Тестирование соответствия реализаций протокола EAP и его методов спецификациям Интернета

[1] *А.В. Никешин <alexn@ispras.ru>*
[1,2] *В.З. Шнитман <vzs@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*

Аннотация. В данной статье представлены результаты проекта по созданию тестового набора для тестирования соответствия реализаций протокола EAP и его методов спецификациям Интернета. В основе проекта лежит использование технологии UniTESK, позволяющей автоматизировать процесс верификации сетевых протоколов на основе их формальных моделей и расширения JavaTesK, реализующего эту технологию на языке программирования. Совместное использование методов мутационного тестирования позволяет протестировать устойчивость реализации протокола к искаженным сообщениям. Данный подход доказал свою эффективность, позволив обнаружить несколько критических уязвимостей и другие отклонения от спецификации в выбранных реализациях протокола.

Ключевые слова: тестирование, верификация, формальные методы, формальные спецификации, тестирование с использованием моделей, безопасность, аутентификация, контроль доступа, EAP, методы EAP, UniTESK, мутационное тестирование

DOI: 10.15514/ISPRAS-2018-30(6)-5

Для цитирования: Никешин А.В., Шнитман В.З. Тестирование соответствия реализаций протокола EAP и его методов спецификациям Интернета. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 89-104. DOI: 10.15514/ISPRAS-2018-30(6)-5

1. Введение

Аутентификация в современных сетях является одним из основных механизмов безопасности. В корпоративных решениях довольно часто используется аутентификация по протоколу EAP на базе сервера RADIUS [1, 2]. Поскольку в данном процессе участвуют разнородные объекты, возникает задача обеспечения совместимости реализаций, одним из основных методов решения которой является тестирование на соответствие

стандарту. Технология UniTESK предоставляет средства автоматизации тестирования соответствия формальным спецификациям, и одна из областей, где эта технология используется, – тестирование протоколов [3]. Особенность тестирования протоколов состоит в том, что требуется также проводить тестирование на нестандартных или искаженных входных данных, что довольно актуально для систем безопасности. Такие ситуации постоянно возникают из-за ошибок пользователей или из-за целенаправленных действий злоумышленника, но при этом часто недостаточно полно определены в спецификации протокола. В наших работах мы совмещаем методы верификации на основе формальных спецификаций и методы мутации данных, что позволяет значительно улучшить качество тестирования реализаций протоколов.

Основные особенности протокола EAP и его методов достаточно подробно изложены в нашей статье «Обзор расширяемого протокола аутентификации и его методов» [4], поэтому здесь мы на них останавливаться не будем.

2. Формальная спецификация

Формальная спецификация протокола EAP состоит из следующих компонентов:

- модели протокола, которая содержит набор структур данных, моделирующих внутреннее состояние и управляющие данные протокола и его методов в соответствии со стандартами;
- спецификационных стимулов, которые формализуют требования к изменению состояния реализации EAP при внешнем воздействии на систему;
- спецификационных реакций, которые формализуют требования к реакциям реализации EAP на внешние воздействия.

Поскольку протокол EAP использует пошаговую схему обработки сообщений (новый запрос отправляется только после получения ответа на предыдущий), в спецификации каждое отправленное тестируемой системе сообщение рассматривается как отдельный стимул, соответствующий обработке исходящего модельного сообщения EAP.

Каждое сообщение от тестируемой системы (или его отсутствие) рассматривается как отдельная реакция на соответствующий стимул.

Параметрами спецификационных методов служат сообщения EAP и его методов в модельном представлении. Наборы полей соответствующих классов зависят от структур данных, описанных в стандартах используемых методов EAP.

3. Модельное состояние

Модельное состояние протокола состоит из структур данных, относящихся к текущей сессии EAP, и структур данных, относящихся к используемому методу.

