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

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Реверс-инжиниринг SMM модулей UEFI BIOS**

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**

Получить базовые навыки реверс-инжиниринга встроенного ПО UEFI BIOS на примере SMM-модулей.

# задачи

1. Изучить процесс инициализации SMM-режима в рамках функционирования фаз загрузки ЭВМ, построенной на базе UEFI BIOS (глава в отчет)
2. Выявить программные компоненты прошивки UEFI BIOS, участвующие в ходе инициализации SMM-режима. (глава в отчет)
3. Описать возможные способы перехода CPU в SMM-режим, соответствующие точки входа и обработчики (глава в отчет)
4. Произвести реверс-инжиниринг 3-х произвольных обработчиков #SWSMI в соответствии с примером (по 3 на каждого студента, если работа ведется в команде их 2-х человек). По каждому модулю нужно приложить IDB файл IDAPro + описать вектора воздействия на обработчик прерываний (см. пример)

# Ход работы

## Процесс инициализации SMM-режима

Режим SMM запускается во время инициализации платформы. Диспетчер DXE вызывает загрузчик начальной программы SMM. IPL загрузит драйверы SMM в SMRAM и будет поддерживать их работу до завершения работы платформы. Загрузка SMM драйверов, как показано на Рисунке 1, выполняется в DXE режиме при загрузке обычных UEFI DXE драйверов.

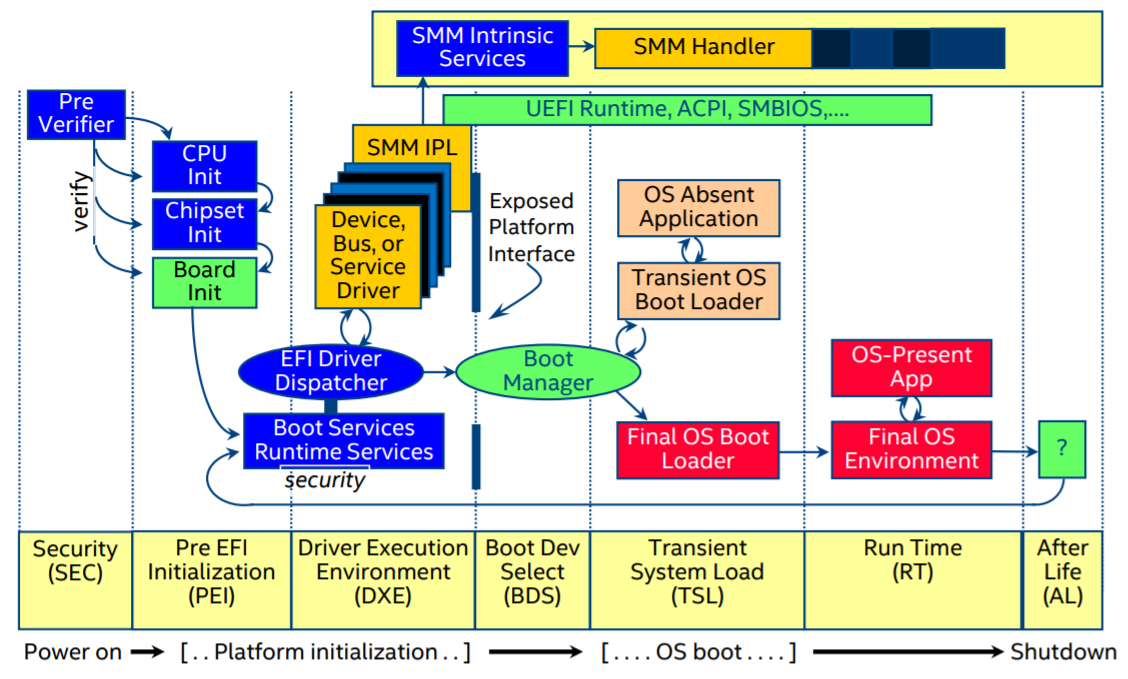


Рисунок — Общий процесс загрузки системы

## Описание работы SMM драйверов

Драйвер SMM — это модуль UEFI BIOS режима SMM, который загружается в SMRAM. Жизненный цикл драйверов SMM можно разделить на два этапа:

1. *Инициализация драйвера SMM.*

Фаза инициализации драйвера SMM начинается, когда драйвер загружается в SMRAM и вызывается его точка входа. Фаза инициализации драйвера SMM заканчивается, когда происходит выход из точки входа драйвера. Во время инициализации драйвера SMM драйверы SMM имеют доступ к двум наборам протоколов: протоколам UEFI и протоколам SMM.

Протоколы UEFI — это протоколы, которые обнаруживаются и устанавливаются с помощью служб загрузки UEFI. Протоколы UEFI могут быть обнаружены и использованы драйверами SMM только во время инициализации SMM.

Протоколы SMM — это протоколы, которые обнаруживаются и устанавливаются с помощью таблицы служб управления системой (System Management Services Table или SMST). Протоколы SMM могут быть обнаружены драйверами SMM на этапе инициализации и на этапе выполнения SMM.

1. *Время выполнения SMM.*

Во время выполнения драйверы SMM имеют доступ только к SMST службам. Кроме того, в зависимости от архитектуры платформы области памяти за пределами SMRAM могут быть недоступны для драйверов SMM. Аналогично, области памяти внутри SMRAM могут быть недоступны для драйверов UEFI. Выполнение драйвера SMM может происходит в результате SMI-прерывания: в случае вызова SMI-прерывания начинает свое исполнение обработчик прерывания драйвера, который зарегистрировал обработчик полученного SMI-прерывания на этапе инициализации.

## Способы перехода CPU в SMM-режим

SMM активируется при помощи прерываний SMI (system management interrupt — прерывание системного управления), которое возникает в одном из следующих трех случаев:

1. По сигналу от чипсета или периферии на материнской плате. Используется выделенный контакт процессора SMI#.
2. Программный SMI, посланный системным ПО через io-порт.
3. Запись по адресу ввода-вывода, для которого микро-программно установлена необходимость активации SMM.

На ближайшей границе инструкций после получения сигнала SMI#, процессор сохраняет своё состояние в памяти и переходит в SMM. В режиме SMM выполняются обработчики пришедшего SMI, которые устанавливаются драйверами SMM на этапе их инициализации. Для выхода из SMM и восстановления состояния процессора используется инструкция RSM.

## PiSmmCommunicationSmm

На рисунке 2 представлена точка входа в модуль.

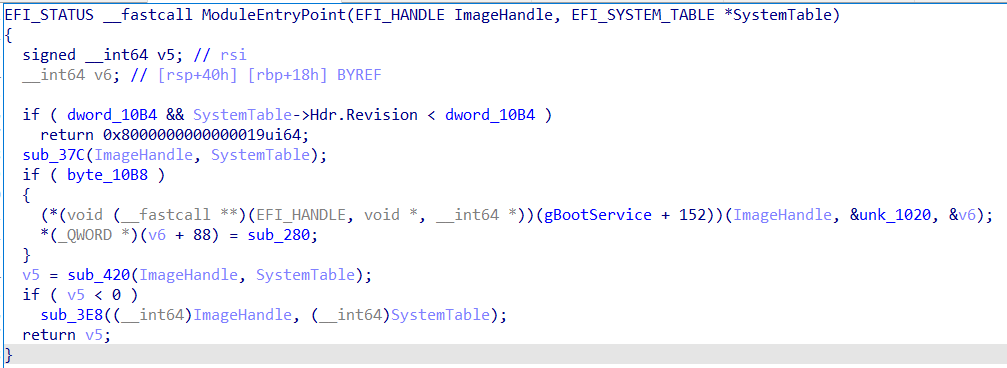


Рисунок — Точка входа

В функции sub\_37C происходит инициализация всех необходимых компонентов. Заменим имена переменных, настроим правильные типы, и получим следующие результаты (рисунки 3-10).

