А.В. Никешин, Н.В. Пакулин,
В.З. Шнитман

**Подходы к разработке тестового
набора для тестирования реализаций
протокола EAP и его методов**

Рекомендуемая форма библиографической ссылки

Никешин А.В., Пакулин Н.В., Шнитман В.З. Подходы к разработке тестового набора для тестирования реализаций протокола EAP и его методов // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19-24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2016. — С. 290-297. — doi:[10.20948/abrau-2016-24](https://doi.org/10.20948/abrau-2016-24)

Размещена также [презентация к докладу](#)

Подходы к разработке тестового набора для тестирования реализаций протокола EAP и его методов

А.В. Никешин, Н.В. Пакулин, В.З. Шнитман

Институт системного программирования Российской академии наук

Аннотация. В данной работе представлен подход к разработке тестового набора для тестирования реализаций протокола EAP. Этот подход использует комбинацию двух направлений тестирования сетевых протоколов. Технология UniTESK позволяет автоматизировать процесс построения тестовых последовательностей с помощью обхода некоторого автомата теста, при этом для построения тестовых воздействий и вынесения вердикта о корректности наблюдаемого поведения реализации используется модель протокола. Использование дополнительно методов мутационного тестирования позволяет вносить изменения в поток данных на любом этапе обмена протокола. Такая комбинация тестовых методов позволяет, с одной стороны, проверять требования спецификации протокола, а с другой - протестировать устойчивость реализации протокола к искаженным пакетам во всех значимых состояниях.

Ключевые слова: безопасность, аутентификация, EAP, методы EAP, протоколы, тестирование, оценка устойчивости, Интернет, стандарты, формальные методы спецификации

1. Введение

EAP является одним из широко используемых в современных сетях протоколов безопасности. В отличие от другого распространенного протокола - TLS, предназначенного для создания защищенного канала передачи данных, EAP в основном используется для контроля доступа к разделяемым ресурсам. В частности EAP нашел широкое применение в маршрутизаторах и беспроводных сетях. Возможные уязвимости реализаций протокола могут создать серьезные проблемы безопасности. Сообщения о найденных уязвимостях периодически появляются в Интернет. Кроме того, производители оборудования нередко используют собственные реализации протокола, а значит возникает задача обеспечения совместимости реализаций, одним из основных методов решения которой является тестирование на соответствие стандарту.

Протокол EAP относится к протоколам аутентификации, поддерживающим расширения своей функциональности за счет добавления новых методов [1]. Изначально EAP был разработан для аутентификации в системах сетевого доступа, где не поддерживается протокол IP. EAP обычно

работает непосредственно на базе протоколов канального уровня типа PPP [2] или IEEE 802 [3], не требуя использования протокола IP [4]. EAP может применяться на выделенных каналах и в коммутируемых устройствах, как в проводных, так и в беспроводных сетях. Данный протокол использует пошаговую (lock-step) схему обработки сообщений: новый запрос отправляется только после получения ответа на предыдущий. Поэтому данный протокол не предназначен для передачи больших объемов данных, в отличие от транспортных протоколов типа TCP [5] или SCTP [6]. EAP обеспечивает собственную поддержку повторной передачи сообщений и избавления от дубликатов, но она основана на гарантированном порядке доставки сообщений протоколом нижнего уровня. Соответственно, обработка пакетов, доставленных с нарушением порядка, не поддерживается.

Одним из преимуществ архитектуры EAP является ее гибкость. Выбор конкретного механизма аутентификации происходит после того, как аутентифицирующая сторона (authenticator) получит дополнительную информацию от партнерского узла. Вместо необходимости каждый раз обновлять аутентификатор для поддержки нового метода аутентификации, архитектура EAP позволяет использовать выделенный сервер, который поддерживает различные методы аутентификации, а сам аутентификатор просто пересылает сообщения от других узлов такому серверу.