Параметры сессии включают следующие блоки данных:

Current state	текущее состояние сессии
Current method	используемый метод EAP
Peer Identity	идентификатор партнера
Result	результат аутентификации
Keys	ключи безопасности, передаваемые другим приложениям
Request Messages	множество запросов сессии
Response Messages	множество ответов сессии

Модельное состояние метода EAP включает следующие данные:

Current state	текущее состояние метода
Identity	идентификатор партнера, согласованный текущим методом (может отличаться от указанного выше идентификатора Peer Identity, а также может заменяться другими структурами данных, если это требуется в конкретном методе EAP)
Peer parameters	параметры безопасности партнера, необходимые для данного метода аутентификации (такие как пароль, сертификат и др.)
Result	результат аутентификации данного метода
Transient Keys	временные ключи криптографических алгоритмов для защиты текущего обмена

Кроме этого, у каждого метода могут быть дополнительные параметры, сильно зависящие от его спецификации. Так, например, для метода EAP-SIM определены следующие параметры [5]:

version list	список версий метода
selected version	согласованная версия метода
nonceMT	случайное число
nonceS	случайное число
peerIDs	множество идентификаторов партнера (EAP-SIM позволяет использовать разные идентификаторы в зависимости от режима аутентификации)
security parameters	параметры безопасности для передачи зашифрованных данных, если такие используются
result indication	дополнительная защита результата аутентификации
counter	счетчик для защиты целостности сообщений

triplet list	список триплетов GSM – параметры, из которых создаются ключи безопасности

4. Модель сообщений EAP

Структура сообщений в основном определяется используемым методом EAP. Спецификация протокола EAP определяет четыре типа сообщений, задающих общий шаблон процесса аутентификации [1]:

Request (запрос);

Response (ответ);

Success (успех);

Failure (неудача).

Также можно выделить несколько вспомогательных подтипов в рамках сообщений Request/Response.

Identity Используется для запроса идентификатора клиента. Спецификация требует использовать данный тип запроса исключительно для целей маршрутизации и выбора подходящего метода EAP. Как правило, это первое сообщение от аутентификатора клиенту.

Notification Необязательный тип сообщений. Используется для передачи аутентификатором сообщений, выводимых на экран клиента. Может передаваться в любое время в процессе обмена EAP.

Nak Используется в ответных сообщениях партнера, когда предложенный аутентификатором метод EAP не поддерживается. Партнер может указать в ответном сообщении желаемые методы EAP.

Каждый метод EAP может определять дополнительные сообщения, свойственные исключительно этому методу. Так, например, метод EAP-SIM задает пять новых сообщений:

Start,

Challenge,

Notification,

Client-Error,

Re-authentication

и два десятка дополнительных атрибутов, инкапсулирующих необходимые данные.

Разработанная библиотека модельного представления сообщений EAP и некоторых его методов позволяет создавать различные варианты как сообщений, так и их последовательностей.

5. Тестирование реализации протокола

Инициализация тестового сценария зависит от используемого протокола нижележащего уровня. Мы используем Radius и 802.1x [2, 6]. Процесс обмена,

как правило, начинает аутентификатор сообщением EAP-Request/Identity. В случае взаимодействия тестового сценария напрямую с сервером (без использования аутентификатора) начальное сообщение предполагается полученным.

Стимулами являются сообщения от инструментального узла в качестве ответов реализации сервера. В тестовом сценарии создаются ответы партнера в модельном представлении, передаваемые затем функции отправки сообщений. В предусловии спецификационных функций стимулов проверяется правильность структуры тестового сообщения и его своевременность, и на основании этого делается вывод о том, должен ли на него быть ответ, сообщение об ошибке или реализация должна его проигнорировать. Из модельного представления тестового сообщения строится реализационное, которое отправляется в сеть. Сборщик реакций в течение заданного времени собирает ответные сообщения целевой системы. Из реализационных сообщений строятся их модельные представления. В постусловии реакций данные проверяются на соответствие требованиям спецификации: допустимость сообщения и его структура. После проверки всех требований, результат передается тестовому сценарию, где в зависимости от плана сценария, принимается решение о продолжении или завершении информационного обмена. В случае выявления нарушения требований принимается решение о критичности ошибки и возможности отправки следующих сообщений.

Модель применяемого метода EAP определяет множество состояний тестируемой системы, на его основе строится конечный автомат, переходы которого сопоставлены с соответствующими тестовыми воздействиями. При выполнении перехода определенное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и проверяется корректность наблюдаемого поведения системы. Обход автомата по всем достижимым состояниям модели протокола осуществляется инструментарием UniTESK. Методы мутации позволяют вносить искажения в сообщения на любом этапе обмена. При этом совместное использование корректных и измененных сообщений позволяет тестовому сценарию "преодолеть" необходимые проверки в реализации, пройти через все значимые состояния протокола и в каждом состоянии выполнить необходимый набор тестовых воздействий.