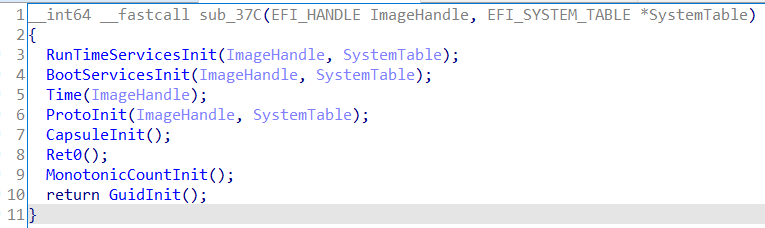


Рисунок — Содержимое функции инициализации

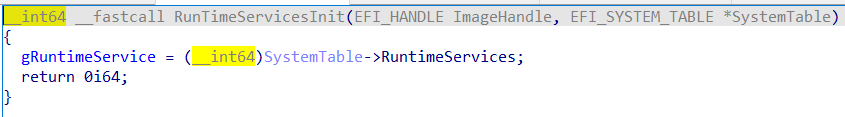


Рисунок — Настройка RuntimeService

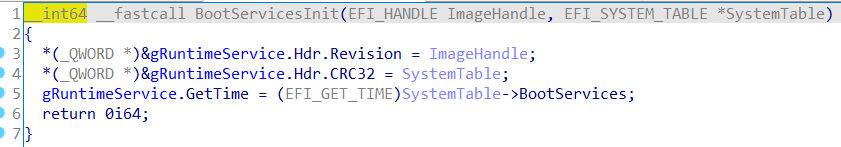


Рисунок — Настройка глобальных переменных, в т.ч. BootServices

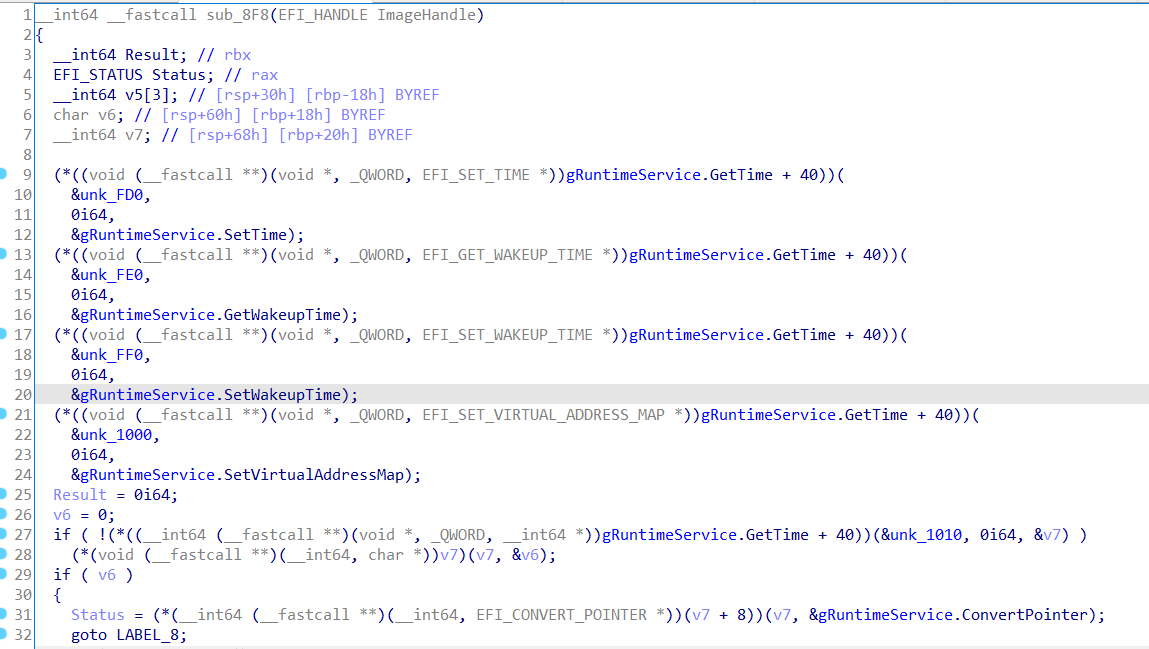


Рисунок — Настройка времени

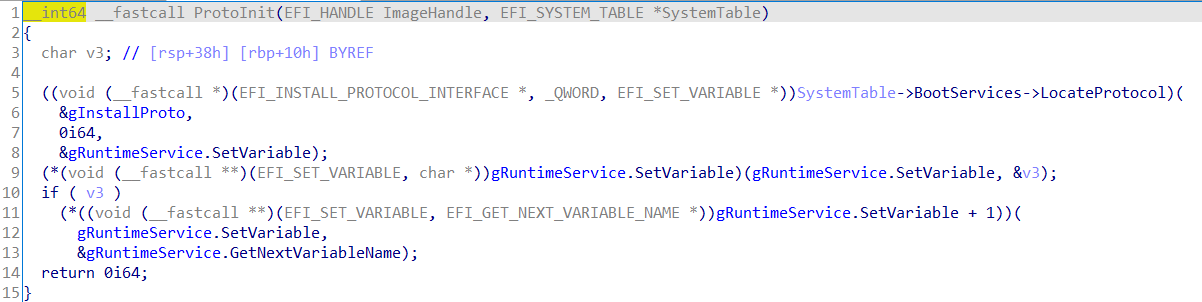


Рисунок — Настройка протокола

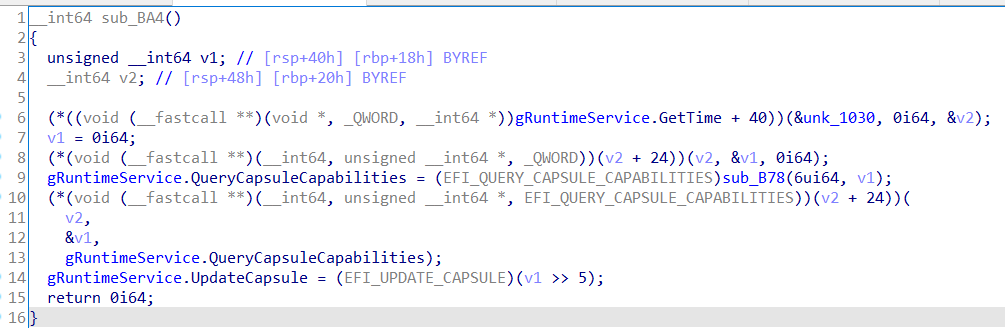


Рисунок — Настройка Capsule

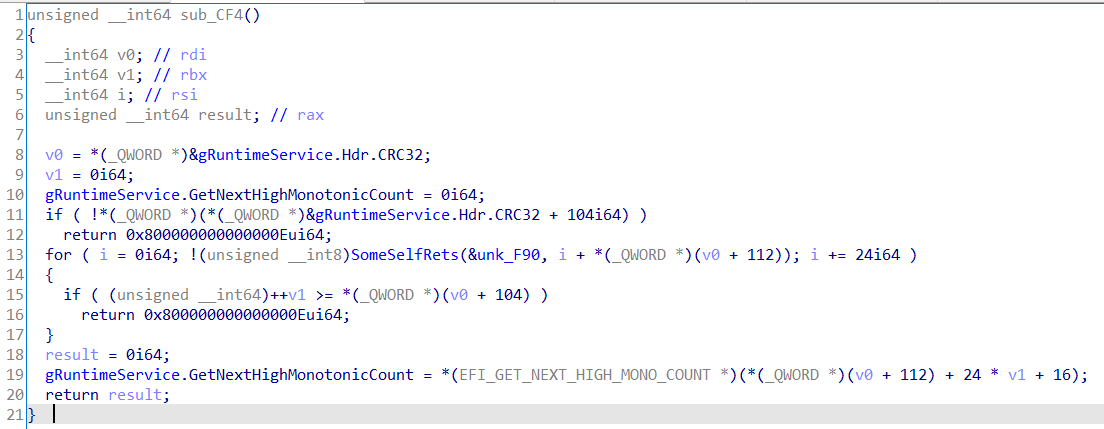


Рисунок — Настройка MonotonicCount

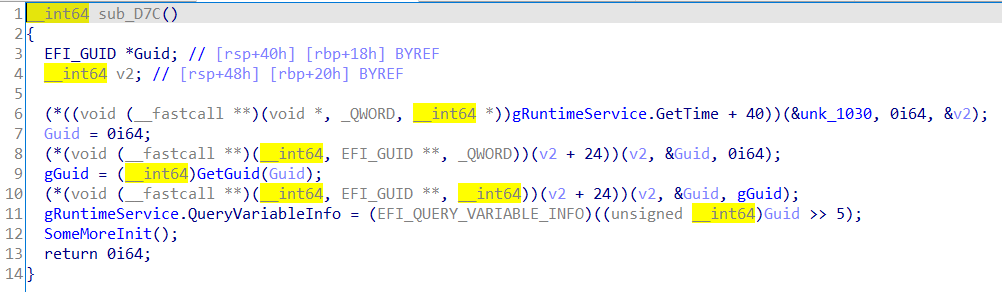


Рисунок — Настройка Guid и некоторые другие настройки

Теперь точка входа в модуль выглядит следующим образом (Рисунок 11).