Архитектура EAP использует следующие роли для сетевых узлов:

- Аутентификатор (authenticator) — сетевой узел, начинающий процесс аутентификации.
- Партнер (peer) — сетевой узел, отвечающий на запросы аутентификатора.
- Внутренний сервер аутентификации (backend authentication server) — выделенный сервер, предоставляющий услуги аутентификации (выполняет методы EAP) для аутентификатора.
- Сервер EAP — объект, реализующий методы EAP. Он либо размещается на сервере аутентификации, либо является частью аутентификатора.

Протокол EAP позволяет использовать две схемы организации системы аутентификации клиентов, различающиеся тем, где реализованы методы аутентификации.

В первом случае клиент (партнер) запрашивает доступ к некоторому ресурсу, обращаясь к системе доступа (аутентификатору). Аутентификатор выбирает метод аутентификации в соответствии со своей политикой безопасности и самостоятельно проводит аутентификацию клиента. Для этого он должен поддерживать данный метод аутентификации.

Во втором случае аутентификатор не поддерживает выбранный метод и для продолжения аутентификации обращается к сторонней системе (внутреннему серверу аутентификации), пересылая ей данные от клиента. Сам процесс аутентификации фактически производится на выделенном сервере.

Аутентификатор в этом случае выполняет роль посредника между клиентом и сервером аутентификации и, на основании результата аутентификации от сервера, принимает решение о предоставлении клиенту доступа к запрошенному ресурсу.

Вторая схема аутентификации позволяет упростить управление доступом к нескольким разделяемым ресурсам, что очень важно в целом ряде случаев. Многие сетевые устройства обычно имеют довольно ограниченные аппаратные ресурсы и, например, не могут хранить в памяти информацию о большом числе пользователей. Кроме того, число пользователей может регулярно меняться, и возникает необходимость в единой базе данных. В такой системе от аутентификатора конкретного ресурса требуется только поддержка базовой функциональности протокола EAP. А реализация всех методов аутентификации и база данных с информацией обо всех пользователях и их правах доступа находятся в единой подсистеме – сервере аутентификации. При этом, однако, увеличивается объем сетевого трафика.

Система доступа позволяет также комбинировать обе схемы аутентификации, поддерживая на аутентификаторе только отдельные методы аутентификации, а для реализации других методов использовать сервер аутентификации.

Можно отметить следующие преимущества протокола:

- Протокол EAP может поддерживать множество механизмов аутентификации без предварительного согласования используемого метода.
- Устройства сетевого доступа не обязаны понимать каждый метод аутентификации и могут действовать как посредники для внутреннего сервера аутентификации. Аутентификатор, например, может поддерживать часть методов EAP или самостоятельно обеспечивать аутентификацию только локальных сетевых узлов, и, в то же время, работать в режиме посредника для внешних узлов и непонятных методов аутентификации.
- Разделение аутентификатора и внутреннего сервера аутентификации упрощает управление сетевым доступом и политиками безопасности.

В настоящее время протокол EAP широко используется в самых разных сетевых средах. В IANA (Internet Assigned Numbers Authority) зарегистрировано несколько десятков расширений данного протокола (такие как MD5-Challenge, EAP-Cisco Wireless, EAP-MSCHAP-V2, EAP-PSK и др.), реализующих широкий набор методов аутентификации и использующие различные средства аутентификации (пароли, сертификаты, токены и др.) [7]. Поэтому тестирование его реализаций является актуальной задачей.

2. Подходы к тестированию протокола EAP

В нашей работе по тестированию протокола EAP мы используем технологию UniTESK [8]. Для текущих экспериментов была разработана модель протокола EAP, которая описывает сложную структуру данных и определяет отношения между различными ее элементами, а также внутреннее состояние и правила функционирования протокола.

Технология UniTESK предоставляет средства автоматизации тестирования соответствия формальным спецификациям. В UniTESK тесты организованы в виде конечных автоматов, состояния автомата определяются состояниями тестируемой системы, а переходы сопоставлены с определенными тестовыми воздействиями. При выполнении перехода заданное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. В UniTESK алгоритм обхода реализован как внутренний компонент и не зависит от протокола и тестируемой системы.

Для тестирования протоколов, как правило, используются два сетевых узла: на одном узле установлена тестируемая реализация протокола, на другом – модельная реализация под управлением UniTESK. Тестовые сообщения протокола, сформированные модельной реализацией, передаются тестируемой системе, после чего регистрируются реакции тестируемого узла.

