Москвичев Н.П., Кульков И.В., Болдырев А.Г. Фаззинг анализ безопасности реализации защищенного протокола сетевого обмена. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2017. – С. 121-130.

УДК 004

ФАЗЗИНГ АНАЛИЗ БЕЗОПАСНОСТИ РЕАЛИЗАЦИИ ЗАЩИЩЕННОГО ПРОТОКОЛА СЕТЕВОГО ОБМЕНА

Н.П. Москвичев, И.В. Кульков, А.Г. Болдырев

FUZZING ANALYSIS OF SECURED NETWORK PROTOCOL IMPLEMENTATION

N.P. Moskvichev, I.V. Kulkov, A.G. Boldyrev

Аннотация. Приведен подход к оценке безопасности реализации защищенного протокола сетевого обмена методом фаззинг тестирования, рассматриваются основные этапы фаззинг тестирования, приводится пример использования автоматизированного инструментария на одном из вариантов защищенного протокола.

Ключевые слова: фаззинг, анализ, безопасность, протокол.

Abstract. This article provides an approach to assessing the safety of secure network protocol implementation using fuzzing testing, describes the main phases of fuzzing testing and provides an example of fuzzing testing process.

Keywords: fuzzing, analysis, security, protocol.

Рассмотрим типовой вариант сетевой инфраструктуры, в которой взаимодействие внешних клиентов с вычислительными ресурсами, расположенными в сегменте защищаемой сети, выполняется с использованием защищенных протоколов сетевого взаимодействия. Упрощенная схема представлена ниже, на рисунке 1.

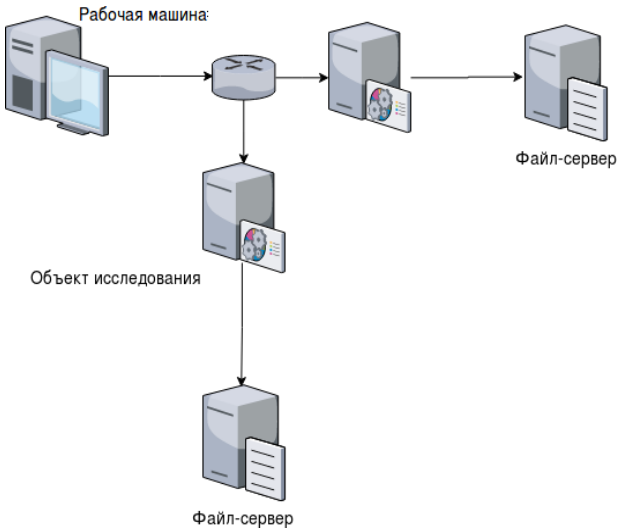

Рис. 1. Схема сети

Проведем эксперимент по оценке безопасности защищенного протокола обмена данными между двумя сетевыми устройствами методом фаззинг тестирования.

Фаззинг тестирование [1] – это техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающаяся в передаче на вход системы неправильных, неожиданных или случайных данных. В нашем случае данный подход позволяет оценить стойкость сервиса, обрабатывающего

входящие данные по защищенному протоколу, к атакам типа RCE, переполнение буфера и т.п.

Все фаззеры делятся на две категории [2]:

– мутационные, которые изменяют существующие образцы данных и создают условия для тестирования;

– порождающие, которые создают условия для тестирования с чистого листа, моделируя необходимый протокол или формат файла.

Выберем подходящий инструмент фаззинг тестирования, для чего проведем анализ наиболее распространенных фаззеров и заполним таблицу 1.

Таблица 1

Выбор инструмента

Название	**Методы фаззинга**	**Документированность**	**Доступность**	**Платформы**	**Поддержка**
SPIKE	“Умный” фаззинг	Плохая документация	Входит в состав Kali Linux	Linux	Не поддерживается
Sfuzz	“Глупый” фаззинг	man-page	Входит в состав Kali Linux	Linux	Не поддерживается
Peach	“Умный” фаззинг	Хорошо документирован [2]	Есть бесплатная версия	Windows, Linux	Поддерживается

На основе выбранных критериев оптимальным инструментом фаззинг тестирования является Peach Fuzzer. Также данный инструмент обеспечивает контроль ответной реакции путем использования агентов и мониторов, что может пригодиться при проведении дальнейших исследований, но не в данный момент, так как отсутствует доступ к сетевому устройству для их запуска на тестируемом устройстве. Шаблоны для фаззера представляют собой xml-файл. В нём описываются модель данных, модель состояний, стратегия фаззинга, агенты, мониторы и тестовое поведение.

