ВЫКЛЮЧАЕМ INTEL ME 11, ИСПОЛЬЗУЯ НЕДОКУМЕНТИРОВАННЫЙ РЕЖИМ

Марк Ермолов
Максим Горячий

POSITIVE TECHNOLOGIES

ВЫКЛЮЧАЕМ INTEL ME 11, ИСПОЛЬЗУЯ НЕДОКУМЕНТИРОВАННЫЙ РЕЖИМ

В ходе исследования внутренней архитектуры Intel Management Engine (ME) 11-й версии был обнаружен механизм, отключающий эту технологию после инициализации оборудования и запуска основного процессора. О том, как мы нашли этот недокументированный режим, и о его связи с государственной программой построения доверительной платформы High Assurance Platform (HAP) мы расскажем в этой статье.

Авторы предупреждают, что использование данных знаний на практике может повлечь за собой повреждение вычислительной техники, и не несут за это никакой ответственности, а также не гарантируют работоспособность или неработоспособность чего-либо и не рекомендуют экспериментировать без наличия SPI-программатора.

ВВЕДЕНИЕ

Intel Management Engine — это закрытая технология, которая представляет собой интегрированный в микросхему Platform Controller Hub (PCH) микроконтроллер с набором встроенных периферийных устройств. Именно через PCH проходит почти все общение процессора с внешними устройствами, следовательно Intel ME имеет доступ практически ко всем данным на компьютере и возможность исполнения стороннего кода позволяет полностью скомпрометировать платформу. Такие безграничные возможности привлекают исследователей не первый год, но сейчас интерес к технологии Intel ME значительно вырос. Одной из причин этого является переход данной подсистемы на новую аппаратную (x86) и программную (доработанный MINIX в качестве операционной системы) архитектуру. Применение платформы x86 позволяет использовать всю мощь средств анализа бинарного кода, что ранее было затруднительно, так как до 11-й версии использовалось ядро с малораспространенной системой команд — ARC.

К сожалению, анализ Intel ME 11-й версии был затруднен тем, что исполняемые модули упакованы кодом Хаффмана с неизвестными таблицами. Но нашей исследовательской группе (Дмитрий Скляров, Марк Ермолов, Максим Горячий) удалось их восстановить (утилиту для распаковки образов можно найти на нашей странице в GitHub.

После распаковки исполняемых модулей мы приступили к изучению программной и аппаратной «начинки» Intel ME. Наша команда занимается этим уже достаточно продолжительное время, и мы накопили большой объем материалов, которые было решено опубликовать. Это первая статья из цикла статей, посвященных внутреннему устройству и особенностям работы Intel ME, и в ней мы расскажем, как отключить основной функционал подсистемы. Этот вопрос терзает специалистов, так как ее отключение позволило бы снизить риски утечки данных, например в случае обнаружения в этой технологии уязвимости нулевого дня.

КАК ВЫКЛЮЧИТЬ ME

Как выключить ME — этот вопрос часто задают некоторые владельцы компьютеров x86-архитектуры. Тема деактивации неоднократно поднималась, в том числе и исследователями нашей компании. [Intel ME. Как избежать восстания машин?, Neutralize ME firmware on SandyBridge and IvyBridge platforms, How to become the sole owner of your PC]. Актуальности этому вопросу добавляет недавно обнаруженная критическая (9,8 из 10) уязвимость в Intel Active Management Technology (AMT) — технологии, которая базируется на Intel ME.

Сразу огорчим читателя — полностью выключить ME на современных компьютерах невозможно. Это связано прежде всего с тем, что именно эта технология отвечает за инициализацию, управление энергопотреблением и запуск основного процессора. Сложности добавляет и тот факт, что часть кода «жестко прошита» внутри микросхемы PCH, которая выполняет функции южного моста на современных материнских платах. Основным средством энтузиастов, которые «борются» с данной технологией, является удаление всего «лишнего» из образа flash-памяти при сохранении работоспособности компьютера. Но сделать это не так просто, так как, если встроенный в PCH код не найдет во flash-памяти модули ME или определит, что они повреждены, система запущена не будет.