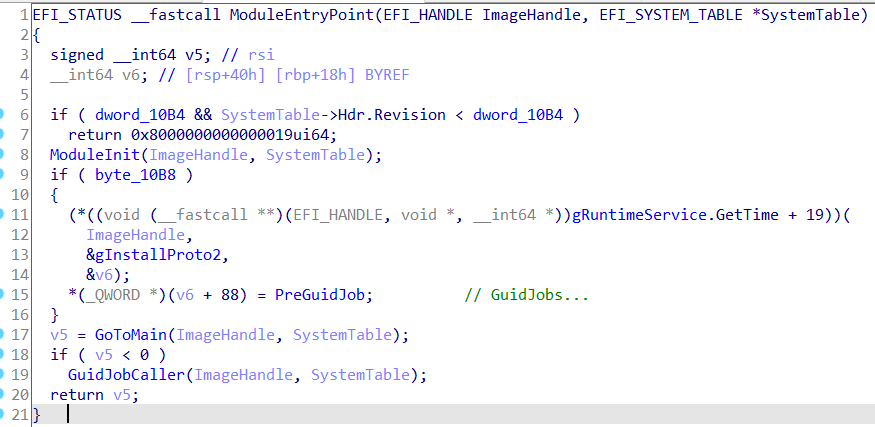


Рисунок — Обновленная точка входа

Рассмотрим, что находится в функции GoToMain (Рисунок 12): там происходит какая-то обработка регистров и затем переход в main (Рисунок 13).

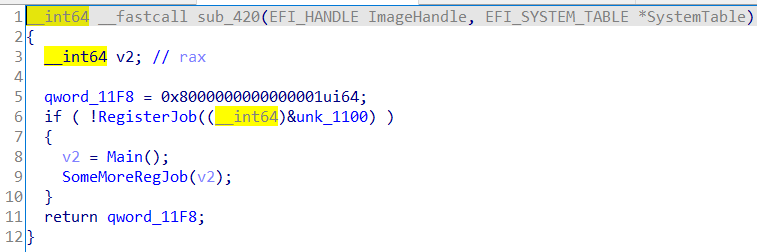


Рисунок — Содержимое функции GoToMain

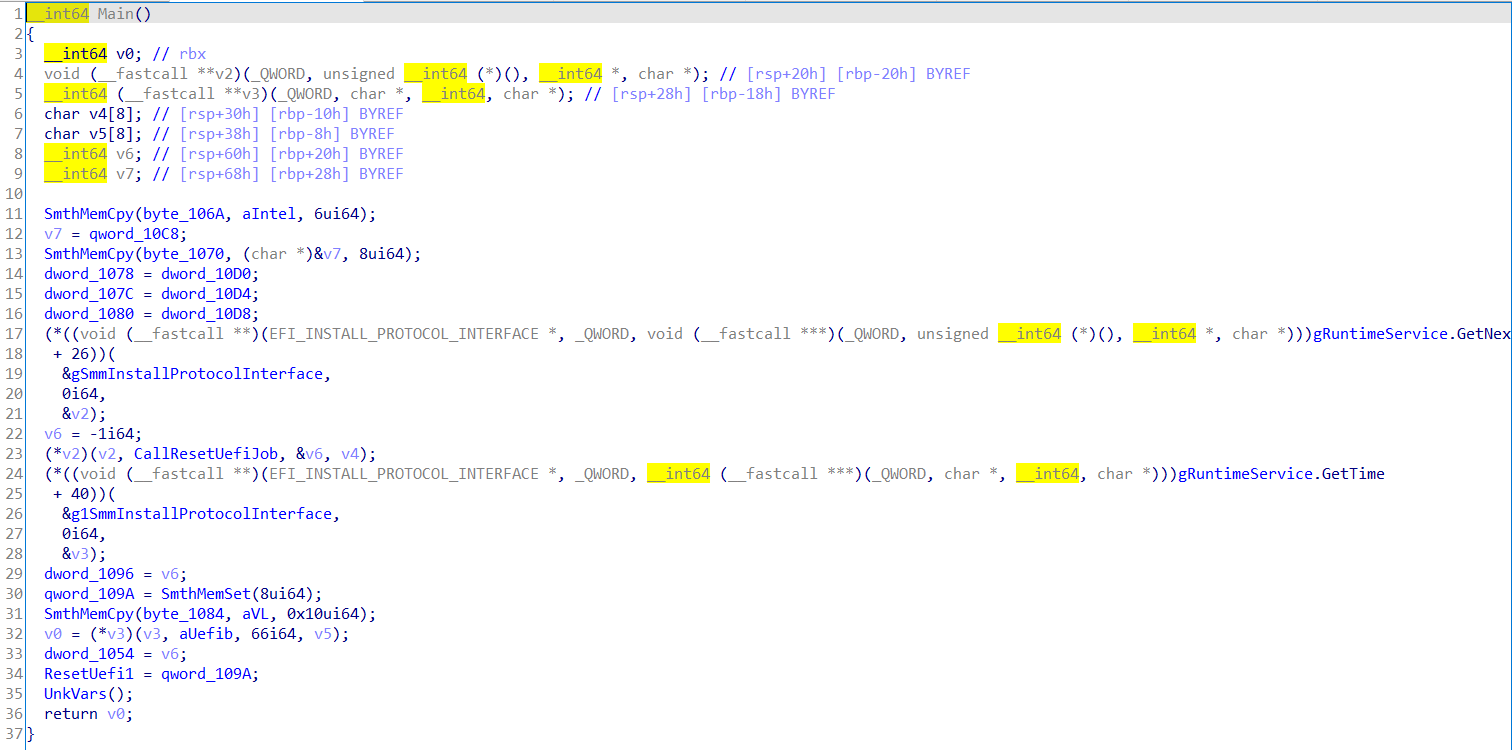


Рисунок — Содержимое функции Main

Как известно, в режим SMM процессор переходит по событию #SMI — специальное прерывание. Необходимо найти так называемые программные обработчики #SMI. Данные обработчики всегда устанавливают свои точки входа через протокол gEfiSmmSwDispatch2ProtocolGuid = { 0x18a3c6dc…}. Найдем данный обработчик с помощью обычного BinarySearch в IDAPro (Рисунок 14).

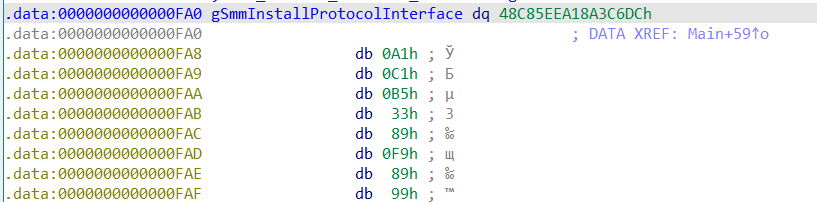


Рисунок — Результаты поиска gEfiSmmSwDispatch2ProtocolGuid

Смотрим на данный код и устанавливаем тип для переменной v2 как EFI\_SMM\_SW\_DISPATCH2\_PROTOCOL (Рисунок 15). Теперь видно, что используется функция Register, где вторым аргументом передаётся функция обработки события #SWSMI (выделена серым). Третьим аргументом передаётся уникальный номер (-1) данного обработчика, который будет использоваться для записи через порты ввода-вывода. Таким образом можно произвольно триггерить тот или иной обработчик, просто записав его номер в порты ввода-вывода.