Работу по осуществлению фаззинг тестирования сетевого протокола можно разбить на следующие этапы:

1. Сниффинг и анализ пакетов трафика обмена данными между двумя сетевыми узлами.
2. Формирование методологии фаззинг тестирования.
3. Разработка шаблона для фаззинг тестирования.
4. Проведение фаззинг тестирования и отслеживание ответной реакции.
5. Анализ полученных результатов.

Этап 1. Сниффинг и анализ пакетов трафика

На первом этапе был выполнен сниффинг сетевого трафика средствами Wireshark. На рисунке 2 представлены отфильтрованные пакеты по адресу 192.168.100.2, являющемуся адресом интерфейса исследуемого сетевого шлюза.

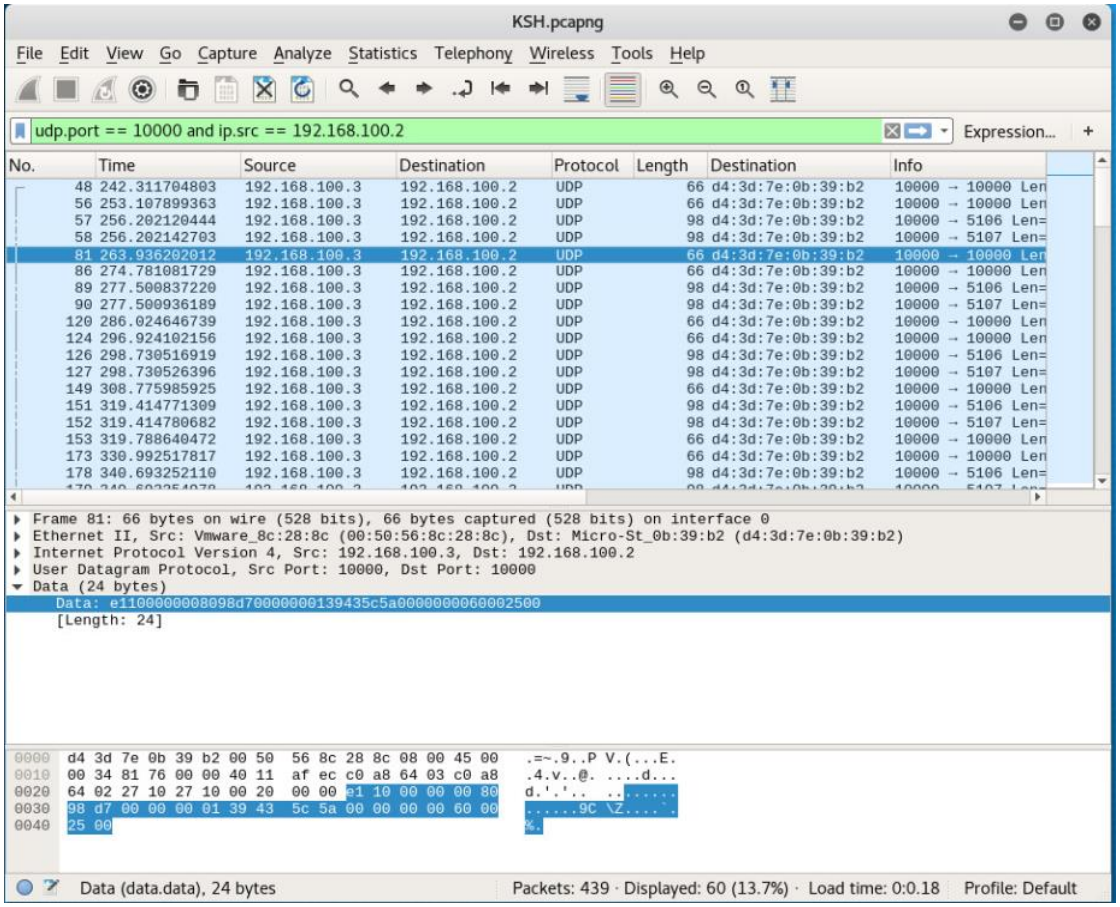

Рис. 2. Дамп трафика

Для уточнения порта и протокола сетевого уровня для защищенного протокола воспользовались доступными сведениями. Протокол – UDP и порт – 10000.

При предварительном просмотре UDP пакетов для порта 10000 были выделены 2 группы пакетов с разными длинами: группа I и группа II. Данные по полям протоколов для каждой группы представлены в таблице 2.

