------------------

PR0 (84) | 87FF06B0 | 006B0000 | 007FFFFF | 1 | 0

PR1 (88) | 85400252 | 00252000 | 00540FFF | 1 | 0

PR2 (8C) | 00000000 | 00000000 | 00000000 | 0 | 0

PR3 (90) | 00000000 | 00000000 | 00000000 | 0 | 0

PR4 (94) | 00000000 | 00000000 | 00000000 | 0 | 0

[!] SPI protected ranges write-protect parts of BIOS region (other parts of BIOS can be modified)

[+] PASSED: BIOS is write protected

[\*] Running module: chipsec.modules.common.cpu.cpu\_info

[x][ =======================================================================

[x][ Module: Current Processor Information:

[x][ =======================================================================

[\*] Thread 0000

[\*] Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz

[\*] Family: 06 Model: 4E Stepping: 3

[\*] Microcode: 000000F0

[\*]

[\*] Thread 0001

[\*] Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz

[\*] Family: 06 Model: 4E Stepping: 3

[\*] Microcode: 000000F0

[\*]

[\*] Thread 0002

[\*] Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz

[\*] Family: 06 Model: 4E Stepping: 3

[\*] Microcode: 000000F0

[\*]

[\*] Thread 0003

[\*] Processor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz

[\*] Family: 06 Model: 4E Stepping: 3

[\*] Microcode: 000000F0

[\*]

[#] INFORMATION: Processor information displayed

[\*] Running module: chipsec.modules.common.cpu.ia\_untrusted

[!] MSR\_BIOS\_DONE.IA\_UNTRUSTED is not defined for platform. Skipping checks.

Skipping module chipsec.modules.common.cpu.ia\_untrusted since it is not applicable in this environment and/or platform

[\*] Running module: chipsec.modules.common.cpu.spectre\_v2

[x][ =======================================================================

[x][ Module: Checks for Branch Target Injection / Spectre v2 (CVE-2017-5715)

[x][ =======================================================================

[\*] CPUID.7H:EDX[26] = 1 Indirect Branch Restricted Speculation (IBRS) & Predictor Barrier (IBPB)

[\*] CPUID.7H:EDX[27] = 1 Single Thread Indirect Branch Predictors (STIBP)

[\*] CPUID.7H:EDX[29] = 1 IA32\_ARCH\_CAPABILITIES

[+] CPU supports IBRS and IBPB

[+] CPU supports STIBP

[\*] Checking enhanced IBRS support in IA32\_ARCH\_CAPABILITIES...

[\*] cpu0: IBRS\_ALL = 0

[-] CPU doesn't support enhanced IBRS

WARNING: CPU supports mitigation (IBRS) but doesn't support enhanced IBRS

[!] OS may be using software based mitigation (eg. retpoline)

WARNING: Retpoline check not implemented in current environment

[\*] Running module: chipsec.modules.common.debugenabled

[x][ =======================================================================

[x][ Module: Debug features test

[x][ =======================================================================

[\*] Checking IA32\_DEBUG\_INTERFACE MSR status

[!] Debug Occurred bit set in IA32\_DEBUG\_INTERFACE MSR

[!] Debug Occurred bit set in IA32\_DEBUG\_INTERFACE MSR

[!] Debug Occurred bit set in IA32\_DEBUG\_INTERFACE MSR

[!] Debug Occurred bit set in IA32\_DEBUG\_INTERFACE MSR

[\*] Checking DCI register status

[+] DCI Debug is disabled

[\*] Module Results:

[!] IA32\_DEBUG\_INTERFACE.DEBUG\_OCCURRED bit is set.

WARNING: An unexpected debug state was discovered on this platform

[\*] Running module: chipsec.modules.common.ia32cfg

[x][ =======================================================================

[x][ Module: IA32 Feature Control Lock

[x][ =======================================================================

[\*] Verifying IA32\_Feature\_Control MSR is locked on all logical CPUs..