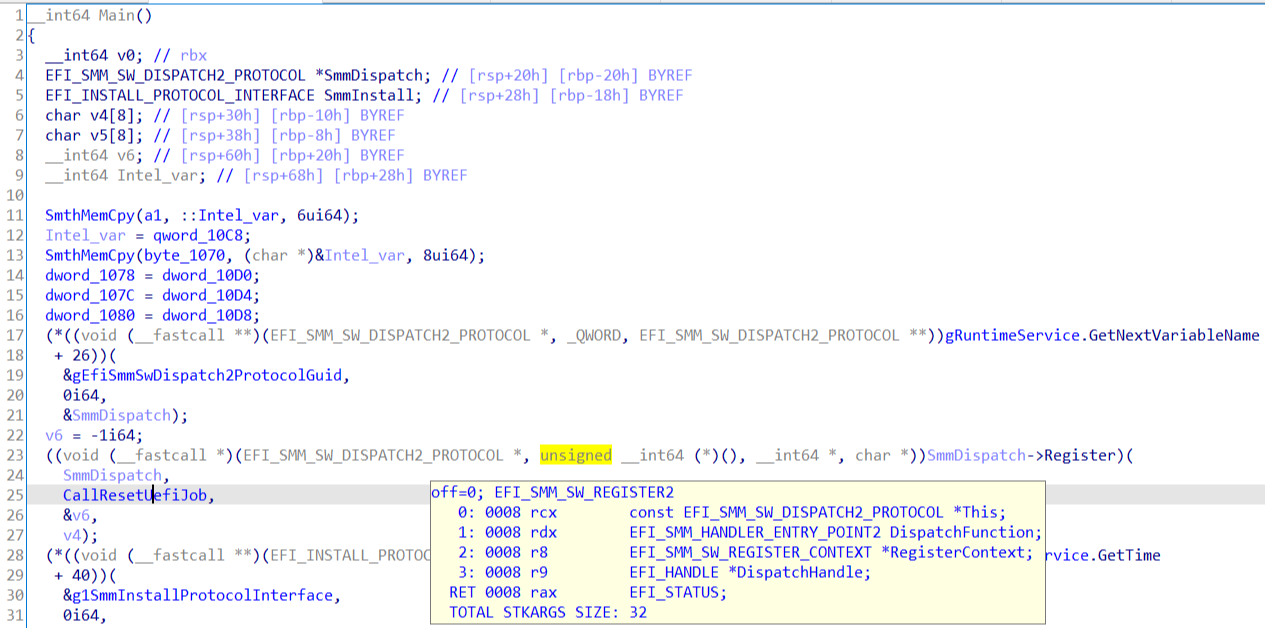


Рисунок — Измененная функция main

Благодаря открытому EDK2 возможно посмотреть аргументы обработчика прерываний (Рисунок 16). Вручную зададим аргументы функции (Рисунки 17 и 18).

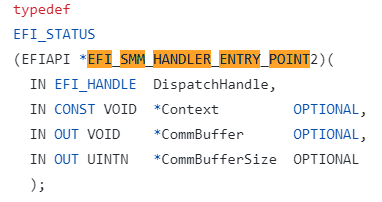


Рисунок — Структура данных

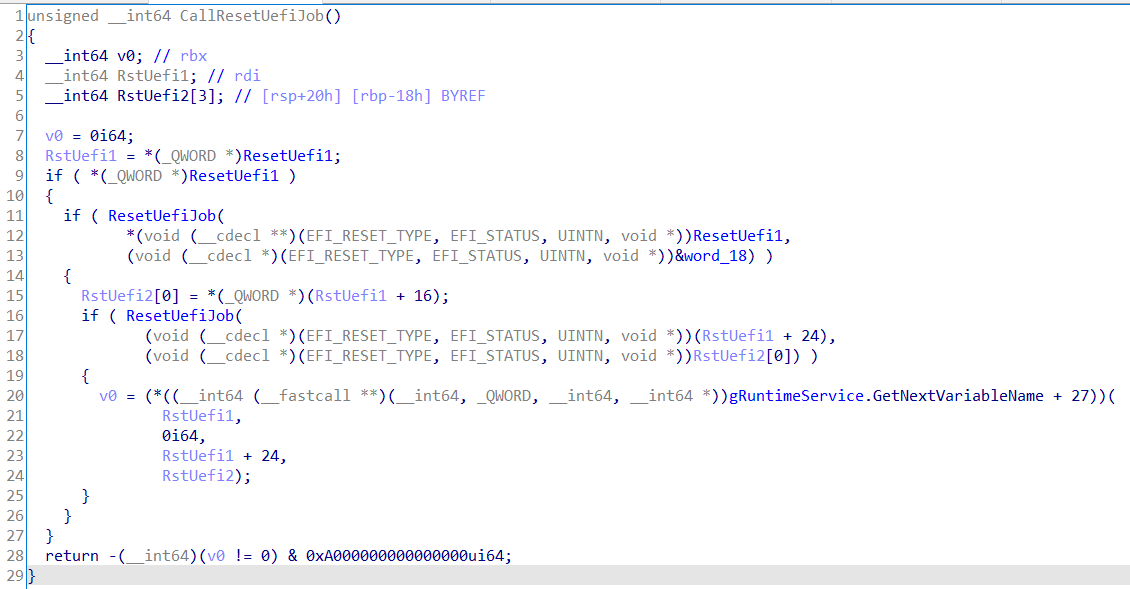


Рисунок — Функция обработчик прерываний (до изменений)

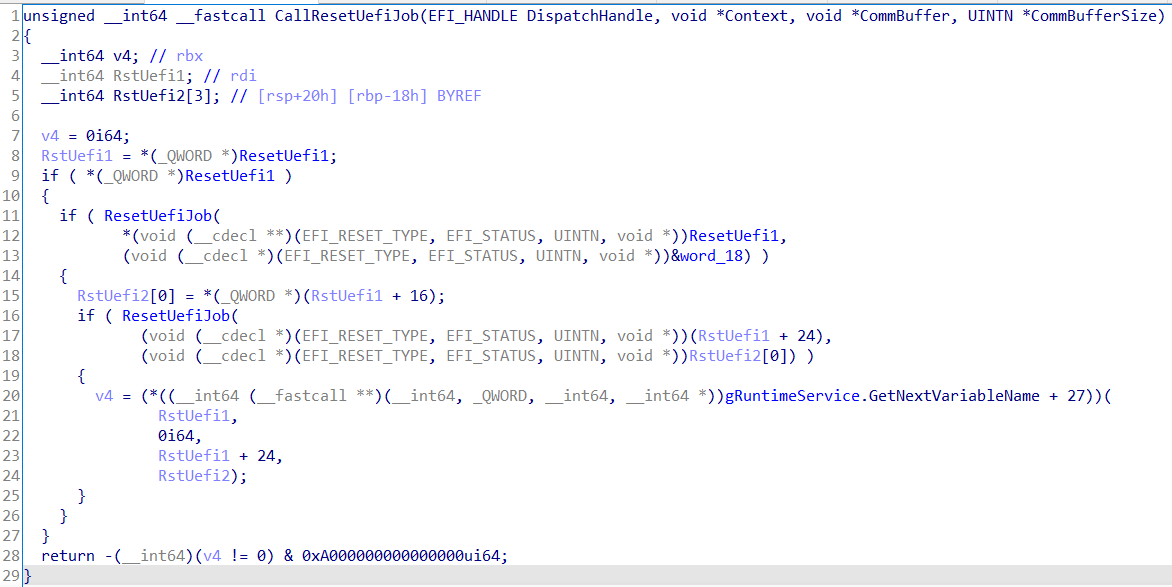


Рисунок — Функция обработчик прерываний (после изменений)

## AsfSecureBootSmm

На рисунке 19 представлена точка входа в модуль. Любой модуль принимает на вход два параметра: ImageHandle (практически бесполезен для реверса) и SystemTable (очень полезен). SystemTable — глобальная таблица сервисов UEFI в виде набора протоколов, которые устанавливают другие модули. Проще говоря это указатель на все доступные функции UEFI, часть из которых реализована в других модулях. А значит становится известным типы входных аргументов — EFI\_HANDLE и EFI\_SYSTEM\_TABLE\*. Укажем это в прототипе функции.

