Исследование возможностей несанкционированного управления промышленными коммутаторами для контроля над ТП

А.В. Ермолов (Компания Digital Security)

Показано, что промышленные коммутаторы являются критическим элементом структуры промышленной сети, поэтому они должны быть надежно защищены от компрометации потенциальным злоумышленником. На примере подробного анализа двух коммутаторов от известнейших производителей промышленного сетевого оборудования продемонстрированы возможности обновлять и подделывать образ прошивки устройств; выявлено отсутствие механизмов от эксплуатации бинарных уязвимостей.

Ключевые слова: промышленные коммутаторы, компрометация, злоумышленник, бинарные уязвимости, обновление образа прошивки.

Введение

Сердцем любой современной инфраструктуры АСУТП является среда передачи данных. В подавляющем большинстве случаев она представляет собой промышленную сеть на основе семейства технологий Ethernet (то есть сеть типа Industrial Ethernet). Такие сети от обычных отличает, прежде всего, использование протокола реального времени, например, Profinet, EtherCAT, Ethernet/IP и пр. [1]. Пример промышленной сети изображен на рис. 1.

Различного рода устройства (ПЛК, панели ЧМИ, управляющие ПК, полевые устройства и пр.) к сети подключаются через сетевые коммутаторы (свитчи) промышленного типа. В отличие от обычных коммутаторов они менее требовательны к условиям эксплуатации (расширенный диапазон рабочих температур, защита от перепадов напряжения, ударопрочность, возможность крепления на DIN-рейку и т.д.) и поддерживают промышленные сетевые протоколы.

Таким образом, промышленные коммутаторы являются критическим элементом структуры промышленной сети. Очень важно, чтобы они были должным образом защищены от компрометации потенциальным злоумышленником. В противном случае последний получает практически безграничные возможности контроля над ТП: компрометацию других промышленных коммутаторов в сети (или других устройств); вмешательство и изменение данных внутри различных соединений между ПЛК и SCADA, между шлюзами и ПЛК; подделывание данных, передаваемых на ЧМИ и в системы журналирования и т.д. Все это может привести к невозможности оператору контролировать реальное состояние ТП, что чревато остановкой ТП или аварией [2].

Для предотвращения подобного сценария территориальное размещение коммутатора должно сводить к минимуму возможность несанкционированного физического доступа к нему. Кроме того, программно-аппаратная архитектура коммутатора должна иметь модель информационной безопасности, учитывающую современные угрозы. Рассмотрим последний тезис подробнее.

Для исследования выбраны по одной модели управляемых промышленных коммутаторов от двух самых распространенных (как в РФ, так и за рубежом) производителей промышленного сетевого оборудования. Назовем их SWITCH 1 и SWITCH 2.

SWITCH 1 имеет Ethernet гнезда (4, 8, 16 или 24 ед. — зависит от конкретной модели) типа RJ-45, разъем RS-232 типа RJ-11 и разъем USB.

SWITCH 2 обладает модульным строением: в него можно подключать слоты расширения. В слоты расширения вставляются модули с гнездами Ethernet (RJ-45 либо оптоволокно), есть разъем RS-232 типа mini DIN.

Управлять работой этих свитчей можно через COM-порт или на сетевом уровне (модели OSI). При работе через COM-порт доступен интерфейс командной строки. Для использования основного набора команд требуется аутентификация (логин и пароль). Однако некоторые опции (вывод на экран информации об устройстве, возможность обновления прошивки и т.п.) доступны до экрана аутентификации.