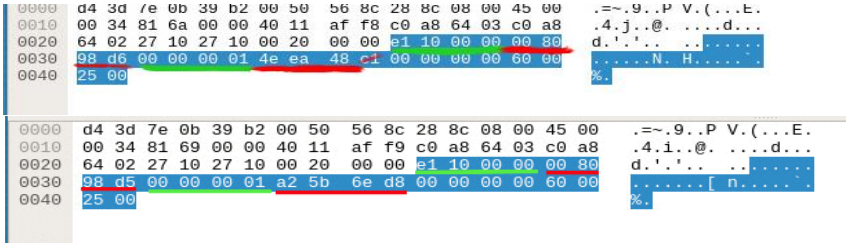

Рис. 3. Сравнение пакетов

По результатам анализа содержимого пакетов (см. рисунок 3) группы I можно предположить, что два поля остаются неизменными (выделенные зеленым цветом), а два изменяются (выделенные красным цветом). Изменяемые поля: первое поле инкрементируется на 1 и является счетчиком отправленных пакетов, второе поле – изменяется полностью от пакета к пакету и его назначение остается под вопросом. Последнее поле длиной 4 байта изменяется побитово и предназначено для выставления флагов.

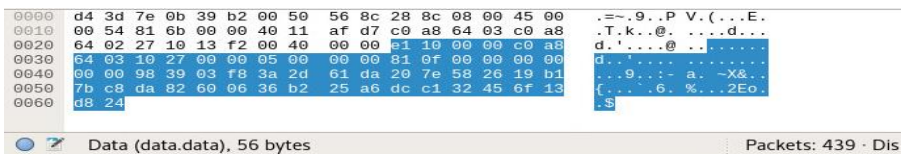

Рис. 4. Пакет II группы

Рассмотрим структуру пакетов из группы II (см. рисунок 4). В данных датаграммах также повторяются константные поля, поле счетчика пакетов и поле флагов.

Значение поля, которое было нулевым в пакетах группы 1, заменено на случайную последовательность, скорее всего из-за вставки в конце датаграммы случайной последовательности. Вероятно, таким образом добавляется преобразованное случайным образом сообщение. Хорошо известно, что для обеспечения защиты от навязывания и контроля целостности данных используются имитовставки и контрольные суммы. Возможно, поле, которое идёт после счётчика, используется для таких же целей, но только касательно значений самого счётчика. Для удобства была сформирована таблица 2.

Таблица 2

Структура пакетов

Название поля	**Значение поля**	**Длина поля**
id	Идентификатор криптошлюза	32 бита
counters	Счётчик x64	64 бита
counters_hash	Контрольное значение для счётчика	32 бита
data_hash	Контрольное значение для данных	32 бита
flags	Флаги для передаваемых данных	32 бита
datagram	Передаваемые шифрованные данные	32 байта

Этап 2. Формирование методологии фаззинг тестирования

На втором этапе составим методологию фаззинг тестирования данного сетевого протокола. Для этого необходимо рассмотреть поля, подвергаемые фаззинг тестированию, и для каждого из них определить метод тестирования. Таким образом, была составлена таблица 3, в которой указывается метод фаззинг тестирования для полей из таблицы 2.

Таблица 3

Описание полей протокола

Название поля	**Значение поля**	**Длина поля**	**Примечание**
id	Идентификатор криптошлюза	32 бита	Поле, содержащее константное значение, не имеет смысла подвергать изменениям, так как шлюз его будет попросту отбрасывать.
counters	Счётчик x64	64 бита	Данные поля должны быть подвергнуты фаззинг тестированию, с учетом случайного подбора контрольного значения для соответствующего значения счетчика.
counters_hash	Контрольное значение для счётчика	32 бита	
data_hash	Контрольное значение для данных	32 бита	Из рассуждений выше, предположим, что это поле - имитовставка для зашифрованных данных. Фаззинг методом генерации нас устраивает.

flags	Флаги для передаваемых данных	32 бита	В данном поле - тяжело определить значение каждого бита. Используем метод мутации.
datagram	Передаваемые шифрованные данные	32 байта	Можно проверить, если будет правильно подобрана имитовставка.

В процессе анализа предварительных результатов определено, что основной проблемой фаззинг тестирования рассматриваемого протокола является наличие полей с контрольными значениями.