6. Инструментальные средства

Разработка тестового набора велась с использованием инструмента автоматизированного тестирования JavaTesK [7], который реализует технологию UniTESK на базе объектно-ориентированного языка с автоматической сборкой мусора и обеспечивает значительное упрощение модели протокола и тестовых сценариев.

7. Тестовый стенд

Основная схема применения протокола EAP, представленная в спецификации, предполагает наличие трех узлов:

- **Клиент (партнер):** Сетевой узел, которому требуется пройти аутентификацию.
- **Аутентификатор:** Сетевой узел, управляющий доступом к ресурсам, с которым соединяется клиент.
- **Сервер EAP:** Внутренний сервер, реализующий методы EAP и выполняющий аутентификацию партнера. Данный объект определяет, прошла ли аутентификация успешно или нет.

Клиент запрашивает доступ к некоторому ресурсу, подключаясь к аутентификатору. Аутентификатор передает запрос с данными клиента серверу EAP. Сервер EAP запрашивает дополнительные данные у клиента. Обмен сообщениями между клиентом и сервером EAP продолжается до тех пор, пока выбранный метод аутентификации не завершится успешно или с ошибкой. Сервер EAP осуществляет обмен данными с аутентификатором и информирует его о результатах. На основании этого аутентификатор, принимает решение о предоставлении клиенту доступа к ресурсу.

На клиентском узле исполняется основной поток управления тестовой системы под управлением UniTESK, обход тестового автомата и верификация наблюдаемых реакций. На сервере EAP функционирует тестируемая реализация протокола. Тестовые сообщения протокола, сформированные модельной реализацией, передаются тестируемой системе, после чего регистрируются реакции тестируемого узла.

Протокол EAP может одновременно использоваться в разных средах передачи данных. Как правило, аутентификатор и сервер EAP общаются через проводной канал по протоколу AAA RADIUS. Клиент и аутентификатор используют другой стек сетевых протоколов и, возможно, другую среду передачи данных.

Следует понимать, что аутентификатор применяется для управления доступом, для расширения топологии сети и из-за возможности использовать проводные и беспроводные каналы. С точки зрения самого процесса аутентификации он используется как ретранслятор, передавая пакеты EAP между клиентом и сервером. Фактически аутентификация выполняется между двумя последними. Поэтому для тестирования непосредственно сервера EAP наличие или отсутствие аутентификатора не принципиально. На разных этапах нашего проекта при тестировании реализаций протокола EAP мы использовали разные механизмы взаимодействия с сервером EAP: как непосредственно по протоколу RADIUS, так и через аутентификатор (преимущественно через коммутатор Dell Networking N2048 по протоколу 802.1x - EAP over LAN [6]).

8. Тестируемые реализации протокола EAP

Используемые в нашем проекте реализации протокола отбирались исключительно по их доступности: бесплатная версия FreeRADIUS, условно-бесплатные Clearbox и TekRADIUS, и имеющаяся в нашем распоряжении версия Windows Server 2012 [8-11]. Также отметим, что в процессе поиска нам попадались реализации, которые давно не обновлялись или совсем заброшены их авторами, такие реализации для нас интереса не представляли.

FreeRADIUS. По сути, единственная многофункциональная бесплатная реализация, пережившая многих конкурентов. Данная реализация позиционируется разработчиком как самый распространенный сервер RADIUS, к достоинствам которого относятся свободное распространение, открытый исходный код, развитая функциональность, поддержка многочисленных методов аутентификации EAP. Сервер установлен под ОС CentOS 7.

Windows Server 2012. Доступная для нашего проекта полнофункциональная коммерческая реализация.

Clearbox Enterprise Server 6.5.2. Условно-бесплатная версия, предлагающая в начальный период эксплуатации полную функциональность. Сервер установлен под ОС Windows 10.