[\*] cpu0: IA32\_FEATURE\_CONTROL Lock = 1

[\*] cpu1: IA32\_FEATURE\_CONTROL Lock = 1

[\*] cpu2: IA32\_FEATURE\_CONTROL Lock = 1

[\*] cpu3: IA32\_FEATURE\_CONTROL Lock = 1

[+] PASSED: IA32\_FEATURE\_CONTROL MSR is locked on all logical CPUs

[\*] Running module: chipsec.modules.common.me\_mfg\_mode

[x][ =======================================================================

[x][ Module: ME Manufacturing Mode

[x][ =======================================================================

[+] PASSED: ME is not in Manufacturing Mode

[\*] Running module: chipsec.modules.common.memconfig

[x][ =======================================================================

[x][ Module: Host Bridge Memory Map Locks

[x][ =======================================================================

[\*]

[\*] Checking register lock state:

[+] PCI0.0.0\_BDSM = 0x000000008C000001 - LOCKED - Base of Graphics Stolen Memory

[+] PCI0.0.0\_BGSM = 0x000000008B800001 - LOCKED - Base of GTT Stolen Memory

[+] PCI0.0.0\_DPR = 0x000000008B000001 - LOCKED - DMA Protected Range

[+] PCI0.0.0\_GGC = 0x00000000000002C1 - LOCKED - Graphics Control

[+] PCI0.0.0\_MESEG\_MASK = 0x0000007FFF000C00 - LOCKED - Manageability Engine Limit Address Register

[+] PCI0.0.0\_PAVPC = 0x000000008FF00047 - LOCKED - PAVP Configuration

[+] PCI0.0.0\_REMAPBASE = 0x00000001FF000001 - LOCKED - Memory Remap Base Address

[+] PCI0.0.0\_REMAPLIMIT = 0x000000026EF00001 - LOCKED - Memory Remap Limit Address

[+] PCI0.0.0\_TOLUD = 0x0000000090000001 - LOCKED - Top of Low Usable DRAM

[+] PCI0.0.0\_TOM = 0x0000000200000001 - LOCKED - Top of Memory

[+] PCI0.0.0\_TOUUD = 0x000000026F000001 - LOCKED - Top of Upper Usable DRAM

[+] PCI0.0.0\_TSEGMB = 0x000000008B000001 - LOCKED - TSEG Memory Base

[\*]

[+] PASSED: All memory map registers seem to be locked down

[\*] Running module: chipsec.modules.common.memlock

[x][ =======================================================================

[x][ Module: Check MSR\_LT\_LOCK\_MEMORY

[x][ =======================================================================

[\*] Checking MSR\_LT\_LOCK\_MEMORY status

[\*] cpu0: MSR\_LT\_LOCK\_MEMORY[LT\_LOCK] = 1

[\*] cpu1: MSR\_LT\_LOCK\_MEMORY[LT\_LOCK] = 1

[\*] cpu2: MSR\_LT\_LOCK\_MEMORY[LT\_LOCK] = 1

[\*] cpu3: MSR\_LT\_LOCK\_MEMORY[LT\_LOCK] = 1

[+] PASSED: MSR\_LT\_LOCK\_MEMORY.LT\_LOCK bit is set

[\*] Running module: chipsec.modules.common.remap

[x][ =======================================================================

[x][ Module: Memory Remapping Configuration

[x][ =======================================================================

[\*] Registers:

[\*] TOUUD : 0x000000026F000001

[\*] REMAPLIMIT: 0x000000026EF00001

[\*] REMAPBASE : 0x00000001FF000001

[\*] TOLUD : 0x90000001

[\*] TSEGMB : 0x8B000001

[\*] Memory Map:

[\*] Top Of Upper Memory: 0x000000026F000000

[\*] Remap Limit Address: 0x000000026EFFFFFF

[\*] Remap Base Address : 0x00000001FF000000

[\*] 4GB : 0x0000000100000000

[\*] Top Of Low Memory : 0x0000000090000000

[\*] TSEG (SMRAM) Base : 0x000000008B000000

[\*] Checking memory remap configuration..

[\*] Memory Remap is enabled

[+] Remap window configuration is correct: REMAPBASE <= REMAPLIMIT < TOUUD

[+] All addresses are 1MB aligned

[\*] Checking if memory remap configuration is locked..