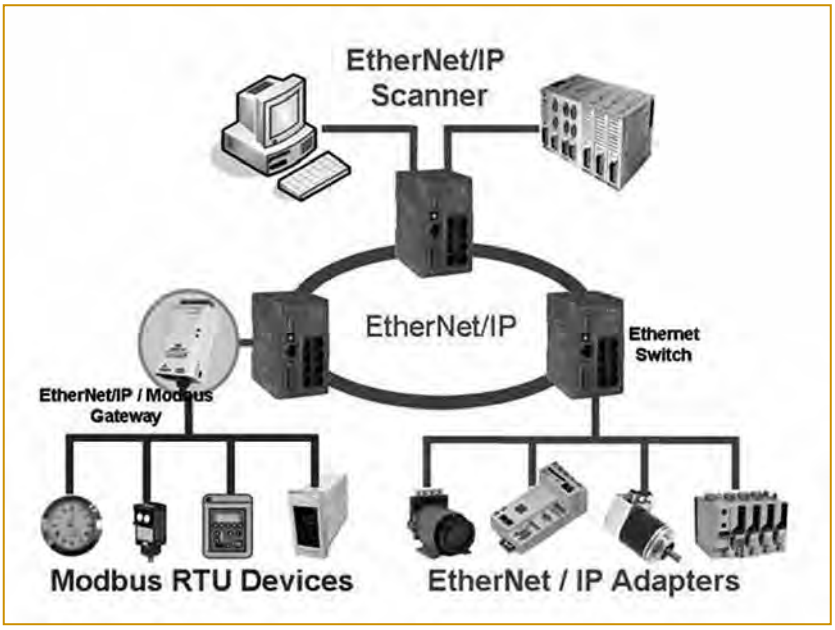

Рис. 1. Образец промышленной сети типа Industrial Ethernet

Удаленное управление осуществляется посредством Web-интерфейса или протокола SNMP (Simple Network Management Protocol), для этого всегда требуется аутентификация.

Следует отметить, что поддержка SNMP в исследуемых свитчах имеет несколько недостатков:

— по умолчанию (а это самая любимая конфигурация большинства сетевых устройств) используется самая небезопасная версия этого протокола — SNMP v1;

— в SNMP v1 производитель устройств настоятельно не рекомендует изменять применяемые по умолчанию логин и пароль;

— в версиях SNMP v1 и SNMP v2c не используется шифрование, а значит, протокол уязвим к атакам типа MITM (man in the middle).

Исследование промышленного коммутатора SWITCH 1

Анализ общей архитектуры

Поиск уязвимостей таких устройств обычно начинают с их встраиваемого ПО (прошивки). Для оценивания среды исполнения кода, а также возможности подключения программатора/отладчика рассмотрим состав компонентов на системной плате:

1. ЦП Digi NET+ARM NS9360B-0-I155, 32-разрядный ARM9, без внутренних запоминающих устройств (следовательно, прошивка должна храниться во внешней памяти);

2. оперативная память SDRAM Micron MT48LC8M16A2 на 16 МБ;

3. флэш-память Intel 28F640JD3D75 на 8 МБ (скорее всего, прошивка здесь);

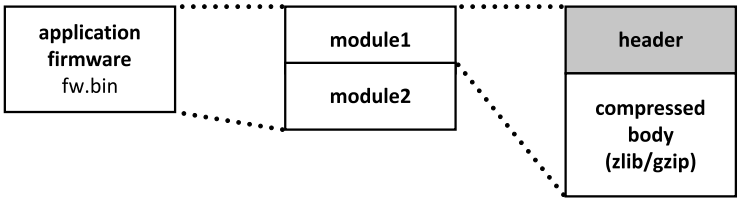

Рис. 2. Общая структура образа прошивки SWITCH 1

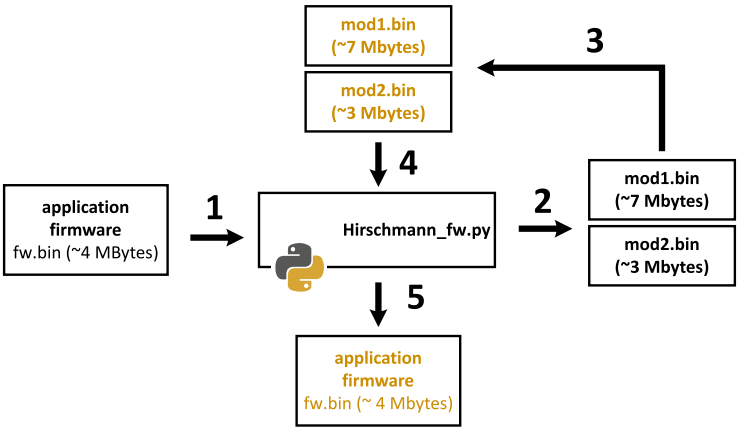

Рис. 3. Работа со скриптом для распаковки и упаковки прошивок SWITCH 1

4. CPLD Marvell 88E6095F-LG01 выполняет функцию Ethernet свитча, имеет внутреннюю конфигурационную память.

Анализ образа прошивки

Образ прошивки для этого коммутатора был найден на открытом ftp-сервере производителя. Один из файлов в скачанном архиве (около 4 МБ) — это образ прошивки. Анализ этого файла показал, что он состоит из двух модулей (рис. 2). Каждый модуль имеет заголовок (их структура одинакова), а также сжатое тело.