TekRADIUS 5.4. Условно-бесплатная версия. Возможности бесплатной версии сильно ограничены. Сервер установлен под ОС Windows 10. Отметим, что данная реализация, хотя и использует SQL-сервер для хранения базы данных пользователей, тем не менее, позволяет использовать некоторую функциональность без добавления каких-либо пользователей, используя настройки по умолчанию. Именно такая конфигурация использовалась в наших тестах.

9. Тестируемые методы EAP

Несмотря на большое разнообразие существующих методов EAP, предлагаемый к использованию указанными выше реализациями набор методов гораздо скромнее (возможно платные программные продукты корректируют набор методов EAP в зависимости от установленного оборудования). Мы условно раздели их на группы, на которые и опирались в своей работе.

Базовая функциональность и базовые методы протокола EAP. Это функциональность и требования, представленные в основной спецификации протокола и обязательные для всех реализаций [1]. Сюда же относятся простейшие методы (такие как NAK, EAP-MD5 и др.), среди которых EAP-MD5 поддерживается всеми реализациями, хотя и не рекомендуется к самостоятельному использованию как небезопасный.

Методы, использующие данные смарт-карт. В основном предлагается метод EAP-SIM [5], иногда дополнительно метод EAP-AKA/AKA’ [12, 13].

Метод аутентификации EAP-SIM использует параметры и алгоритмы SIM-карты для аутентификации и создания криптографических ключей. Данный метод основан на механизмах аутентификации GSM (GSM – стандарт мобильных сетей второго поколения) и разрабатывался для аутентификации специализированных устройств, использующих SIM-карты [14]. В Windows Server заявлена поддержка данных методов, но вероятно, требуется наличие физического устройства. Реализация Clearbox данные методы не поддерживает. Реализация TekRADIUS предлагает метод EAP-SIM в платной версии и, вероятно, также требует наличия физического устройства. В наших экспериментах такие устройства не применяются, а ручная настройка необходимых параметров поддерживается только реализацией FreeRADIUS для метода EAP-SIM, которым мы и ограничились.

Туннельные методы EAP. Переход на данные механизмы аутентификации является общей тенденцией и поэтому они представлены во всех современных реализациях EAP. Появившиеся вначале несколько туннельных методов: PEAP (Protected EAP), EAP-TTLS (EAP tunneled TLS), EAP-FAST (EAP Flexible Authentication via Secure Tunneling), не имеют статуса «Стандарта Интернет» [15-17]. Поэтому разработчики выбирают поддержку какого-либо метода исходя из своих предпочтений. Созданный позднее метод TEAP [18] на данный момент не поддерживается реализациями. Все указанные методы EAP для создания туннеля используют протокол TLS [19]. Основным туннельным методом FreeRADIUS является TTLSv0. Windows Server использует исключительно PEAP собственной разработки – PEAPv0. Clearbox предлагает метод PEAP. TekRADIUS в бесплатной версии предлагает метод PEAP. Также отметим, что метод Microsoft PEAP не имеет единого стандарта, в него довольно часто по мере необходимости вносятся изменения. Поэтому в нашей работе тестировалась только общая функциональность PEAP, свойственная всем туннельным методам на базе протокола TLS.

10. Результаты тестирования

В данном разделе представлены результаты прогона разработанного нами тестового набора, позволившего обнаружить целый ряд уязвимостей и отклонений от спецификации в выбранных реализациях протокола.

10.1 Тестирование базовой функциональности и базовых методов протокола EAP

FreeRADIUS

- Если на запрос аутентификации от сервера отправить сообщение с типом NAK и пустым полем данных (данное поле должно содержать не менее одного байта), то сервер отвечает сообщением об ошибке (хотя, неправильно сформированные сообщения должны просто отбрасываться).

- Если на запрос аутентификации от сервера отправить сообщение с типом MD5-Challenge и пустым полем данных (данное поле должно содержать не менее одного байта), то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с кодом Request (ответные сообщения всегда должны иметь код Response) и типом NAK, то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с кодом Request (ответные сообщения всегда должны иметь код Response) и типом MD5-Challenge, то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с типом Identity, то сервер отвечает повторным сообщением с запросом аутентификации (сообщение нарушает порядок обмена и должно быть отброшено).

Windows Server 2012

- Реализация полностью игнорирует содержание поля Identity (идентификатор клиента) в первом сообщении клиента.