[+] TOUUD is locked

[+] TOLUD is locked

[+] REMAPBASE and REMAPLIMIT are locked

[+] PASSED: Memory Remap is configured correctly and locked

[\*] Running module: chipsec.modules.common.rtclock

[x][ =======================================================================

[x][ Module: Protected RTC memory locations

[x][ =======================================================================

[\*] RC = 0x8000001C << RTC Configuration (msgbus port 0xC3, off 0x3400)

[02] UE = 1 << Upper 128 Byte Enable

[03] LL = 1 << Lower 128 Byte Lock

[04] UL = 1 << Upper 128 Byte Lock

[31] BILD = 1 << BIOS Interface Lock-Down

[+] Protected bytes (0x38-0x3F) in low 128-byte bank of RTC memory are locked

[+] Protected bytes (0x38-0x3F) in high 128-byte bank of RTC memory are locked

[+] PASSED: Protected locations in RTC memory are locked

[\*] Running module: chipsec.modules.common.secureboot.variables

[!] OS does not support UEFI Runtime API. Skipping module.

Skipping module chipsec.modules.common.secureboot.variables since it is not applicable in this environment and/or platform

[\*] Running module: chipsec.modules.common.sgx\_check

[x][ =======================================================================

[x][ Module: Check SGX feature support

[x][ =======================================================================

[\*] Test if CPU has support for SGX

[\*] SGX BIOS enablement check

[\*] Verifying IA32\_FEATURE\_CONTROL MSR is configured

[!] Intel SGX is not enabled in BIOS

[\*] Verifying IA32\_FEATURE\_CONTROL MSR is locked

[+] IA32\_Feature\_Control locked

[\*] Verifying if Protected Memory Range (PRMRR) is configured

[+] Protected Memory Range configuration is supported

[\*] Verifying PRMRR Configuration on each core.

[+] PRMRR config is uniform across all CPUs

[\*] PRMRR config supports: 1M, 32M, 64M, 128M

[\*] PRMRR Base and Mask are set to zero. PRMRR appears to be disabled.

[\*] Skipping Base/Mask settings.

[\*] Verifying if SGX is available to use

[!] Intel SGX instructions disabled by firmware

[\*] BIOS\_SE\_SVN = 0xFFFFFFFF0000 << BIOS SGX SVN values (MSR 0x302 Thread 0x0)

[16] SINIT\_SE\_SVN = FF << SINIT\_SE\_SVN

[24] SCLEAN\_SE\_SVN = FF << SCLEAN\_SE\_SVN

[32] ANC\_SE\_SVN = FF << ANC\_SE\_SVN

[40] PFAT\_SE\_SVN = FF << PFAT\_SE\_SVN

[\*] BIOS\_SE\_SVN\_STATUS = 0x600000000 << BIOS SGX SVN status register (MSR 0x500 Thread 0x0)

[00] LOCK = 0 << ACM threshold locked

[16] SINIT\_SE\_SVN = 0 << SINIT\_SE\_SVN

[\*] Check SGX debug feature settings

[\*] SGX Debug Enable : 0

[\*] Check Silicon debug feature settings

[\*] IA32\_DEBUG\_INTERFACE : 0xC0000000

[\*] Debug enabled : 0

[\*] Lock : 1

[+] SGX debug mode is disabled

[+] Silicon debug features are disabled

[+] Silicon debug Feature Control register is locked

WARNING: One or more SGX checks detected a warning

[\*] Running module: chipsec.modules.common.smm

[x][ =======================================================================

[x][ Module: Compatible SMM memory (SMRAM) Protection

[x][ =======================================================================

[\*] PCI0.0.0\_SMRAMC = 0x1A << System Management RAM Control (b:d.f 00:00.0 + 0x88)

[00] C\_BASE\_SEG = 2 << SMRAM Base Segment = 010b

[03] G\_SMRAME = 1 << SMRAM Enabled

[04] D\_LCK = 1 << SMRAM Locked

[05] D\_CLS = 0 << SMRAM Closed

[06] D\_OPEN = 0 << SMRAM Open

[\*] Compatible SMRAM is enabled

[+] PASSED: Compatible SMRAM is locked down

[\*] Running module: chipsec.modules.common.smm\_code\_chk

[x][ =======================================================================

[x][ Module: SMM\_Code\_Chk\_En (SMM Call-Out) Protection

[x][ =======================================================================

[\*] MSR\_SMM\_FEATURE\_CONTROL = 0x00000005 << Enhanced SMM Feature Control (MSR 0x4E0 Thread 0x0)