> For pre-Skylake firmware (ME version < 11) this tool removes almost everything, leaving only the two fundamental modules needed for the correct boot, ROMP and BUP. The code size is reduced from 1.5 MB (non-AMT firmware) or 5 MB (AMT firmware) to ~90 kB of compressed code.
>
> For Skylake and the later architectures (ME version >= 11), since the internal structure of the partitions is not yet known, the FTPR partition is left intact. The code size is reduced from 1.5 MB/5 MB to ~650 kB of compressed code.

Рисунок 1. Поддержка архитектур Skylake+ в me_cleaner

Уже несколько лет в сети развивается проект me_cleaner, в рамках которого доступна специальная утилита, позволяющая удалить большую часть образа и оставить только жизненно необходимые для основной системы компоненты. Но даже если система запустилась, радоваться рано — приблизительно через 30 минут может произойти автоматическое отключение, так как при некоторых сбоях ME переходит в Recovery-режим, в котором не функционирует больше некоторого фиксированного времени. В итоге процесс очистки усложняется. Например, до 11-й версии удавалось уменьшить размер образа до 90 КБ, но в 11-й — уже только до 650 КБ.

СЕКРЕТЫ В QRESOURCE

Intel дает производителям материнских плат возможность задать небольшое количество параметров ME. Для этого компания предоставляет производителям оборудования специальный набор программного обеспечения, в который входят такие утилиты, как Flash Image Tool (FIT) для настройки параметров ME и Flash Programming Tool (FPT), реализующая поддержку программирования flash-памяти напрямую через встроенный SPI-контроллер. Данные программы недоступны конечному пользователю, но их без труда можно найти в интернете.

```
C:\MEU>python C:\Python27\Scripts\binwalk meu.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
2810520       0x2AE298        XML document, version: "1.0"
2842816       0x2B60C0        Copyright string: "Copyright (c) "
2851456       0x2B8280        Zlib compressed data, default compression
2858473       0x2B9DE9        XML document, version: "1.0"
2860580       0x2BA624        Zlib compressed data, default compression
2867878       0x2BC2A6        Zlib compressed data, default compression
                              ...
```

Рисунок 2. Упакованные XML-файлы

Из этих утилит можно извлечь большое количество файлов формата XML (подробное описание процесса можно найти в докладе Intel ME: The Way of the Static Analysis, изучение которых позволяет узнать много интересного: структуру прошивки ME и описание PCH strap — специальных конфигурационных битов для различных подсистем, интегрированных в микросхему PCH. Нас заинтересовало одно из таких полей с именем «reserve_hap», так как напротив него имелся комментарий — High Assurance Platform (HAP) enable.

```
<LayoutEntry name="reserve_hap" type="bitfield32" value="0x0" offset="0x0" bitfield_high="16" bitfield_low="16" /> <!-- High Assurance Platform (HAP) enable -->
```

Рисунок 3. PCH strap для High Assurance Platform

Поиск в Google не был долгим. Буквально вторая ссылка говорит, что такое название носит программа по созданию доверительных платформ, связанная с Агентством национальной безопасности (АНБ) США. Презентацию с описанием программы можно найти тут. Нашей первой мыслью было поставить этот бит и посмотреть, что будет. Это может сделать любой желающий, если у него есть SPI-программатор или доступ в Flash Descriptor (на многих материнских платах некорректно выставлены права доступа к регионам flash-памяти).