Этап 3. Разработка шаблона

На третьем этапе необходимо создать шаблон тестирования, соответствующий выбранной на этапе 2 методологии. Процесс создания шаблона тестирования представлен ниже. В качестве примера шаблона использовался template.xml, который идёт в составе фаззера.

```
<DataModel name="CryptoProtocol">
    <Blob name="id" valueType="hex" length="32" lengthType="bits" mutable=
"false" value="e1 10 00 00" />
    <Blob name="counter_low" valueType="hex" length="32" lengthType="bits"
mutable="false" value="00 00 26 3e" />
    <Blob name="counter_high" valueType="hex" length="32" lengthType="bits"
mutable="false" value="00 00 00 01" />
    <Blob name="cnt_imit" valueType="hex" length="32" lengthType="bits"
value="ec 36 4d 7e" />
    <Blob name="data_imit" valueType="hex" length="32" lengthType="bits"
value="00 00 00 00" />
    <Blob name="flags" valueType="hex" length="32" lengthType="bits" value="60
00 25 00" />
    <Blob name="ipdatagram" valueType="hex" length="32" lengthType="bytes" />
</DataModel>
```

В вышеприведенном тексте описана модель данных "CryptoProtocol", в которой определены поля исследуемого протокола, указаны данные, которые нужно изменять, и те, которые изменять не имеет смысла. Изменению не подлежат поля с идентификатором и флагами.

```
<StateModel name="TheState" initialState="Initial">
    <State name="Initial">
        <Action type="output">
            <DataModel ref="CryptoProtocol"/>
        </Action>
    </State>
</StateModel>
```

В блоке текста <StateModel> описываются действия, которые должны выполняться фаззером. В данном случае фаззер выполняет отправку данных, сформированных в нашей модели "CryptoProtocol".

```
<Test name="Default">
  <StateModel ref="TheState"/>
  <Publisher class="Udp">
```

```
    <Param name="Host" value="192.168.100.2" />
    <Param name="Port" value="10000" />
  </Publisher>
  <Logger class="File">
    <Param name="Path" value="logs"/>
  </Logger>
</Test>
```

Собственно, в блоке <Test> происходит основная настройка: указание модели состояний, указание адреса и порта для нашего сервиса и создание логов. Данный блок является последним при описании шаблона фаззинг тестирования.

Этап 4. Проведение фаззинг тестирования

На четвертом этапе были проведены работы по тестированию и контролированию ответной реакции.

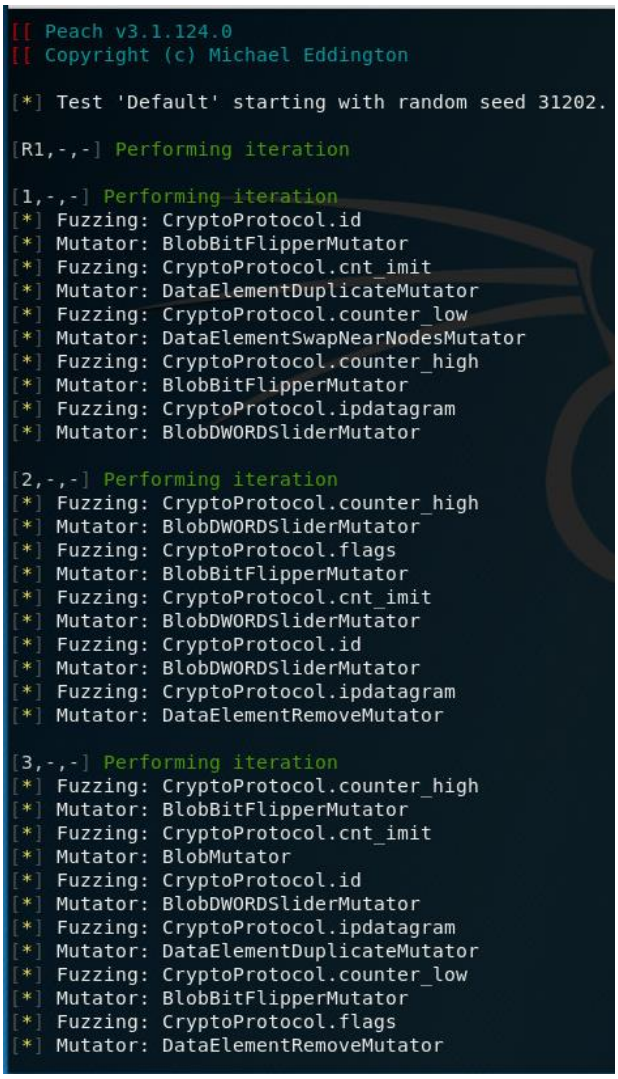

Рис. 5. Работа фаззера

При запуске фаззера отображается каждая итерация и какие комбинации полей используются (см. рисунок 5). Далее был запущен сниффер Wireshark, чтобы контролировать работу фаззера и ответную реакцию (см. рисунок 6).