[00] LOCK = 1 << Lock bit

[02] SMM\_Code\_Chk\_En = 1 << Prevents SMM from executing code outside the ranges defined by the SMRR

[\*] MSR\_SMM\_FEATURE\_CONTROL = 0x00000005 << Enhanced SMM Feature Control (MSR 0x4E0 Thread 0x1)

[00] LOCK = 1 << Lock bit

[02] SMM\_Code\_Chk\_En = 1 << Prevents SMM from executing code outside the ranges defined by the SMRR

[\*] MSR\_SMM\_FEATURE\_CONTROL = 0x00000005 << Enhanced SMM Feature Control (MSR 0x4E0 Thread 0x2)

[00] LOCK = 1 << Lock bit

[02] SMM\_Code\_Chk\_En = 1 << Prevents SMM from executing code outside the ranges defined by the SMRR

[\*] MSR\_SMM\_FEATURE\_CONTROL = 0x00000005 << Enhanced SMM Feature Control (MSR 0x4E0 Thread 0x3)

[00] LOCK = 1 << Lock bit

[02] SMM\_Code\_Chk\_En = 1 << Prevents SMM from executing code outside the ranges defined by the SMRR

[+] PASSED: SMM\_Code\_Chk\_En is enabled and locked down

[\*] Running module: chipsec.modules.common.smm\_dma

[x][ =======================================================================

[x][ Module: SMM TSEG Range Configuration Check

[x][ =======================================================================

[\*] TSEG : 0x000000008B000000 - 0x000000008B7FFFFF (size = 0x00800000)

[\*] SMRR range: 0x000000008B000000 - 0x000000008B7FFFFF (size = 0x00800000)

[\*] Checking TSEG range configuration..

[+] TSEG range covers entire SMRAM

[+] TSEG range is locked

[+] PASSED: TSEG is properly configured. SMRAM is protected from DMA attacks

[\*] Running module: chipsec.modules.common.smrr

[x][ =======================================================================

[x][ Module: CPU SMM Cache Poisoning / System Management Range Registers

[x][ =======================================================================

[+] OK. SMRR range protection is supported

[\*] Checking SMRR range base programming..

[\*] IA32\_SMRR\_PHYSBASE = 0x8B000006 << SMRR Base Address MSR (MSR 0x1F2 Thread 0x0)

[00] Type = 6 << SMRR memory type

[12] PhysBase = 8B000 << SMRR physical base address

[\*] SMRR range base: 0x000000008B000000

[\*] SMRR range memory type is Writeback (WB)

[+] OK so far. SMRR range base is programmed

[\*] Checking SMRR range mask programming..

[\*] IA32\_SMRR\_PHYSMASK = 0xFF800800 << SMRR Range Mask MSR (MSR 0x1F3 Thread 0x0)

[11] Valid = 1 << SMRR valid

[12] PhysMask = FF800 << SMRR address range mask

[\*] SMRR range mask: 0x00000000FF800000

[+] OK so far. SMRR range is enabled

[\*] Verifying that SMRR range base & mask are the same on all logical CPUs..

[CPU0] SMRR\_PHYSBASE = 000000008B000006, SMRR\_PHYSMASK = 00000000FF800800

[CPU1] SMRR\_PHYSBASE = 000000008B000006, SMRR\_PHYSMASK = 00000000FF800800

[CPU2] SMRR\_PHYSBASE = 000000008B000006, SMRR\_PHYSMASK = 00000000FF800800

[CPU3] SMRR\_PHYSBASE = 000000008B000006, SMRR\_PHYSMASK = 00000000FF800800

[+] OK so far. SMRR range base/mask match on all logical CPUs

[\*] Trying to read memory at SMRR base 0x8B000000..