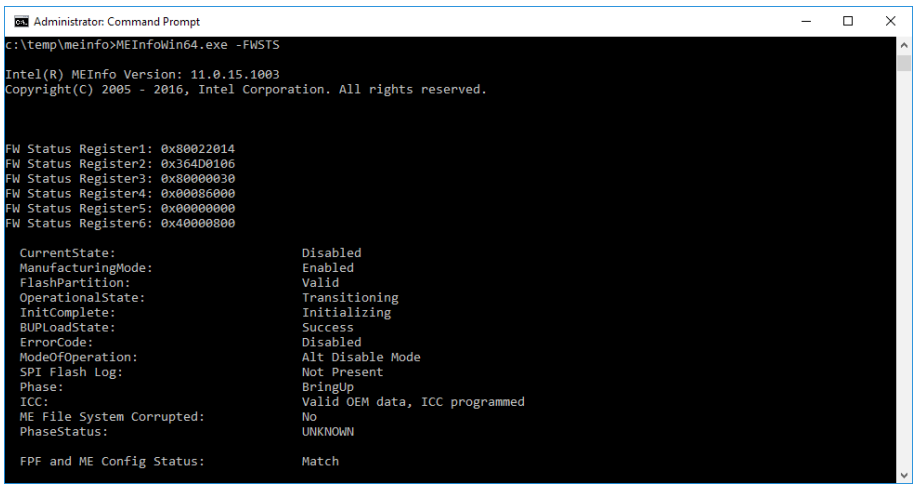

Рисунок 4. Статус ME после активации HAP-бита

После загрузки платформы утилита MEinfo сообщает странный статус — Alt Disable Mode. Беглые проверки показали, что ME не отвечает на команды и никак не реагирует на воздействия из операционной системы. Мы решили разобраться, как система переходит в этот режим и что он означает. К этому времени у нас уже была проанализирована основная часть модуля BUP, который отвечает за начальную инициализацию платформы и, исходя из вывода meinfo, устанавливает этот статус. Для понимания алгоритма работы BUP необходимо более подробно описать программное окружение Intel ME.

ПРОГРАММНАЯ ЧАСТЬ INTEL ME 11

Начиная с PCH 100-й серии компания Intel полностью переработала эту микросхему. Был осуществлен переход на новую архитектуру встроенных микроконтроллеров — с ARCompact компании ARC на x86. За основу был выбран 32-битный микроконтроллер Minute IA (MIA), который используется в микрокомпьютерах Intel Edison и SoC Quark. Он основан на дизайне весьма старого, скалярного микропроцессора Intel 486 с добавлением системы команд (ISA) от процессора Pentium. Однако для PCH компания выпускает данное ядро с применением 22-нм полупроводниковой технологии, получая высокую энергоэффективность микроконтроллера. Таких ядер в новом PCH три: Management Engine (ME), Integrated Sensors Hub (ISH) и Innovation Engine (IE). Последние два могут активироваться и деактивироваться в зависимости от модели PCH и целевой платформы, а ME-ядро работает всегда.

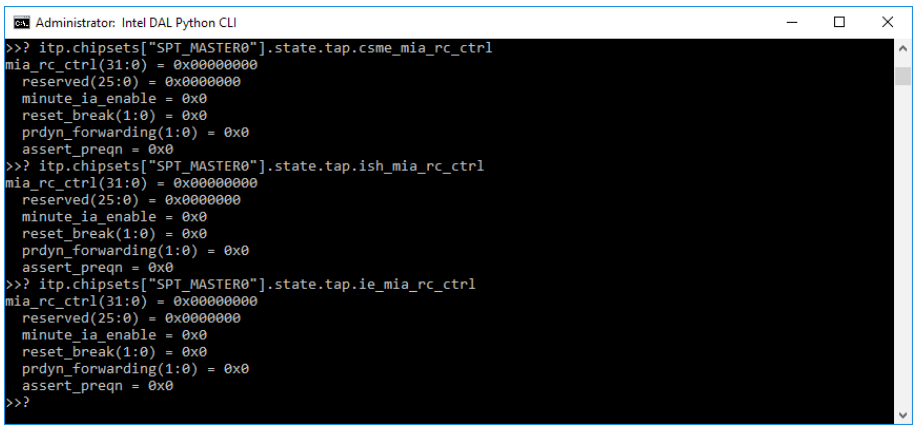

Рисунок 5. Три x86-процессора в PCH

Такие глобальные изменения потребовали изменения и программной составляющей ME. В частности, в качестве основы для операционной системы был выбран MINIX (ранее ThreadX RTOS). Теперь прошивка ME включает полноценную операционную систему со своими процессами, потоками, диспетчером памяти, драйвером аппаратных шин, файловой системой и многим другим. В ME интегрирован аппаратный криптопроцессор, поддерживающий алгоритмы SHA256, AES, RSA, HMAC. Доступ к оборудованию для пользовательских процессов производится посредством локальной таблицы дескрипторов (LDT). Через LDT организовано также и адресное пространство процесса — оно является всего лишь частью глобального адресного пространства ядра, границы которого заданы в локальном дескрипторе. Таким образом, ядру не нужно переключаться на память разных процессов (меняя каталоги страниц), как, например, в Microsoft Windows или Linux.