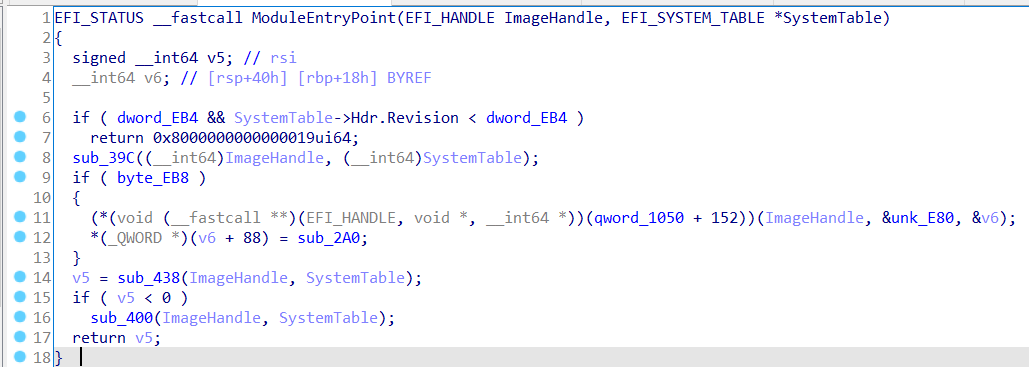


Рисунок — Точка входа в модуль

Данный модуль оказалось сложнее разреверсить, поскольку не все GUID’ы получилось найти. Также в данном модуле множество функций и переменных, которые обрабатывают протоколы. В итоге после некоторых модификаций точка входа приобрела следующий вид (Рисунок 20).

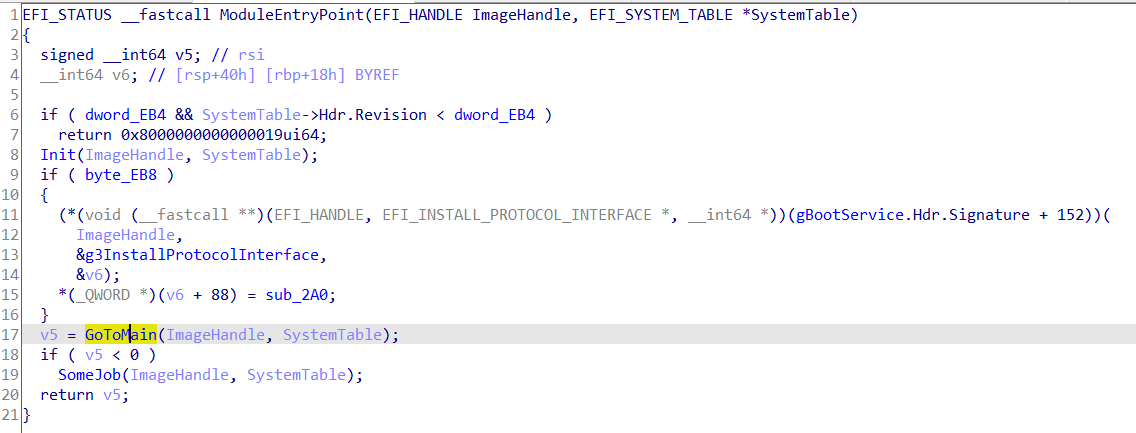


Рисунок — Обновленная точка входа

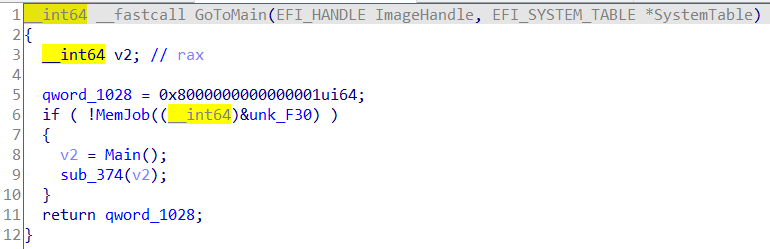


Рисунок — Содержимое функции GoToMain

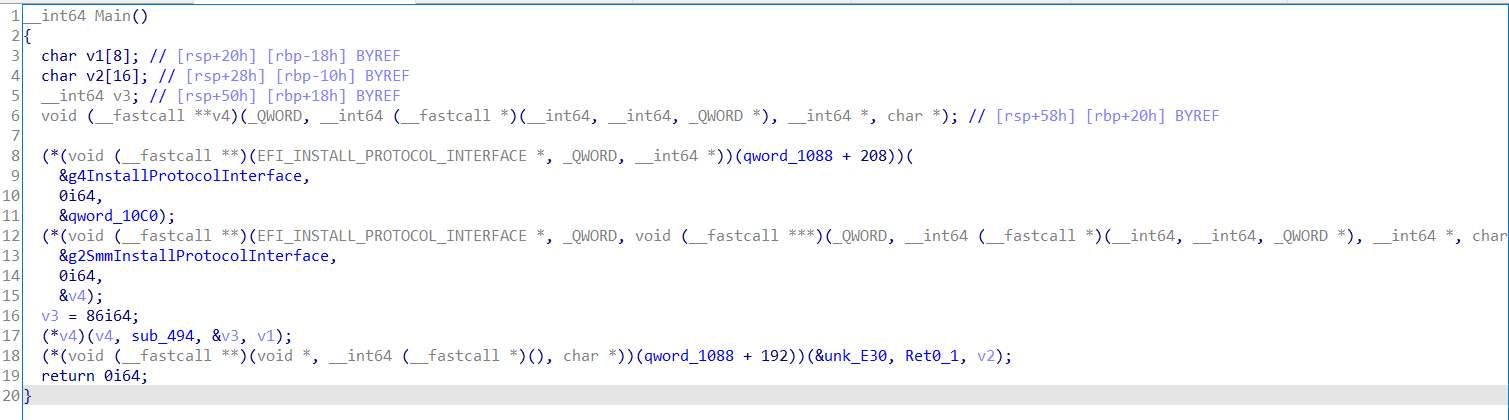


Рисунок — Содержимое функции Main

Найдем протокол gEfiSmmSwDispatch2ProtocolGuid по значению 0x18a3c6dc (находится в мейне). Установим все необходимые типы данных и переименуем переменные. Получим функцию обработки прерывания с номером 86 (Рисунок 23).