Clearbox enterprise server

- При получении от клиента первого сообщения EAP типа REQUEST / RESPONSE с любым значением метода EAP и любыми дополнительными данными реализация начинает аутентификацию, используя настройки политики безопасности с методом EAP по умолчанию (первым сообщением от клиента предполагается RESPONSE/ Identity).
- При получении от клиента допустимого сообщения EAP (Request, Response, Success, Failure; тип метода EAP > 0 для Request/Response) со значением поля длины 0, 1 или 2 на любом этапе сессии реализация падает. Данная уязвимость не требует знания каких–либо данных пользователя. Поскольку данная уязвимость относится к общему заголовку EAP, умный аутентификатор может заблокировать подобные сообщения.

TekRADIUS

- Во входящих сообщениях EAP-Identity, если данные не пустые, реализация не проверяет поле длины.

10.2 Тестирование метода EAP-SIM

FreeRADIUS

- Почти всегда в ответ на неправильный первый ответ от клиента сервер отправляет повторный начальный запрос (начинается заново процесс аутентификации).
- Согласно спецификации метода EAP-SIM первое ответное сообщение клиента EAP-Response/SIM/Start должно содержать атрибуты IDENTITY, NONCE_MT, SELECTED_VERSION. Если в сообщении присутствуют атрибуты NONCE_MT и SELECTED_VERSION, на другие неуместные атрибуты сервер не обращает внимания (в таких случаях сервер должен прерывать аутентификацию и отправлять сообщение об ошибке),
    - в том числе сервер не обращает внимания на присутствие или отсутствие атрибута клиента IDENTITY.
- Поле длины атрибута IDENTITY клиента не проверяется.
- Содержание атрибута IDENTITY клиента полностью игнорируется.
- Если длина атрибута NONCE_MT больше положенного (спецификация определяет фиксированную длину данного атрибута 20 байт), сервер начинается процесс аутентификации заново.
- Спецификация определяет фиксированную длину атрибута SELECTED_VERSION 4 байта. Сервер принимает данный атрибут большей длины, обрабатывая только первые 4 байта.

10.3 Тестирование туннельных методов

FreeRADIUS, метод TTLSv0

- В заголовке TTLSv0 поле Длина присутствует во всех фрагментах (При использовании протокола TLS большие блоки данных разбиваются на фрагменты и отправляются последовательно несколькими сообщениями EAP, при этом полная длина исходного TLS-сообщения передается только с первым фрагментом и не должна присутствовать в остальных фрагментах).
- Если во входящем сообщении установлен бит M (указывает, что используется фрагментация и текущий фрагмент не последний), реализация ждет следующие фрагменты независимо от значений других полей.
- Если во входящем сообщении установлен бит M, реализация требует, чтобы в следующих входящих фрагментах в заголовке TTLSv0 присутствовало поле длины. Если поле длины отсутствует, реализация отправляет пустое сообщение и ждет следующий фрагмент (такой пустой цикл можно повторить до 50 раз, после чего реализация завершает сессию).

- Значение поля Version (версия метода EAP-TTLS) не проверяется.
- В различных случаях не проверяется согласование длины пакета EAP, длины данных TLS и длины в заголовке TTLSv0.
- Во входящем сообщении Acknowledgement Packet (пустой пакет, означающий ожидание от партнера дополнительных данных или следующего фрагмента) значение поля длины при установленном бите L игнорируется.
- Во входящем сообщении Acknowledgement Packet значение бита S не проверяется (Данный бит устанавливается только в самом первом сообщении сессии для начала обмена TLS).
- Также обнаружена критическая уязвимость реализации. После установления TLS-туннеля при получении внутри туннеля данных определенного формата реализация выдает ошибку "Segmentation fault" и "падает". Данная уязвимость проявляется до аутентификации клиента (т.е. до использования туннелируемых методов аутентификации) и позволяет любому клиенту (злоумышленнику достаточно знать имя клиента, для которого применяется метод EAP-TTLS) провести DoS атаку ("отказ в обслуживании"). Уязвимость наблюдается для FreeRADIUS 3.0.13 под ОС CentOS 7 (release 7.4.1708).