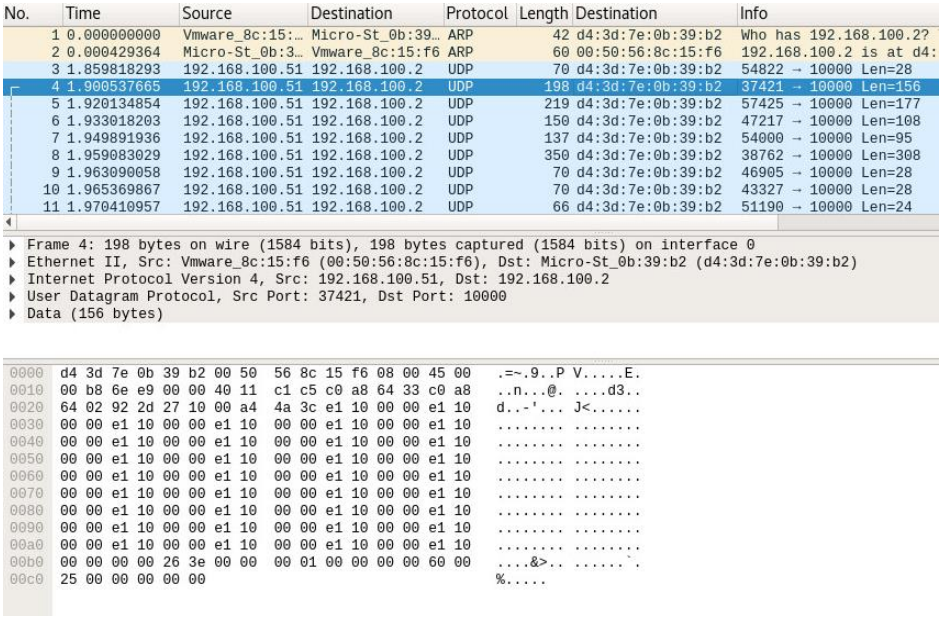

Рис. 6. Дамп трафика в процессе фаззинга

В процессе анализа сетевого трафика был настроен фильтр для отображения только входящих пакетов от исследуемого объекта (IP=192.168.100.2). В результате было выявлено, что в сетевом трафике (см. рисунок 7) отсутствует ответная реакция на исходящие от фаззера пакеты.

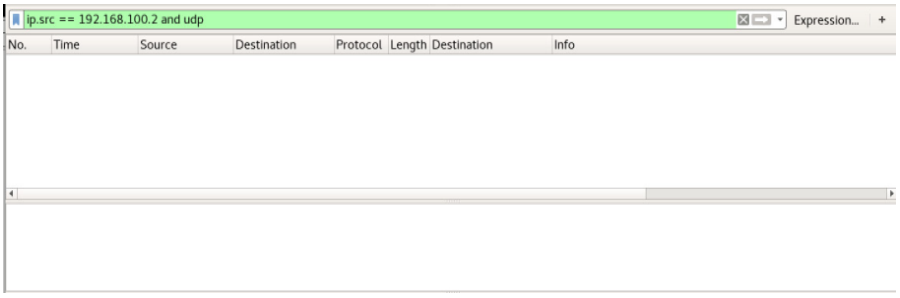

Рис. 7. Ответная реакция

Таким образом, отсутствие ответной реакции от исследуемого устройства подразумевает, что у сетевого устройства существуют правила отбраковки некорректных пакетов. Скорее всего, причиной данной отбраковки пакетов является срабатывание контроля целостности пакетов с использованием имитовставки. Из этого можно сделать вывод, что при фаззинг тестировании защищенного протокола взаимодействия существует необходимость подбора и формирования контрольных значений для прохождения встроенных механизмов защиты и дальнейшего тестирования реализации защищенного протокола, что является задачей для дальнейших исследований.

Библиографический список

1. Фаззинг. [Электронный ресурс]: 2017. URL: https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B7%D0%B7%D0%B8%D0%BD%D0%B3 (дата обращения 27.09.2017)

2. Майкл Дж.Д. Саттон, Адам Грин, Педрам Амини. Fuzzing. Исследование уязвимостей методом грубой силы / пер. с англ. А.Коробейников. СПб.: Символ-Плюс, 2009.

3. Peach Community Edition Official Page. URL: http://www.peach.tech/resources/peachcommunity/

Москвичев Никита Петрович
ООО “Открытые решения»,
г. Пенза, Россия

Moskvichev N.P.
LLC Otkrytye resheniya,
Penza, Russia

Кульков Иван Васильевич
ООО “Открытые решения»,
г. Пенза, Россия

Kulkov I.V.
LLC Otkrytye resheniya,
Penza, Russia

Болдырев Александр Геннадьевич
ООО “Открытые решения»,
г. Пенза, Россия

Boldyrev A.G.
LLC Otkrytye resheniya,
Penza, Russia