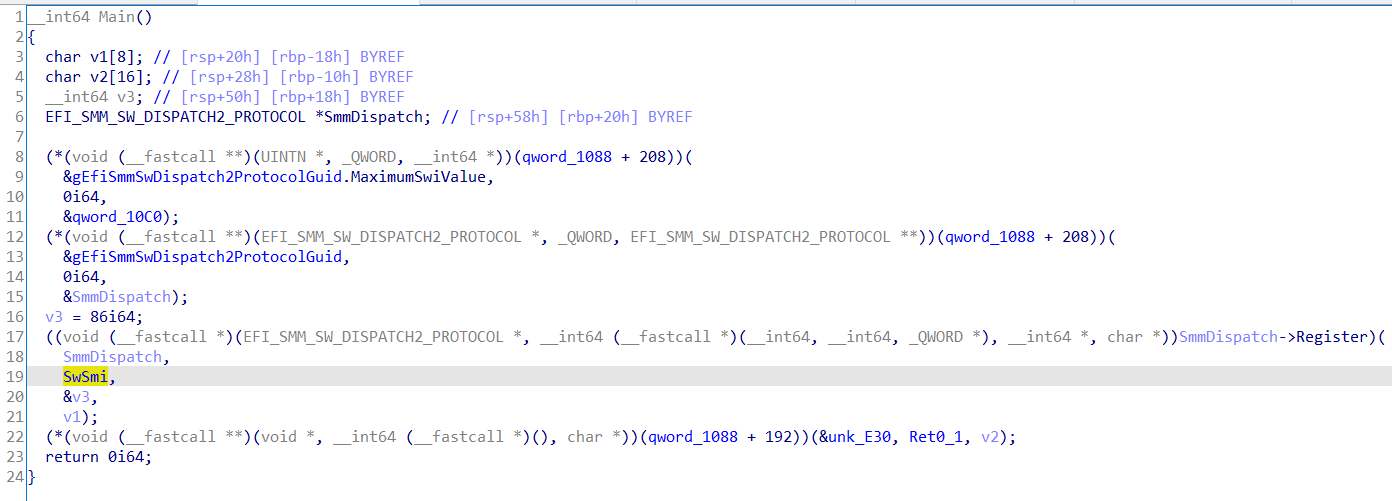
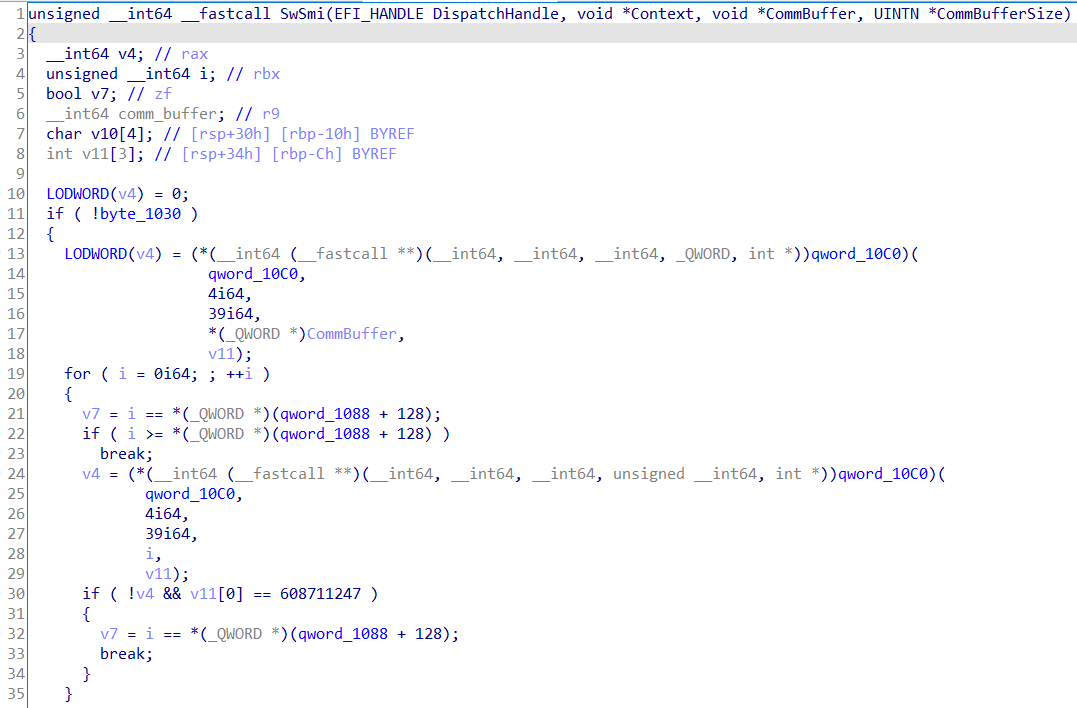


Рисунок — Содержимое функции Main (обновленное)



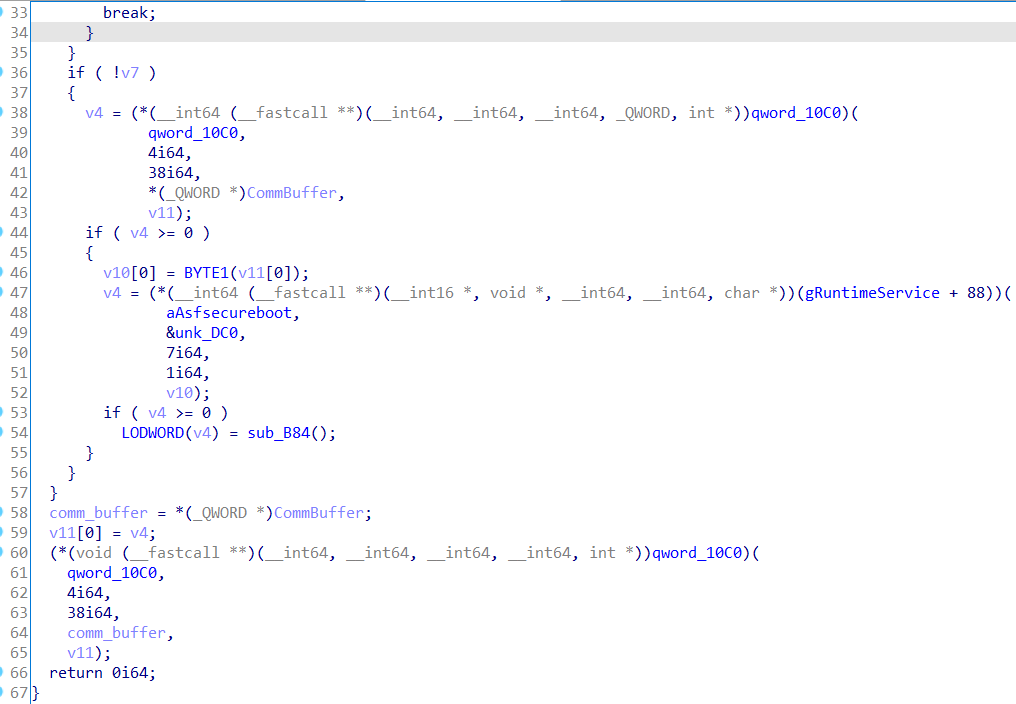


Рисунок 24 — Содержимое функции обработчика

Наличие ключевого слова CommBuffer сигнализирует об использовании протокола gEfiSmmCpuProtocolGuid со значением 0xeb346b97. Однако бинарный поиск указал на ту переменную, которая уже используется (Рисунок 25).

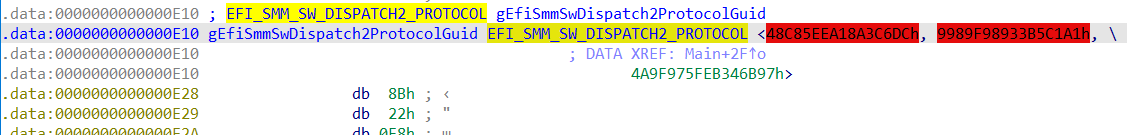


Рисунок 25 — Поиск протокола gEfiSmmCpuProtocolGuid

## Tcg2Smm

На рисунке 26 представлена точка входа в модуль.