На этом закончим обзор программного окружения Intel ME и рассмотрим более подробно, как происходит загрузка операционной системы и модулей.

СТАДИИ ЗАГРУЗКИ INTEL ME

Запуск начинается с программы ROM, которая содержится во встроенной в PCH статической памяти. К сожалению, способ прочесть или перезаписать эту память широкой общественности не известен, но в интернете можно найти «предпродажные» версии прошивки ME с разделом ROMB (ROM BYPASS), который, по нашему предположению, дублирует функционал ROM. Таким образом, исследуя такие прошивки, можно восстановить основной функционал программы первичной инициализации.

Изучение ROMB позволяет понять назначение ROM — выполнение начальной инициализации оборудования, например SPI-контроллера, проверка цифровой подписи заголовка раздела FTPR, загрузка модуля RBE, который расположен уже во flash-памяти. RBE, в свою очередь, проверяет контрольные суммы модулей KERNEL, SYSLIB, BUP и передает управление на точку входа в ядро.

Следует заметить, это эти три сущности — ROM, RBE и KERNEL — выполняются на нулевом уровне привилегий (в ring-0) ядра MIA.

```
mnfProcExt = v5;
lockedRangesExt = rbe_find_key_vals(metadata, metaLen, 0xB);
if ( modIdx == 1 )
  return RbeProcessSyslibModule(spi_offset, metadata, metaLen, modAttr, lockedRangesExt);
if ( modIdx < 1 )
  return RbeProcessKernelModule(metadata, metaLen, modAttr, mnfProcExt, lockedRangesExt);
if ( modIdx == 2 )
  return RbeProcessBupModule(spi_offset, metadata, metaLen, modAttr, mnfProcExt, lockedRangesExt);
return 1;
```

Рисунок 6. Проверка целостности SYSLIB, KERNEL и BUP в RBE

Первый процесс, который создается ядром, — это BUP, который уже выполняется в своем адресном пространстве, в ring-3. Других процессов ядро по своей инициативе не запускает, этим занимается сам BUP, а также отдельный модуль LOADMGR, к нему вернемся позже. Назначение BUP (platform BringUP) — это инициализация всего аппаратного окружения платформы (в том числе процессора), выполнение первичных функций управления энергопитанием (например, запуск системы по нажатию кнопки включения) и запуск все остальных процессов ME. Таким образом, можно с уверенностью говорить, что PCH в 100-й серии и выше просто физически не имеют возможности запуска без корректной прошивки ME. Во-первых, BUP инициализирует контроллер управления энергопитанием (power management controller, PMC) и ICC-контроллер. Во-вторых — запускает целую вереницу процессов; часть из них «жестко прошита» в коде (SYNCMAN, PM, VFS), а другая часть содержится в InitScript (аналог автозапуска), который хранится в заголовке тома FTPR и защищен цифровой подписью.

```
1 char bup_start_syncman_pm()
2 {
3   bup_start_process("syncman", 1, 0x4000400u, 0, 0);
4   return bup_start_process("pm", 1, 0x1000100u, 0, 0);
5 }
```