Первый модуль сжат по алгоритму zlib (RFC 1951). Именно в нем хранится исполнимый код. В распакованном виде модуль весит примерно 7 МБ.

Второй модуль сжат по алгоритму gzip (RFC 1952), в нем находится pack200-архив, после распаковки которого получается JAR (Java Archive) файл, представляющий собой реализацию Web-интерфейса устройства. После распаковки модуль весит примерно 3 МБ. Следует отметить, что этот файл передается на хост (управляющий ПК) при обращении к коммутатору по его IP-адресу и исполняется именно на нем (на хосте).

Суммарный объем распакованных модулей составляет 10 МБ, а объем флэш-памяти на системной плате устройства — 8 МБ. Предположим, что образ прошивки хранится в запакованном виде. А значит, есть загрузчик, который эту прошивку распаковывает и запускает.

Размер каждого заголовка модуля составляет 256 байт. Методом проб и ошибок мы почти полностью разобрали его структуру (она нигде не документирована) и увидели в ней следующие интересные поля:

1) сигнатура модуля;
2) тип модуля;
3) размер модуля;
4) контрольная сумма модуля (CRC32);
5) смещение до конца файла (eof offset);
6) контрольная сумма модуля (повтор) (CRC32);
7) контрольная сумма заголовка (CRC32).

Исходя из этого, можно сделать вывод о наличии контроля целостности прошивки. Но контроль подлинности, обычно это ЭЦП (электронно-цифровая подпись), отсутствует.

Для проверки возможности подделывать прошивку был написан небольшой скрипт (рис. 3), позволяющий быстро извлекать оба модуля из образа прошивки и после внесения в них изменений собирать обратно в готовый образ.

В процессе анализа исполнимого кода прошивки было выявлено нахождение в ней кода ОС РВ VxWorks, причем довольно древней версии 5.4.2 (где-то 1998 г.). Анализ «исходников» ОС РВ, найденных на просторах Internet, показал наличие базовых компонентов ОС, модулей Web-сервера (EmWeb), SnmpOverHttp и пр.

О наполнении разработчик SWITCH 1 определенно позаботился. А вот о безопасности

не очень. Устройство допускает исполнение программного кода из любого места, отсутствует защита от переполнений на стеке, используются фиксированные адреса. То есть элементарные механизмы защиты от эксплуатации бинарных уязвимостей отсутствуют напрочь. Хотя для данной версии VxWorks известно более 10 зарегистрированных уязвимостей.

В ходе анализа прошивки были найдены фрагменты кода:

1) обработчики SNMP-запросов;

2) обработчики консольных команд;

3) чтение и перезапись флэш-памяти;

4) чтение и перезапись конфигурационной памяти CPLD Marvell!

Имея всю описанную информацию, можно модифицировать прошивку коммутатора. Для этого следует грамотно выбрать место для внедрения кода: нельзя допустить, чтобы он препятствовал нормальному функционированию устройства, и лучше всего, если он будет вызываться по требованию.

Для проведения модификации был выбран один из обработчиков консольных команд — обработчик команды logout. На его место в прошивке мы записали код, считывающий память заданного размера по заданному адресу (все задается в переменных в конце вставленного фрагмента) и выводящий дамп в COM-порт.

Далее был подготовлен готовый образ прошивки, который был залит в коммутатор через COM-порт. Проверяем: после ввода команды logout вместо выхода из системы теперь выводился дамп памяти, — возможность подделывать прошивки экспериментально подтверждена.

Теперь рассмотрим, как злоумышленник может внедрить свою прошивку в устройство, реально использующееся на промышленном объекте. Первый вариант — через COM-порт. В этом случае не требуется знать логин и пароль, но требуется иметь физический доступ к устройству (чтобы подключиться к соответствующему разъему). В этот вариант слабо верится, если устройство уже используется на объекте. Тем не менее, с завода-изготовителя устройство сразу не попадает на промышленный объект. В схеме доставки до заказчика всегда есть промежуточные организации, которые имеют возможность, в качестве бесплатного бонуса, загрузить на коммутатор неоригинальную прошивку. Также обновить прошивку коммутаора можно с предварительно скомпрометированного управляющего ПК, который уже подключен к устройству по COM-порту.