Windows Server 2012, метод PEAP

- Реализация игнорирует бит S ("Start TLS") во входящем сообщении от клиента (данный бит используется для инициации сервером TLS обмена и применяется только в первом пустом сообщении от сервера клиенту).
- В некоторых случаях игнорирует значение поля длины в заголовке PEAP (данное поле указывает полную длину TLS сообщения до фрагментации).
- В некоторых случаях не проверяется согласование длины пакета EAP, длины данных TLS и длины в заголовке PEAP.
- Реализация отбрасывает входящие сообщения с битом M (указывает, что используется фрагментация и текущий фрагмент не последний), если считает, что текущее сообщение не должно быть фрагментировано (например, сообщение EAP-TTLS-TLS_ClientHello).
- Если реализация ждет следующий фрагмент от клиента, а получает пустое сообщение, то отправляет в ответ также пустое сообщение (EAP-TTLS Acknowledgement packet). Такой цикл можно повторять очень долго, в реализации вероятно отсутствует ограничение количества таких сообщений (например, FreeRADIUS прерывает аутентификацию после 50 таких циклов).

Clearbox Enterprise Server, метод PEAP

- В некоторых случаях игнорирует значение поля длины в заголовке PEAP (данное поле указывает полную длину TLS-сообщения до фрагментации).

- Значение поля Version (версия метода PEAP) не проверяется.
- Реализация требует, чтобы в сообщении Acknowledgement Packet (пустой пакет) в поле флаг все биты были 0, включая зарезервированные (Reserved bits). Согласно спецификации, зарезервированные биты должны устанавливаться в 0 при отправке и игнорироваться при получении.
- При получении пустого сообщения EAP-Response/type=EAP-TLS (без каких-либо данных) реализация падает. Данная ошибка наблюдается в туннельных методах при отправлении такого сообщения после создания TLS-туннеля (в качестве внутреннего метода EAP). Также такое поведение часто наблюдается в ответ на сообщение EAP-Request/TLS-Start (т.е. до согласования туннеля), но почему-то не всегда, иногда требуется несколько раз повторить тест.
- После установления защищенного туннеля (после сообщения TLS-Finished) для продолжения обмена партнер должен отправить серверу специальный пустой пакет Acknowledgement Packet. Если в этом сообщении в поле флаг установлен бит L, то реализация отвечает пакетом RADIUS/Access-Accept без атрибута EAP-Message.

TekRADIUS, метод PEAP

- Во время создания TLS-туннеля при получении сообщения EAP/TLS-ClientHello:
  - если установлен бит L (поле флаг метода PEAP, указывает на присутствие поля длины TLS сообщения), реализация игнорирует значения полей флага (в.ч. биты S и Version) и длины,
  - если установлен бит L, реализация игнорирует добавление лишних байт к сообщению,
  - если бит L не установлен, а другие биты поля флаг не 0 (включая Reserved bits), реализация игнорирует сообщение.
- Во время создания TLS-туннеля при получении сообщения EAP/TLS-Finished:
  - если установлен бит L, реализация игнорирует значения полей флага (в.ч. биты S и Version) и длины,
  - если бит L не установлен, а другие биты поля флаг не 0 (включая Reserved bits), реализация сообщает об ошибке (TLS-Alert),
  - если установлен бит M (указывает наличие следующих фрагментов) и сообщение не пустое (присутствуют TLS данные), реализация отправляет EAP пакет из одних нулей, включая заголовок EAP (т.е. значение Radius атрибута EAP-Message заполнено нулями, при этом остальные поля сообщения Radius правильные).
- После создания TLS-туннеля при инициализации внутреннего метода и получении неправильного сообщения реализация обычно отвечает

сообщением EAP из единственного нулевого байта (внутри туннеля).

- Также обнаружено интересное поведение реализации. После создания TLS-туннеля и получения сообщения Acknowledgement Packet, начинается согласование внутреннего метода EAP под защитой созданного туннеля. Реализация отправляет сообщение EAP-Identity. Если от партнера в ответ приходит EAP-Response/type=MD5 с произвольным значением контрольной MD5-суммы, то реализация отвечает сообщением RADIUS-Access-Challenge/EAP-Success. Такое поведение представляется нам очень странным и, несомненно, нарушает безопасность системы аутентификации.