В нашей работе сочетаются два подхода к тестированию протокола: тестирование реализаций на соответствие требованиям спецификации и методы мутационного тестирования.

Первое направление предполагает разработку модели протокола на основе формальной спецификации и набора тестов, для проверки требований спецификации.

Формальная спецификация EAP состоит из следующих компонентов:

- модельного состояния, которое содержит набор структур данных, моделирующих концептуальные структуры данных из стандарта EAP;
- 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.

В спецификации каждое отправленное целевой системе сообщение рассматривается как последовательность стимулов. Каждое сообщение от целевой системы рассматривается как последовательность реакций.

В отличие от выше указанного подхода, методы мутационного тестирования больше предназначены для обнаружения неадекватного поведения тестируемой системы (завершение из-за фатальной ошибки,

"подвисание", ошибки доступа к памяти) в ситуациях, не определенных спецификацией.

Мутационное тестирование является развитием идеи fuzz-генератора, предложенного в 1988 году Бартоном Миллером [9, 10]. Он предложил подавать на вход приложениям неструктурированные случайные данные. Данный метод, опробованный на системных утилитах ОС Unix, показал хорошие результаты. В дальнейшем данный подход был усовершенствован: на вход стали подаваться более-менее структурированные данные с измененными отдельными частями, соответствующими конкретной тестируемой системе. При тестировании сетевых протоколов использование бессистемных входных данных дает довольно низкую эффективность. Различные механизмы контроля данных, используемые в современных протоколах (такие как контрольные суммы, одноразовые значения, взаимосвязь различных сообщений и полей и т.п.), требуют более "умного" построения входных сообщений.

В нашем подходе мутации подвергаются правильные сообщения, сформированные на основе разработанной модели протокола, что позволяет изменять данные на любом этапе обмена этого протокола. При этом совместное использование корректных и измененных сообщений позволяет тестовому сценарию "преодолеть" необходимые проверки в реализации, пройти через все значимые состояния и в каждом состоянии протестировать устойчивость реализации протокола к искаженным пакетам.

Для изменения корректных сообщений протокола используются механизмы, называемые операторами мутации, которые определяют область значений и правила изменения конкретного типа данных. Как правило, сообщения сетевых протоколов используют комбинацию двух типов данных: целых значений и байтовых массивов. Для них определены следующие виды мутации.

Для целых типов данных:

- установление произвольного значения,
- установление значения нуль,
- установление максимального для данного типа значения.

Для байтовых массивов:

- изменение произвольного байта в массиве,
- изменение всех байтов в массиве,
- использование массива нулевой длины
- уменьшение/увеличение длины массива

Встроенные механизмы UniTESK позволяют менять порядок сообщений: соблюдение порядка сообщений – одно из требований спецификации протокола EAP.

3. Устройство тестового стенда

В нашей работе на данном этапе мы используем первую схему организации доступа по протоколу EAP, состоящую из двух сетевых узлов. На одном узле выполняется основной поток управления тестовой системы, обход тестового автомата и верификация наблюдаемых реакций. На другом узле функционирует тестируемая реализация.

В качестве реализации протокола EAP выбрана реализация FreeRADIUS[11]. Данная реализация позиционируется разработчиком как самый распространенный сервер RADIUS, к достоинствам которого относятся свободное распространение, открытый исходный код, развитая функциональность, в том числе поддержка многочисленных методов аутентификации EAP. Такая хорошо отлаженная реализация полезна для тестирования на первом этапе проекта для более качественной отладки самого тестового набора. Кроме того, в силу ее широкого распространения, представляет интерес ее тестирование на соответствие стандарту протокола EAP (что необходимо, как указывалось выше, для обеспечения совместимости с другими реализациями). Стоит напомнить, что популярность программных продуктов не избавляет их от ошибок. Периодически публикуются сообщения о найденных уязвимостях FreeRADIUS. Сервер установлен под ОС Ubuntu 16.04 [12].