Удаленно модифицировать прошивку сложнее, но векторов атак здесь больше. Для доступа к Web-интерфейсу коммутатора требуется аутентификация. Можно попробовать:

1) ввести известные логин и пароль «по умолчанию»;

2) подобрать с учетом известных ограничений на пароль для конкретной модели, так как защиты от brute-force атак здесь нет;

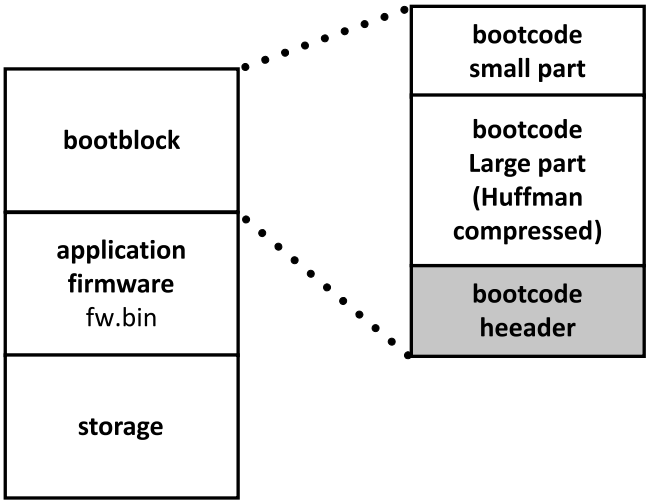

Рис. 4. Структура флэш-памяти SWITCH 1

3) эксплуатировать бинарную уязвимость (известную или честно найти свою), опять же, защиты от эксплуатации бинарных уязвимостей нет.

Кстати, удаленно, это не только с ПК, подключенного к коммутатору, а также с одного из полевых устройств, которые могут находиться на неохраняемой территории, а значит, подлежат обязательной компрометации.

Таким образом, потенциальный злоумышленник при внедрении своего кода в прошивку коммутатора имеет возможность исполнять свой код на:

1) ЦП коммутатора (используя весь спектр его аппаратных ресурсов);

2) стороне хоста (управляющего ПК) или любого другого клиентского ПК, случайно обратившемуся по IP-адресу коммутатора. Конечно, этот код будет исполняться в Java машине, но она, как известно, не оплот безопасности.

Заметим, что скомпрометированный коммутатор можно «вылечить» путем штатной процедуры обновления прошивки. Главное быть уверенными в подлинности нового образа.

Анализ загрузчика прошивки

Возможность «вылечить» коммутатор наводит на вопрос о необходимости закрепления модификаций на устройстве так, чтобы вносимые в прошивку изменения «переживали» бы любые обновления. Здесь поможет загрузчик образа.

Понадеявшись, что загрузчик оставил след в оперативной памяти, мы искали его при помощи вышеописанной модификации в разных местах, но загрузчика нигде не было. Решили попробовать «достать» его из флэш-памяти коммутатора.

Ранее найденные процедуры для чтения флэш-памяти помогли получить ее полный дамп. Итак, содержимое флэш-памяти можно условно разделить на три региона (рис. 4):

1. Bootblock — это первые 80000h байт, загрузочная область, где хранится загрузчик, состоит из: стартового кода загрузчика, основного кода загрузчика

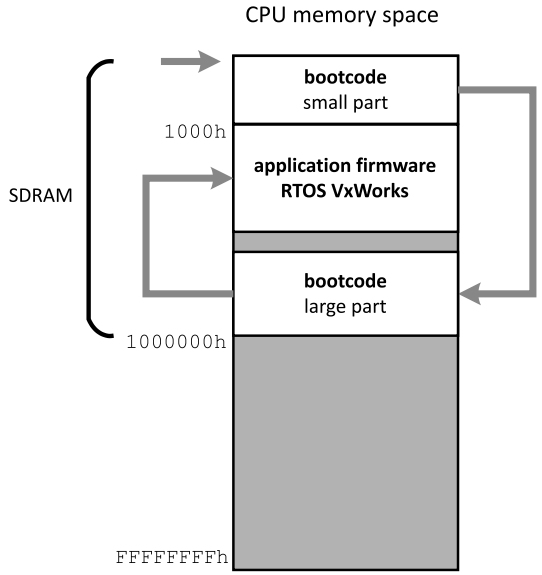

Рис. 5. Процесс старта прошивки SWITCH 1

(сжат по алгоритму Хаффмана), заголовка загрузчика (структура идентична ранее описанной структуре заголовка модуля прошивки);