Рисунок 7. Запуск SYNCMAN и PM

Таким образом, BUP считывает InitScript и запускает все процессы, которые удовлетворяют типу запуска ME и являются IBL-процессами.

```
if ( g_bup_vfs_started )
  goto skip_stage_check;
bup_start_vfs();
g_bup_vfs_started = 1;
for ( i = bup_is_cur_stage_block_exec_script_processing(); !i; i = bup_is_cur_stage_block_exec_script_processing() )
{
  while ( 1 )
  {
skip_stage_check:
    if ( g_bup_exec_sc_cur_mod >= g_bup_init_script->number_of_modules )
    {
      err = bup_write_snowball_file("ibl_list", g_bup_ibl_list, g_bup_cur_ibl_list_mod - g_bup_ibl_list, 0, 0, 6);
      goto cleanup_exit;
    }
    exec_sc_cur_mod = g_bup_exec_sc_cur_mod;
    if ( (g_bup_init_script->init_mod_entries[g_bup_exec_sc_cur_mod].init_flags & 5) == 1// IBL
      && g_bup_cur_exec_sc_init_flags & g_bup_init_script->init_mod_entries[exec_sc_cur_mod].init_flags
      && g_bup_cur_exec_sc_boot_type & g_bup_init_script->init_mod_entries[exec_sc_cur_mod].boot_type )
    {
      break;
    }
    ++g_bup_exec_sc_cur_mod;
  }
  LOBYTE(err) = bup_start_process(g_bup_init_script->init_mod_entries[exec_sc_cur_mod].name, 0, 0, 0, 0);
  if ( err )
    goto cleanup_exit;
  sys_strncpy_s(g_bup_cur_ibl_list_mod, 0xC, g_bup_init_script->init_mod_entries[g_bup_exec_sc_cur_mod].name, 0xC);
  g_bup_cur_ibl_list_mod[0xB] = 0;
  g_bup_cur_ibl_list_mod += 0xC;
  ++g_bup_exec_sc_cur_mod;
}
```

Рисунок 8. Обработка InitScript

```
Ext#1 InitScript[41]:
   1: FTPR:kernel       Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   2: FTPR:syslib       Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   3: FTPR:rbe          Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   4: FTPR:bup          Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   5: FTPR:evtdisp      Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
   6: FTPR:busdrv       Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
   7: FTPR:prtc         Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
   8: FTPR:crypto       Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
   9: FTPR:storage      Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
  10: FTPR:fpf          Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
  11: FTPR:loadmgr      Init: 00010001 (Ibl, Cm0_u) Boot: 00000011 (Normal, Recovery)
  12: NFTP:mca_boot     Init: 00010004 (InitImmediately, Cm0_u) Boot: 00000001 (Normal)
```

Рисунок 9. Список модулей с флагом IBL

В случае если запуск процесса не удался, BUP не будет запускать систему или переведет ее в Recovery режим, в котором произойдет автоматическое отключение питания после нескольких десятков минут. Как можно видеть на иллюстрации, последним в списке IBL-процессов является LOADMGR. Именно он дает старт оставшимся процессам, но в отличие от BUP, если в процессе запуска модуля происходит ошибка, LOADMGR просто перейдет к следующему.

Таким образом, первый вариант ограничить функционирование Intel ME — удалить все модули, которые не имеют флаг IBL в InitScript, что позволит существенно уменьшить размер прошивки. Но первоначально мы хотели выяснить, что происходит с ME в режиме HAP. Для этого рассмотрим программную модель BUP подробнее.

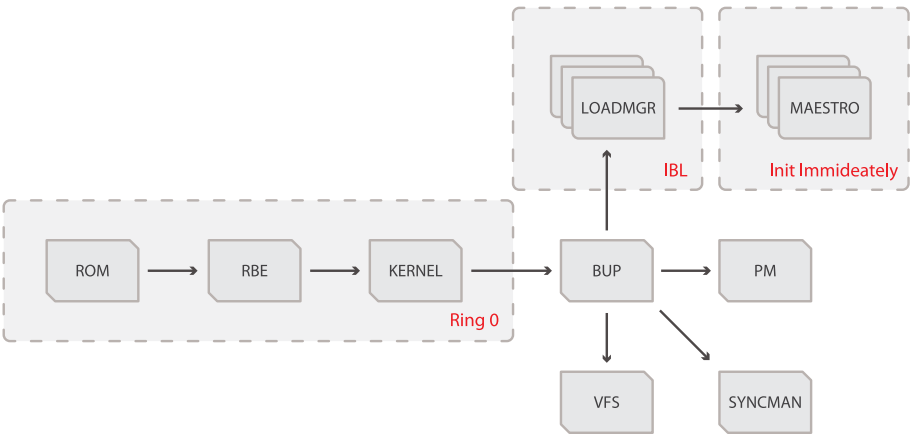

Рисунок 10. Схема запуска модулей в ME

BRINGUP

Если присмотреться к алгоритму работы модуля BUP, можно сказать, что внутри него реализован классический конечный автомат. Выполнение функционально разделено на две составляющие: стадии инициализации (представляют собой тот самый конечный автомат) и выполнение сервисных запросов других процессов после инициализации системы. Число стадий инициализации разное, в зависимости от платформы и SKU (TXE, CSME, SPS, consumer, corporate), но основные, общие для всех версий, все же можно выделить.