4. Результаты тестирования

Выполняемый нами проект начат в январе 2016 года. Для текущих экспериментов были выбраны методы аутентификации EAP-SAKE [13] и EAP-PSK [14], основанные на заранее распределенных криптографических ключах. На данный момент в рамках технологии UniTESK (с использованием инструмента JavaTesK [15])

- разработаны модели базового протокола EAP и методов аутентификации EAP-PSK и EAP-SAKE,
- разработаны спецификации и медиаторы для протокола EAP и указанных методов,
- разработан набор тестов, покрывающий часть требований спецификаций.

На данный момент каких-либо отклонений реализации от требований спецификации не выявлено. Разработка тестового набора продолжается.

5. Заключение

В данной работе представлен подход к разработке тестового набора для тестирования реализаций протокола EAP. Этот подход использует комбинацию двух направлений тестирования сетевых протоколов. Технология UniTESK позволяет автоматизировать процесс построения тестовых последовательностей с помощью обхода некоторого автомата теста, при этом для построения

тестовых воздействий и вынесения вердикта о корректности наблюдаемого поведения реализации используется модель протокола.

Использование дополнительно методов мутационного тестирования позволяет вносить изменения в поток данных на любом этапе обмена протокола. Такая комбинация тестовых методов позволяет, с одной стороны, проверять требования спецификации протокола, а с другой — протестировать устойчивость реализации протокола к искаженным пакетам во всех значимых состояниях.

Этот подход доказал свою эффективность и в наших предыдущих проектах, обеспечив обнаружение различных отклонений от спецификации и других ошибок при тестировании сетевых протоколов [16, 17].

Проект выполняется при поддержке РФФИ, проект № 16-07-00603 «Верификация функций безопасности и оценка устойчивости к атакам реализаций протокола аутентификации EAP».

Литература

1. IETF RFC 3748, B. Aboba, et al. “Extensible Authentication Protocol (EAP)”, June 2004.
2. IETF RFC 1661, W. Simpson “The Point-to-Point Protocol (PPP)”, July 1994.
3. IEEE Standard 802, Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Overview and Architecture", 1990.
4. IETF RFC 791, “Internet Protocol”, September 1981.
5. IETF RFC 793, “Transmission Control Protocol”, September 1981.
5. IETF RFC 2960, Stewart, et al. “Stream Control Transmission Protocol”, October 2000.
7. Extensible Authentication Protocol (EAP) Registry, <http://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml>
8. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture. //Proceedings of FME 2002. LNCS 2391, pp. 77-88, Springer-Verlag, 2002
9. B. P. Miller, D. Koski, C. P. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl. Fuzz revisited: A re-examination of the reliability of UNIX utilities and services // Office, vol. 1525, no. October 1995, pp. 1–23, 1995.
10. B. P. Miller, L. Fredriksen, and B. So. An empirical study of the reliability of UNIX utilities // Commun. ACM, vol. 33, pp. 32–44, December 1990.
11. FreeRADIUS, <http://freeradius.org> .
12. Ubuntu 16.04, <http://www.ubuntu.com/>.
13. IETF RFC 4763, M. Vanderveen and H. Soliman “Extensible Authentication Protocol Method for Shared-secret Authentication and Key Establishment (EAP-SAKE)”, November 2006.
14. IETF RFC 4764, F. Bersani and H. Tschofenig “The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method”, January 2007.
15. JavaTESK, <http://www.unitesk.ru/content/category/5/25/60/>.

16. А.В.Никешин, Н.В.Пакулин, В.З.Шнитман "Мутационное тестирование сетевых протоколов с использованием формальных моделей" // Научный сервис в сети Интернет: труды XVII Всероссийской научной конференции (21-26 сентября 2015 г., г. Новороссийск). - М.: ИПМ им. М.В.Келдыша, 2015. ISBN 978-5-98354-015-6. Стр. 259-266.
17. А.В.Никешин, Н.В.Пакулин, В.З. Шнитман "Тестирование реализаций клиента протокола TLS" // Труды Института системного программирования РАН. Том 27. Выпуск 2. 2015 г. Стр. 145-160. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).