11. Заключение

В данной статье представлены результаты работы по созданию тестового набора для тестирования соответствия реализаций протокола EAP и его методов спецификациям Интернет. Наша методика тестирования основана на использовании комбинации двух подходов к верификации реализаций сетевых протоколов. Первый подход базируется на создании формальных моделей протоколов и опирается на технологию UniTESK, обеспечивающую автоматизацию процесса верификации. В нашем проекте мы использовали расширение JavaTesK, реализующего эту технологию на языке программирования Java. Второй подход связан с использованием методов мутационного тестирования, которые позволяют протестировать устойчивость реализации протокола к искаженным сообщениям и значительно улучшить качество тестирования реализаций. Такая методика доказала свою эффективность, позволив обнаружить несколько критических уязвимостей и другие отклонения от спецификаций в выбранных реализациях протокола EAP и его методов.

Благодарности

Работа выполнялась при поддержке РФФИ (проект № 16-07-00603 «Верификация функций безопасности и оценка устойчивости к атакам реализаций протокола аутентификации EAP»).

Список литературы

[1]. IETF RFC 3748. B. Aboba, et al. Extensible Authentication Protocol (EAP). June 2004. Доступно по ссылке: https://tools.ietf.org/html/rfc3748, дата обращения 01.12.2018.

[2]. IETF RFC 3579. B. Aboba and P. Calhoun. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). September 2003. Доступно по ссылке: https://tools.ietf.org/html/rfc3579, дата обращения 01.12.2018.

[3]. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture. Proceedings of FME 2002. LNCS 2391, 2002, pp. 77–88

[4]. Никешин А.В., Шнитман В.З. Обзор расширяемого протокола аутентификации и его методов. Труды ИСП РАН, том 30, вып. 2, 2018 г., стр. 113-148. DOI: 10.15514/ISPRAS-2018-30(2)-7

[5]. IETF RFC 4186. Haverinen & Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). January 2006. Доступно по ссылке: https://tools.ietf.org/html/rfc4186, дата обращения 01.12.2018.

[6]. IEEE Standard 802.1X-2010 - IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control, 2010.

[7]. JavaTESK. Доступно по ссылке: http://www.unitesk.ru/content/category/5/25/60/, дата обращения 01.12.2018.

[8]. FreeRADIUS. Доступно по ссылке: http://freeradius.org, дата обращения 01.12.2018.

[9]. Clearbox Enterprise Server. Доступно по ссылке: http://xperiencetech.com/, дата обращения 01.12.2018.

[10]. TekRADIUS. Доступно по ссылке: https://www.kaplansoft.com/tekradius/, дата обращения 01.12.2018.

[11]. Windows Server 2012 R2. Доступно по ссылке: https://www.microsoft.com, дата обращения 01.12.2018.

[12]. IETF RFC 4187. Arkko & Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). January 2006. Доступно по ссылке: https://tools.ietf.org/html/rfc4187, дата обращения 01.12.2018.

[13]. IETF RFC 5448. Arkko, et al. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). May 2009. Доступно по ссылке: https://tools.ietf.org/html/rfc5448, дата обращения 01.12.2018.

[14]. European Telecommunications Standards Institute. GSM Technical Specification GSM 03.20 (ETS 300 534): Digital cellular telecommunication system (Phase 2); Security related network functions, August 1997.

[15]. Microsoft Corporation. [MS-PEAP]: Protected Extensible Authentication Protocol (PEAP). December 2017. Доступно по ссылке: https://msdn.microsoft.com/en-us/library/cc238354.aspx, 06.12.2018, дата обращения 01.12.2018.

[16]. IETF RFC 5281. Funk & Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). August 2008. Доступно по ссылке: https://tools.ietf.org/html/rfc5281, дата обращения 01.12.2018.

[17]. IETF RFC 4851. Cam-Winget, et al. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). May 2007. Доступно по ссылке: https://tools.ietf.org/html/rfc4851, дата обращения 01.12.2018.

[18]. IETF RFC 7170. Zhou, et al. Tunnel Extensible Authentication Protocol (TEAP) Version 1. May 2014. Доступно по ссылке: https://tools.ietf.org/html/rfc7170, дата обращения 01.12.2018.