Первая стадия

На начальной стадии происходит создание внутренней диагностической файловой системы sfs (SUSRAM FS — файловая система, расположенная в энерго-зависимой памяти), считывание конфигурации, и, самое главное, получение информации от PMC о том, что привело к данному старту — включение питания платформы, глобальный перезапуск всей платформы, перезапуск только ME или же пробуждение из состояния сна. Эта стадия называется boot flow determination. От нее зависят последующие стадии работы конечного автомата инициализации. Кроме того, поддерживаются несколько режимов работы: нормальный и набор сервисных режимов, при которых основной функционал ME не выполняется — HAP, HMRFPO, TEMP_DISABLE, RECOVERY, SAFE_MODE, FW_UPDATE и FD_OVERRIDE.

Вторая стадия

На следующей стадии происходят инициализация ICC-контроллера и загрузка ICC-профиля (отвечает за тактовые частоты основных потребителей), инициализация Boot Guard и начало циклического опроса подтверждения запуска процессора.

Третья стадия

BUP ожидает сообщение от PMC о том, что основной процессор запустился. После этого BUP запускает асинхронный цикл опроса PMC на предмет возникновения событий энергопитания (перезапуск или отключение платформы) и переходит к следующей стадии. Если такое событие произошло, BUP выполнит запрашиваемое действие в момент перехода между стадиями инициализации.

Четвертая стадия

На этой стадии происходит инициализация внутреннего оборудования. Также BUP запускает цикл опроса heci (специального устройства, предназначенного для получения команд от BIOS или операционной системы) на предмет получения DID (DRAM Init Done message) от BIOS. Именно это сообщение позволяет ME понять, что основной BIOS инициализировал оперативную память и зарезервировал для ME специальный регион, UMA, и после этого перейти к следующей стадии.

Пятая стадия

Как только DID получен, BUP, в зависимости от режима работы, который определяется по разным составляющим, либо запускает IBL-процессы из InitScript (при нормальном режиме работы), либо зависает в цикле, выйти из которого он может только при получении сообщения от PMC, например в результате запроса на перезагрузку или выключение системы.

Именно на этой стадии мы и находим обработку HAP, причем в этом режиме BUP не выполняет InitScript, а зависает. Таким образом, остальная последовательность действий при нормальном режиме работы не имеет отношения к HAP и нами рассматриваться не будет. Главное, что хочется отметить: в режиме HAP BUP выполняет всю инициализацию платформы (ICC, Boot Guard), но не запускает основные процессы ME.

```
void __cdecl bup_check_hap()
{
  BUP_RUNTIME_CTX *rt_ctx; // edx@3
  char boot_type; // al@3
  int pch_strap_0; // [esp+0h] [ebp-8h]@1
  int cookie; // [esp+4h] [ebp-4h]@1

  cookie = gRmlbCookie;
  if ( !bup_get_pch_straps(0, &pch_strap_0) && BYTE2(pch_strap_0) & 1 )
  {
    rt_ctx = g_bup_rt_ctx_ptr;
    boot_type = g_bup_rt_ctx_ptr->boot_type;
    g_bup_rt_ctx_ptr->pm_ctx.field_9 = 4;
    LOBYTE(rt_ctx->boot_type) = boot_type & 0xFE | INIT_SCRIPT_BOOT_HAP;
  }
  if ( gRmlbCookie != cookie )
    sys_fault();
}
```