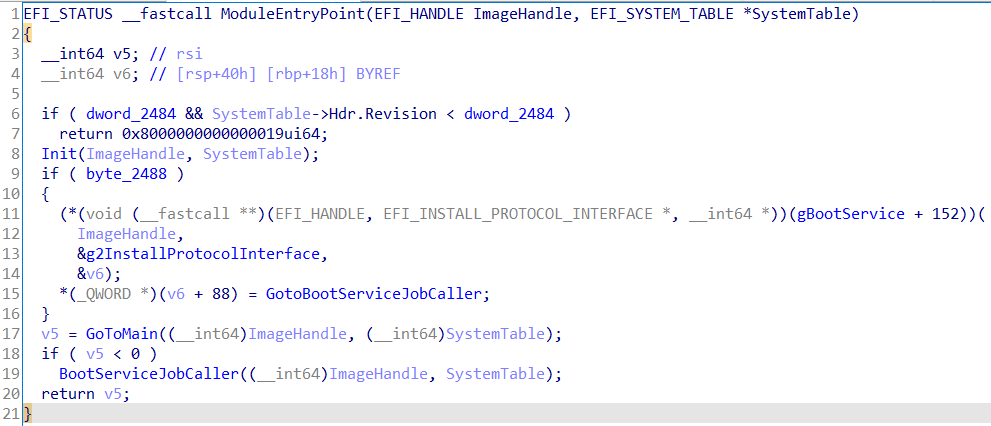


Рисунок — Точка входа в модуль

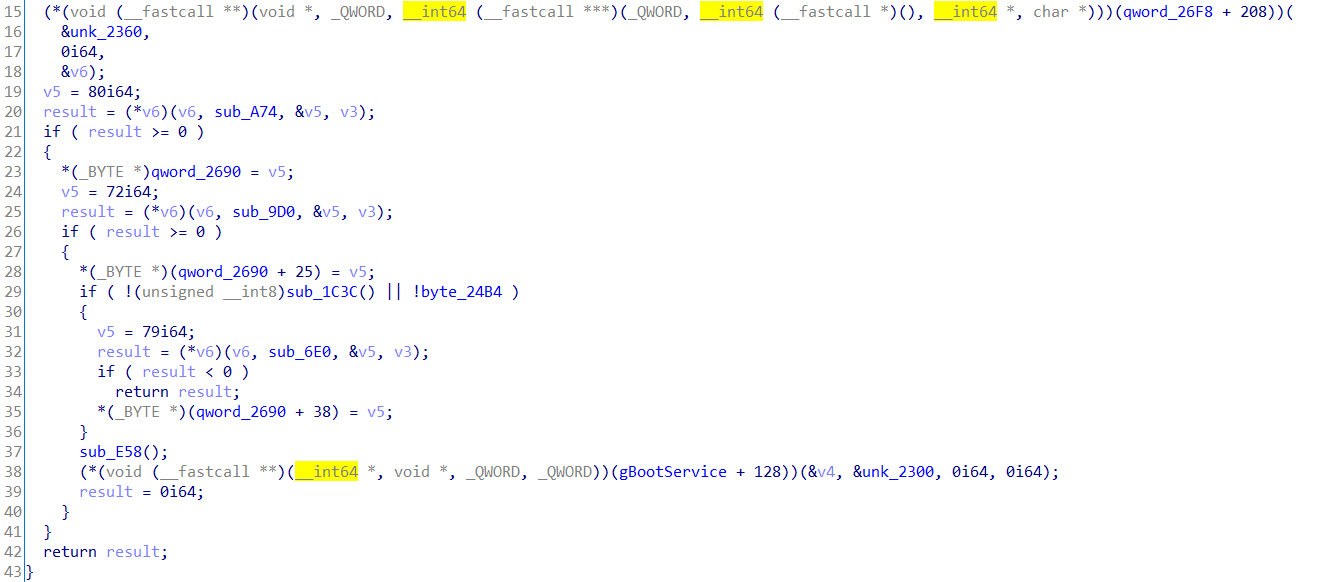
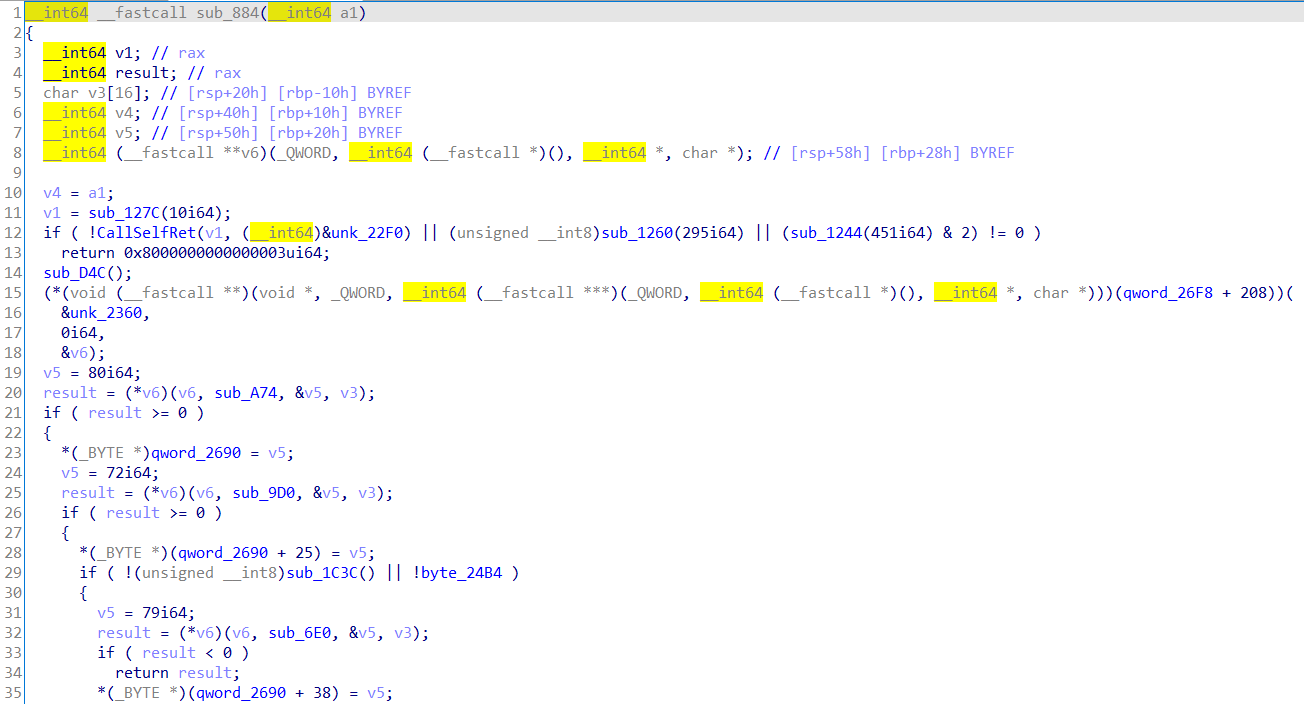


Рисунок — Содержимое функции Main

Найдем протокол gEfiSmmSwDispatch2ProtocolGuid по значению 0x18a3c6dc (находится в мейне). Установим все необходимые типы данных и переименуем переменные. Получим функцию обработки прерывания с номером 80 (Рисунок 28).

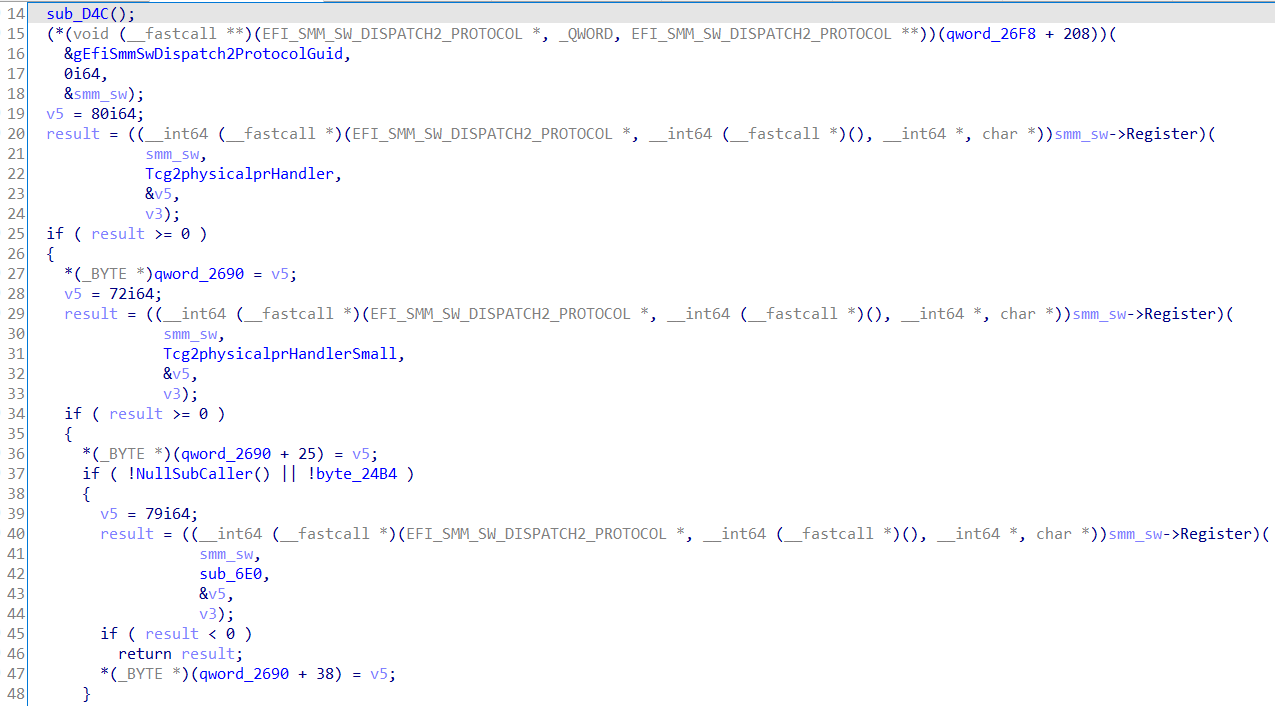


Рисунок 28 — Содержимое функции Main (обновленное)