[19]. IETF RFC 5246. Dierks, T. and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. Доступно по ссылке: https://tools.ietf.org/html/rfc5246, дата обращения 01.12.2018.

Conformance testing of Extensible Authentication Protocol implementations

[1] *A.V. Nikeshin <alexn@ispras.ru>*
[1,2] *V.Z. Shnitman <vzs@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*

Abstract. The paper presents a model-based approach to conformance testing of Extensible Authentication Protocol (EAP) implementations. Conformance testing is the basic tool to ensure interoperability between implementations of a protocol. Using UniTESK technology allows automating the verification of network protocols based on their formal models. Additional applying of mutation testing allows evaluating the robustness of the implementations to receive incorrect packets. We applied the test suite to several implementations of EAP and present brief results. This approach has proved to be effective in finding several critical vulnerabilities and other specification deviations in the EAP implementations.

Keywords: testing, verification, formal methods, formal specifications, model-based testing, security, authentication, access control, EAP, EAP methods, UniTESK, mutation testing.

DOI: 10.15514/ISPRAS-2018-30(6)-5

For citation: Nikeshin A.V., Shnitman V.Z. Conformance testing of Extensible Authentication Protocol implementations. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 89-104 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-5

References

[1]. IETF RFC 3748. B. Aboba, et al. Extensible Authentication Protocol (EAP). June 2004. Available at: https://tools.ietf.org/html/rfc3748, accessed 01.12.2018.
[2]. IETF RFC 3579. B. Aboba and P. Calhoun. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). September 2003. Available at: https://tools.ietf.org/html/rfc3579, accessed 01.12.2018.
[3]. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture. Proceedings of FME 2002. LNCS 2391, 2002, pp. 77–88.
[4]. Nikeshin A.V., Shnitman V.Z. The review of Extensible Authentication Protocol and its methods. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue. 2, 2018, pp. 113-148 (in Russian). DOI: 10.15514/ISPRAS-2018-30(2)-7.
[5]. IETF RFC 4186. Haverinen & Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). January 2006. Available at: https://tools.ietf.org/html/rfc4186, accessed 01.12.2018.
[6]. IEEE Standard 802.1X-2010 - IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control, 2010.

[7]. JavaTESK. Available at: http://www.unitesk.ru/content/category/5/25/60/, accessed 01.12.2018.
[8]. FreeRADIUS. Available at: http://freeradius.org, accessed 01.12.2018.
[9]. Clearbox Enterprise Server. Available at: http://xperiencetech.com/, accessed 01.12.2018.
[10]. TekRADIUS. Available at: https://www.kaplansoft.com/tekradius/, accessed 01.12.2018.
[11]. Windows Server 2012 R2. Available at: https://www.microsoft.com/, accessed 01.12.2018.
[12]. IETF RFC 4187. Arkko & Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). January 2006. Available at: https://tools.ietf.org/html/rfc4187/, accessed 01.12.2018.
[13]. IETF RFC 5448. Arkko, et al. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). May 2009. Available at: https://tools.ietf.org/html/rfc5448/, accessed 01.12.2018.
[14]. European Telecommunications Standards Institute, GSM Technical Specification GSM 03.20 (ETS 300 534): Digital cellular telecommunication system (Phase 2); Security related network functions, August 1997.
[15]. Microsoft Corporation. [MS-PEAP]: Protected Extensible Authentication Protocol (PEAP). December 2017. Available at: https://msdn.microsoft.com/en-us/library/cc238354.aspx, 06.12.2018/, accessed 01.12.2018.
[16]. IETF RFC 5281. Funk & Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). August 2008. Available at: https://tools.ietf.org/html/rfc5281/, accessed 01.12.2018.
[17]. IETF RFC 4851. Cam-Winget, et al. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). May 2007. Available at: https://tools.ietf.org/html/rfc4851/, accessed 01.12.2018.
[18]. IETF RFC 7170. Zhou, et al. Tunnel Extensible Authentication Protocol (TEAP) Version 1. May 2014. Available at: https://tools.ietf.org/html/rfc7170.
[19]. IETF RFC 5246. Dierks, T. and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. Available at: https://tools.ietf.org/html/rfc5246/, accessed 01.12.2018.