Рисунок 11. Определение режима HAP

```
else if ( g_bup_rt_ctx_ptr->boot_type & (INIT_SCRIPT_BOOT_TYPE_FD_OVERRIDE|INIT_SCRIPT_BOOT_TEMP_DISABLE|INIT_SCRIPT_BOOT_HMRFPO|INIT_SCRIPT_BOOT_HAP|0x80)
      || g_bup_rt_ctx_ptr->state_sec_boot < 0 )
{
  bup_change_stage(5u);
  g_bup_rt_ctx.prev_stage = g_bup_rt_ctx.cur_stage;
  g_bup_rt_ctx.cur_stage = 5;
  bup_heci1_set_fwsts_error_code(2);
  bup_set_heci1_fwsts_working_state(FWSTS_WS_DISABLED);
  bup_set_heci1_fwsts2_bup_phase_status(0x28);// CM0_TEMP_DISABLE
}
```

Рисунок 12. Перевод ME в пятую стадию, что равноценно зависанию

```
void __cdecl bup_update_heci1_fwsts_disabled_operating_mode()
{
  char boot_type; // al@1

  boot_type = g_bup_rt_ctx_ptr->boot_type;
  if ( boot_type & INIT_SCRIPT_BOOT_HAP )
  {
    bup_set_heci1_fwsts_operation_mode(2);      // Alt Disable Mode
  }
  else if ( boot_type & 0x88 )
  {
    bup_set_heci1_fwsts_operation_mode(3);      // Temporary Disable mode
    if ( g_bup_rt_ctx_ptr->field_1C < 0 )
      bup_set_heci1_fwsts2_bup_phase_status(0x47);// MFG_CMRST
  }
  else if ( boot_type & INIT_SCRIPT_BOOT_HMRFPO )
  {
    bup_set_heci1_fwsts_operation_mode(5);      // Unsecured mode by HECI message
  }
  else if ( BYTE1(g_bup_rt_ctx_ptr->boot_type) & 1 )
  {
    bup_set_heci1_fwsts_operation_mode(4);      // Unsecured mode by H/W jumper
  }
}
```

Рисунок 13. Пятая стадия

УСТАНОВКА HAP-БИТА

Исходя из вышесказанного, второй вариант отключения состоит в установке HAP-бита и удалении или повреждении всех модулей, кроме тех, которые необходимы BUP для старта — RBE, KERNEL, SYSLIB, BUP. Сделать это можно просто убрав их из CPD-раздела FTPR и пересчитав контрольную сумму заголовка CPD (подробнее структура прошивки ME (подробнее: Intel ME: The Way of the Static Analysis).

Остается еще один вопрос: как установить этот бит? Можно воспользоваться конфигурационными файлами FIT и определить, где он расположен в образе, но есть путь проще. Если открыть FIT, то в секции ME Kernel можно найти некий параметр Reserved. Именно этот бит и отвечает за включение режима HAP.