[+] PASSED: SMRR reads are blocked in non-SMM mode

[+] PASSED: SMRR protection against cache attack is properly configured

[\*] Running module: chipsec.modules.common.spd\_wd

[x][ =======================================================================

[x][ Module: SPD Write Disable

[x][ =======================================================================

[\*] SMBUS\_HCFG = 0x11 << Host Configuration (b:d.f 00:31.4 + 0x40)

[00] HST\_EN = 1 << HST\_EN

[01] SMB\_SMI\_EN = 0 << SMB\_SMI\_EN

[02] I2C\_EN = 0 << I2C\_EN

[03] SSRESET = 0 << SSRESET

[04] SPD\_WD = 1 << SPD\_WD

[+] PASSED: SPD Write Disable is set

[\*] Running module: chipsec.modules.common.spi\_access

[x][ =======================================================================

[x][ Module: SPI Flash Region Access Control

[x][ =======================================================================

SPI Flash Region Access Permissions

------------------------------------------------------------

BIOS Region Write Access Grant (00):

FREG0\_FLASHD: 0

FREG1\_BIOS : 0

FREG2\_ME : 0

FREG3\_GBE : 0

FREG4\_PD : 0

FREG5 : 0

FREG6 : 0

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Read Access Grant (00):

FREG0\_FLASHD: 0

FREG1\_BIOS : 0

FREG2\_ME : 0

FREG3\_GBE : 0

FREG4\_PD : 0

FREG5 : 0

FREG6 : 0

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Write Access (4A):

FREG0\_FLASHD: 0

FREG1\_BIOS : 1

FREG2\_ME : 0

FREG3\_GBE : 1

FREG4\_PD : 0

FREG5 : 0

FREG6 : 1

FREG7 : 0

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

BIOS Region Read Access (CB):

FREG0\_FLASHD: 1

FREG1\_BIOS : 1

FREG2\_ME : 0

FREG3\_GBE : 1

FREG4\_PD : 0

FREG5 : 0

FREG6 : 1

FREG7 : 1

FREG8\_EC : 0

FREG9 : 0

FREG10 : 0

FREG11 : 0

WARNING: Software has write access to GBe region in SPI flash

WARNING: Certain SPI flash regions are writeable by software

[\*] Running module: chipsec.modules.common.spi\_desc

[x][ =======================================================================

[x][ Module: SPI Flash Region Access Control

[x][ =======================================================================

[\*] FRAP = 0x00004ACB << SPI Flash Regions Access Permissions Register (SPIBAR + 0x50)

[00] BRRA = CB << BIOS Region Read Access

[08] BRWA = 4A << BIOS Region Write Access

[16] BMRAG = 0 << BIOS Master Read Access Grant

[24] BMWAG = 0 << BIOS Master Write Access Grant

[\*] Software access to SPI flash regions: read = 0xCB, write = 0x4A

[+] PASSED: SPI flash permissions prevent SW from writing to flash descriptor

[\*] Running module: chipsec.modules.common.spi\_fdopss

[x][ =======================================================================

[x][ Module: SPI Flash Descriptor Security Override Pin-Strap

[x][ =======================================================================

[\*] HSFS = 0x3F00E800 << Hardware Sequencing Flash Status Register (SPIBAR + 0x4)

[00] FDONE = 0 << Flash Cycle Done

[01] FCERR = 0 << Flash Cycle Error

[02] AEL = 0 << Access Error Log

[03] BERASE = 0 << Block/Sector Erase Size

[05] SCIP = 0 << SPI cycle in progress

[11] WRSDIS = 1 << Write status disable

[12] PR34LKD = 0 << PRR3 PRR4 Lock-Down

[13] FDOPSS = 1 << Flash Descriptor Override Pin-Strap Status

[14] FDV = 1 << Flash Descriptor Valid

[15] FLOCKDN = 1 << Flash Configuration Lock-Down

[16] FGO = 0 << Flash cycle go

[17] FCYCLE = 0 << Flash Cycle Type

[21] WET = 0 << Write Enable Type

[24] FDBC = 3F << Flash Data Byte Count

[31] FSMIE = 0 << Flash SPI SMI# Enable

[+] PASSED: SPI Flash Descriptor Security Override is disabled

[\*] Running module: chipsec.modules.common.spi\_lock

[x][ =======================================================================

[x][ Module: SPI Flash Controller Configuration Locks

[x][ =======================================================================

[\*] HSFS = 0x3F00E800 << Hardware Sequencing Flash Status Register (SPIBAR + 0x4)