Функция Tcg2physicalprHandler отвечает за обработку прерывания, связанного с проверкой и настройкой физического присутствия TPM. Реализация данной функции изображена на рисунке 29.

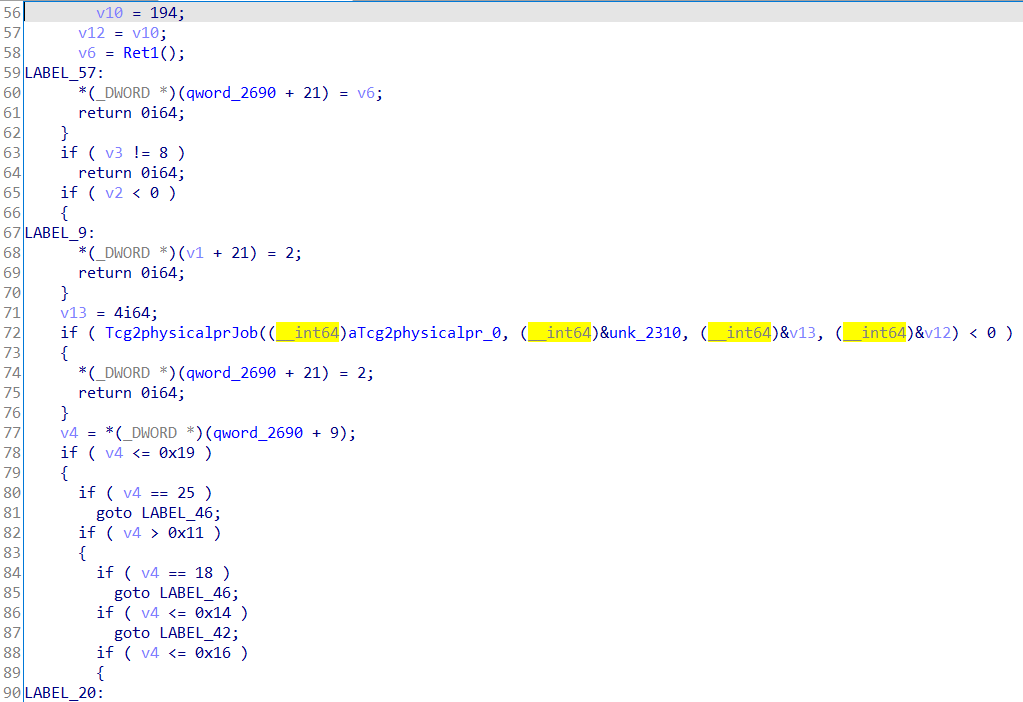


Рисунок 29 — Содержимое функции обработчика (1)

Злоумышленник может воздействовать на данный обработчик при помощи работы с NVRAM-переменной Tcg2PhysicalPresence. При помощи утилиты CHIPSEC было выяснено, что данная переменная может быть изменена в режим Runtime (рисунок 30).

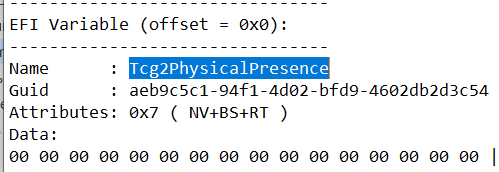


Рисунок 30 — Описание NVRAM-переменной Tcg2PhysicalPresence

Второй обработчик на рисунке 28 отвечает за очищение памяти. Реализация представлена на рисунке 31.

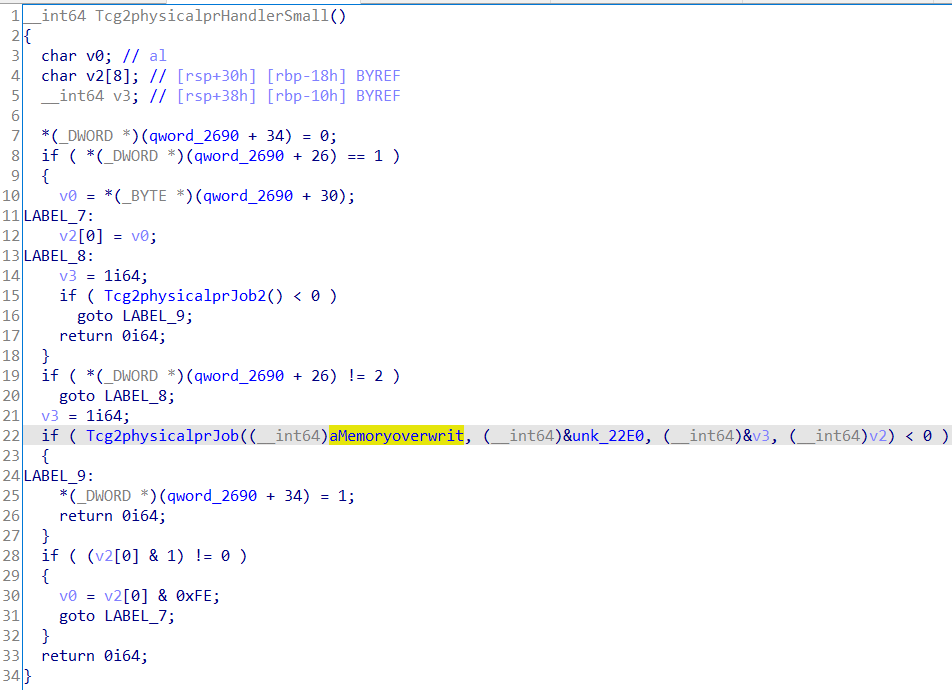


Рисунок 31 — Содержимое функции обработчика (2)

Злоумышленник может воздействовать на данный обработчик при помощи работы с NVRAM-переменной MemoryOverwriteRequestControl. При помощи утилиты CHIPSEC было выяснено, что данная переменная может быть изменена в режим Runtime (рисунок 32).

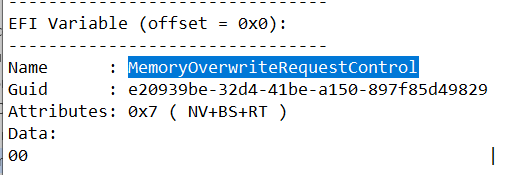


Рисунок — Описание NVRAM-переменной MemoryOverwriteRequestControl

# Вывод

В ходе выполнения лабораторной работы были изучены SMM модули: общая структура, порядок загрузки и способы перехода в режим SMM, в котором исполняются драйвера. С помощью полученных знаний были изучены SMM драйвера образа UEFI BIOS: из образа UEFI BIOS с помощью утилиты UEFITool были извлечены модули в виде PE32 файла, а с помощью IDA PRO были декомпилированы. Анализ декомпилированного кода SMM драйверов показал следующие особенности:

* драйвера имеют общую структуру и порядок исполнения;
* вызвать зарегистрированный драйвером обработчик можно с помощью SMI с определенным номером;
* передать данные SMI обработчику можно несколькими способами:

1. через CommBuffer;
2. через память;
3. через NVRAM-переменные.