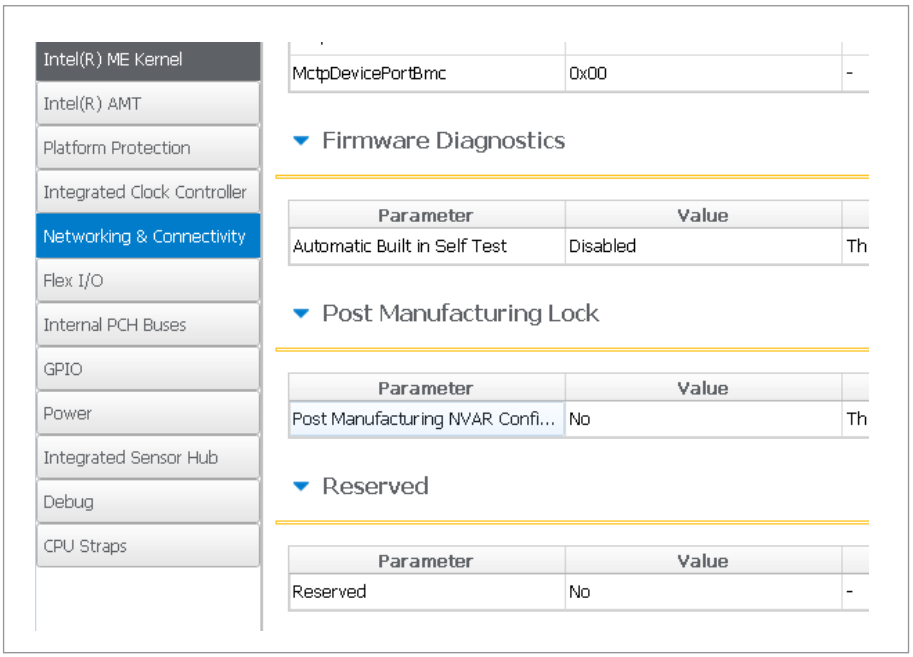

Рисунок 14. Бит активации режима HAP

HAP И BOOT GUARD

Нами также был найден код в BUP, который при активированном режиме HAP устанавливает дополнительный бит в политиках Boot Guard. К сожалению, выяснить, чем управляет этот бит, нам пока не удалось. HAP и Boot Guard

```
if ( bup_get_boot_type() & INIT_SCRIPT_BOOT_HAP )
{
  g_bup_sb_oem_data.poly_type[0] |= 4u;
  bup_thunk9();
}
```

Рисунок 15. Установка дополнительного бита для Boot Guard

ПОДДЕРЖКА ME 11 В ME_CLEANER

Пока эта статья готовилась к печати, разработчики обновили me_cleaner, в результате чего он стал так же удалять из образов все модули, кроме RBE, KERNEL, SYSLIB и BUP, но без установки HAP-бита, что вводит ME в режим «TemporaryDisable». Нам стало любопытно, что происходит при таком подходе.

Мы выяснили, что удаление разделов с файловой системой ME приводит к ошибке при чтении файла cfg_rules. Данный файл содержит ряд различных настроек системы. Среди них, как мы полагаем, есть флаг, который мы назвали «bup_not_temporary_disable». Если он не установлен, вся подсистема переводится в режим TemporaryDisable, а так как этот флаг — глобальная переменная, по умолчанию инициализированная нулем, то ошибка чтения расценивается как конфигурация, требующая отключения.

Отметим также, что мы проверили также прошивки от серверной и мобильной версии ME (SPS 4.x и TXE 3.x). В серверной версии этот флаг всегда устанавливается в 1, а в мобильно не анализируется. Из вышесказанного следует, что данный способ не сможет работать на серверных и мобильных версиях (Apollo Lake) ME.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Таким образом, мы нашли недокументированный PCH strap, который позволяет перевести Intel ME в режим отключения основного функционала на ранней стадии. Хотя физическое удаление модулей из образа с сохранением работоспособности неявно доказывает, что этот режим производит отключение ME, бинарный анализ не оставляет поводов для сомнений. С большой долей уверенности можно сказать, что выйти из этого режима Intel ME уже не в состоянии, так как в модулях RBE, KERNEL и SYSLIB не найдено функционала, который позволял бы это осуществлять.

Также мы считаем, что ROM, интегрированный в PCH, практически не отличается от ROMB, в котором тоже не найдено ничего похожего. Таким образом, HAP позволит защититься от уязвимостей, присутствующих во всех модулях, кроме RBE, KERNEL, SYSLIB, ROM и BUP, но, к сожалению, от эксплуатации ошибок на более ранних этапах этот режим не предохранит.

Мы ознакомили представителей Intel с деталями исследования. Их ответ подтвердил нашу догадку о связи недокументированного режима с программой High Assurance Platform. С разрешения компании приведем отрывок из ответа:

"Mark/Maxim,

In response to requests from customers with specialized requirements we sometimes explore the modification or disabling of certain features. In this case, the modifications were made at the request of equipment manufacturers in support of their customer's evaluation of the US government's "High Assurance Platform" program. These modifications underwent a limited validation cycle and are not an officially supported configuration."

Мы полагаем, что данный механизм — удовлетворение обычной просьбы любой правительственной службы, которая хочет уменьшить вероятность утечки по побочным каналам. Но остается главный вопрос: как HAP влияет на функционирование Boot Guard? Из-за закрытости этой технологии ответить на этот вопрос пока не представляется возможным, но мы надеемся, что в скором будущем нам это удастся.

Горячий Максим, Ермолов Марк