[00] FDONE = 0 << Flash Cycle Done

[01] FCERR = 0 << Flash Cycle Error

[02] AEL = 0 << Access Error Log

[03] BERASE = 0 << Block/Sector Erase Size

[05] SCIP = 0 << SPI cycle in progress

[11] WRSDIS = 1 << Write status disable

[12] PR34LKD = 0 << PRR3 PRR4 Lock-Down

[13] FDOPSS = 1 << Flash Descriptor Override Pin-Strap Status

[14] FDV = 1 << Flash Descriptor Valid

[15] FLOCKDN = 1 << Flash Configuration Lock-Down

[16] FGO = 0 << Flash cycle go

[17] FCYCLE = 0 << Flash Cycle Type

[21] WET = 0 << Write Enable Type

[24] FDBC = 3F << Flash Data Byte Count

[31] FSMIE = 0 << Flash SPI SMI# Enable

[+] SPI write status disable set.

[+] SPI Flash Controller configuration is locked

[+] PASSED: SPI Flash Controller locked correctly.

[\*] Running module: chipsec.modules.common.uefi.access\_uefispec

OS does not support UEFI Runtime API

Skipping module chipsec.modules.common.uefi.access\_uefispec since it is not applicable in this environment and/or platform

[\*] Running module: chipsec.modules.common.uefi.s3bootscript

OS does not support UEFI Runtime API

Skipping module chipsec.modules.common.uefi.s3bootscript since it is not applicable in this environment and/or platform

[CHIPSEC] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SUMMARY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[CHIPSEC] Time elapsed 9.825

[CHIPSEC] Modules total 27

[CHIPSEC] Modules failed to run 0:

[CHIPSEC] Modules passed 17:

[+] PASSED: chipsec.modules.common.bios\_smi

[+] PASSED: chipsec.modules.common.bios\_ts

[+] PASSED: chipsec.modules.common.bios\_wp

[+] PASSED: chipsec.modules.common.ia32cfg

[+] PASSED: chipsec.modules.common.me\_mfg\_mode

[+] PASSED: chipsec.modules.common.memconfig

[+] PASSED: chipsec.modules.common.memlock

[+] PASSED: chipsec.modules.common.remap

[+] PASSED: chipsec.modules.common.rtclock

[+] PASSED: chipsec.modules.common.smm

[+] PASSED: chipsec.modules.common.smm\_code\_chk

[+] PASSED: chipsec.modules.common.smm\_dma

[+] PASSED: chipsec.modules.common.smrr

[+] PASSED: chipsec.modules.common.spd\_wd

[+] PASSED: chipsec.modules.common.spi\_desc

[+] PASSED: chipsec.modules.common.spi\_fdopss

[+] PASSED: chipsec.modules.common.spi\_lock

[CHIPSEC] Modules information 1:

[#] INFORMATION: chipsec.modules.common.cpu.cpu\_info

[CHIPSEC] Modules failed 0:

[CHIPSEC] Modules with warnings 5:

WARNING: chipsec.modules.common.bios\_kbrd\_buffer

WARNING: chipsec.modules.common.cpu.spectre\_v2

WARNING: chipsec.modules.common.debugenabled

WARNING: chipsec.modules.common.sgx\_check

WARNING: chipsec.modules.common.spi\_access

[CHIPSEC] Modules skipped 0:

[CHIPSEC] Modules not applicable 4:

NOT APPLICABLE: chipsec.modules.common.cpu.ia\_untrusted

NOT APPLICABLE: chipsec.modules.common.secureboot.variables

NOT APPLICABLE: chipsec.modules.common.uefi.access\_uefispec

NOT APPLICABLE: chipsec.modules.common.uefi.s3bootscript

[CHIPSEC] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*