2. Application firmware — в этой области хранится образ прошивки в упакованном виде (как и предполагалось);

3. Storage — область для хранения служебной информации, конфигураций, логов и т. д.

Код загрузчика позволил реконструировать ранние этапы процесса загрузки коммутатора (рис. 5). При старте первые 4 КБ флэш-памяти (а это как раз размер стартового кода) проецируются в адресное пространство CPU по адресу 0. С этого адреса начинается исполнение. Задача стартовой части загрузчика заключается в конфигурировании карты памяти, распаковке основной части загрузчика в память и передаче ей управления.

Основной код загрузчика должен: проинициализировать аппаратные ресурсы ЦП; сконфигурировать модель прерываний; распаковать в память прошивку и передать ей управление.

Внесение модификаций в загрузчик производится по следующей схеме (рис. 6 и 7): коммутатор обновляется модифицированной версией прошивки, которая, используя штатные процедуры для работы с флэш-памятью, перепишет область bootblock.

В дальнейшем модифицированный загрузчик будет иметь возможность при каждой загрузке вносить изменения в распакованный образ прошивки. Скомпрометированный таким образом коммутатор «вылечить» уже не так просто. Для этого надо обновить на оригинальные и прошивку, и загрузчик.

В консольном интерфейсе возможности обновлять загрузчик нет, хотя в коде прошивки были найдены процедуры, предназначенные для обновления области bootblock таким же образом, как обновляется прошивка. Но эти функции нигде не вызываются, стало быть это недокументированный отладочный функционал.

А вот в Web-интерфейсе устройства была найдена штатная опция для обновления загрузчика. Но проблема в том, что обновлять нечем. В архиве с образом прошивки для многих моделей своих устройств разработчики не прикладывают образ загрузчика. И исследуемая модель в их числе.

Таким образом, потенциальный злоумышленник имеет возможность через модификацию прошивки закрепляться в загрузчике устройства. И удалить эту модификацию оттуда штатными средствами практически невозможно.

Но можно пойти еще дальше. У анализируемого устройства есть еще одна исполнимая среда со своей прошивкой. Это ПЛИС (программируемая логическая интегральная схема), типа CPLD от Marvell с внутренней энергонезависимой конфигурационной памятью. Прошивка для них пишется на языке описания логики (VHDL, Verilog — самые распространенные). Но не будем останавливаться на этой возможности компрометации. Перейдем к анализу SWITCH 2.

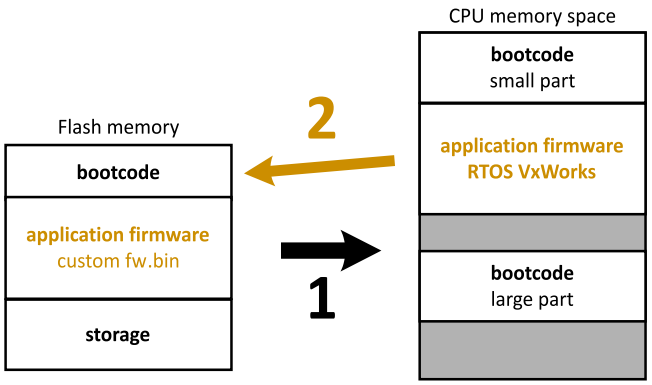

Рис. 6. Схема модификации загрузчика, часть 1

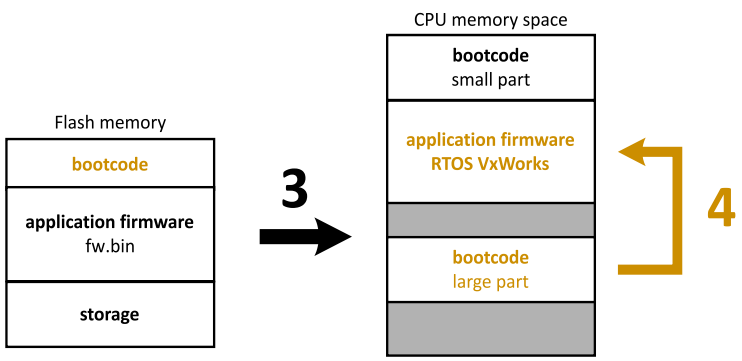

Рис. 7. Схема модификации загрузчика, часть 2

Исследование промышленного коммутатора SWITCH 2

Состав компонентов на системной плате:

1. ЦП PMC RM5231A, 32-разрядный MIPS IV, без внутренних запоминающих устройств;

2. Оперативная память SDRAM Micron MT48LC8M16A2 на 16 МБ, две ед.;

3. Флэш-память от Intel неизвестной модели;

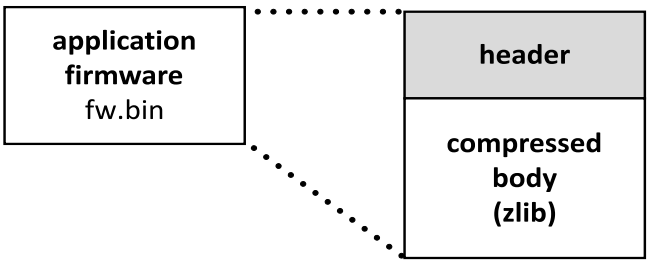

Рис. 8. Структура образа прошивки SWITCH 2

4. Чипсет Galileo GT-64115.

Образ прошивки для этого коммутатора мы скачали с официального сайта производителя и таким же способом разобрали ее структуру (рис. 8). У нее есть заголовок и сжатое по алгоритму zlib (RFC 1951) тело.

Основные поля в структуре заголовка:

1) сигнатура;

2) контрольная сумма заголовка (ADLER32);

3) контрольная сумма распакованного тела (ADLER32);

4) размер распакованного тела;

5) контрольная сумма упакованного тела (ADLER32);

6) размер упакованного тела.

В этом коммутаторе тоже отсутствует контроль подлинности прошивки (проверено экспериментально). В качестве ОС РВ используется VxWorks, правда немного более новой версии 6.1, но имеющая те же проблемы: отсутствие механизмов защиты от эксплуатации бинарных уязвимостей и набор зарегистрированных уязвимостей.

Обновлять прошивку можно теми же способами:

1) через COM-порт аутентификации также не требуется;

2) через Web-интерфейс. Для доступа к нему не требуется ввод логина и пароля, но при внесении изменений в настройки (или при обновлении прошивки) необходимо знать пароль.

К слову о пароле. Согласно документации, существует инженерный пароль, на случай утери текущего пароля. Для этого необходимо обратиться в техподдержку, предоставив MAC-адрес и серийный номер коммутатора. Это говорит о том, что существует алгоритм преобразования этих чисел в инженерный пароль. Тоже есть загрузчик, который может быть перезаписан. И это экспериментально доказано. Штатных возможностей для обновления загрузчика в SWITCH 2 нет вообще.

Заключение

Таким образом, в результате проведенного исследования выявлено, что модель безопасности современных промышленных сетевых коммутаторов явно сильно устарела. Сформулируем следующие недостатки.

1. Возможность нелегитимно обновлять прошивку (для некоторых штатных процедур аутентификация не требуется).

2. Возможность подделывать образ прошивки устройства (загрузчика, прошивку CPLD и т.д.), так как отсутствует контроль подлинности кода.

3. Отсутствуют механизмы от эксплуатации бинарных уязвимостей, что открывает дополнительные возможности для удаленного исполнения кода на коммутаторе.

Список литературы

1. *Аристова Н.И.* Ethernet в промышленной автоматизации: преодоление преград // Автоматизация в промышленности. 2013. №1.
2. *Евдокимов Д.С.* Разработка эксплойтов для АСУТП: двойная игра // Автоматизация в промышленности. 2015. №2.

Ермолов Александр Викторович *— инженер-исследователь компании Digital Security.*
Контактный телефон (911) 218-27-40.