

Научная статья

УДК 004.056.5

DOI:10.31854/1813-324X-2023-9-6-59-67



## Предложения по построению универсального фаззера протоколов

**Дмитрий Александрович Васинев** ✉, vda33@academ.msk.rsnet.ru  
 **Михаил Викторович Соловьев**, saintsdertr@gmail.com

Академия Федеральной службы охраны Российской Федерации,  
Орел, 302015, Российская Федерация

**Аннотация:** В статье исследуется проблема обеспечения информационной безопасности в сфере телекоммуникаций с использованием метода фаззинга. Проведен анализ современных программных продуктов, предназначенных для тестирования методом фаззинга, выявлены их недостатки и предложен подход к созданию универсального мутационного фаззера протоколов. Способ формирования тестовых данных позволяет автоматизировать процесс поиска уязвимостей в телекоммуникационных протоколах и программном обеспечении. Его новизна заключается в формировании тестовых конструкций на основе параметров полей телекоммуникационных протоколов. Предложенное решение фаззера позволяет формировать вектор атаки на основе известных параметров, представленных в банке данных угроз, так и модифицировать эти вектора атак.

**Ключевые слова:** информационная безопасность, фаззинг, уязвимости, телекоммуникационные протоколы, генетический фаззер

**Ссылка для цитирования:** Васинев Д.А., Соловьев М.В. Предложения по построению универсального фаззера протоколов // Труды учебных заведений связи. 2023. Т. 9. № 6. С. 59–67. DOI:10.31854/1813-324X-2023-9-6-59-67

## Proposals for Universal Protocol Fuzzer Construction

**Dmitriy Vasinev** ✉, vda33@academ.msk.rsnet.ru  
 **Mikhail Solovev**, saintsdertr@gmail.com

The Academy of the Federal Guard Service of Russian Federation,  
Orel, 302015, Russian Federation

**Abstract:** The article studies the problem of ensuring information security in the field of telecommunications using the phasing method. The analysis of modern software products designed for testing by the fuzzing method is carried out, their disadvantages are revealed, and an approach to the creation of a universal mutation protocol fuzzer is proposed. The method of test data formation allows to automate the process of searching for vulnerabilities in telecommunication protocols and pro-software. Its novelty lies in the formation of test constructs on the basis of parameters of fields of telecommunication protocols. The proposed fuzzy solution allows to form an attack vector on the basis of known parameters presented in the threat database, as well as to modify these attack vectors.

**Keywords:** information security, fuzzing, vulnerabilities, telecommunication protocols, genetic fuzzing

**For citation:** Vasinev D., Solovev M. Proposals for Universal Protocol Fuzzer Construction. *Proceedings of Telecommunication Universities*. 2023;9(6):59–67. DOI:10.31854/1813-324X-2023-9-6-59-67

## Введение

В связи с постоянным появлением новых угроз и уязвимостей значительные усилия специалистов по информационной безопасности посвящены разработке методов поиска уязвимостей в телекоммуникационных протоколах, программном обеспечении и информационных системах, функционирующих в объектах критической информационной инфраструктуры (КИИ). Предпосылкой к появлению уязвимостей является значительная размерность пространства состояний параметров протокольных конструкций, участвующих в обработке и передаче данных. Таким образом, актуальной задачей является автоматизация процесса поиска уязвимостей в протокольных конструкциях, разработка автоматических и автоматизированных методов, уменьшающих рутинные функции специалиста по информационной безопасности, связанные с поиском уязвимостей в телекоммуникационном оборудовании объектов КИИ.

## Анализ существующих решений

В современных исследованиях в области тестирования программных средств актуальным направлением является фаззинг [1–3]. Под термином «фаззинг» (эквив. на русск. – тестирование) понимается методология тестирования, традиционно применяемая для исследования программ, в которых в качестве данных для анализа защищенности объекта применяются значения, выходящие за рамки корректных [1]. Под фаззером понимается специализированное программное обеспечение (ПО), на основе которого из исходных данных выполняется формирование тестируемой последовательности, осуществляется ее передача для обработки тестируемым объектам с целью обнаружения отклонений от нормального функционирования. Традиционным стало применение методологии фаззинга для тестирования безопасности программных средств.

Таким образом, исходя из работ [1, 2], наиболее близкой по функциональным возможностям поиска уязвимости в протокольных конструкциях является тестирование программ методом фаззинга. Известны смешанные программные и протокольные методы решения этой задачи. В рамках исследования разрабатываются функциональные и технические элементы, позволяющие осуществлять поиск уязвимостей в протокольных стеках объектов КИИ.

Основным отличием предлагаемого решения является тестирование не отдельного протокола, а группы протоколов, с учетом существующих служебных и сигнальных каналов передачи данных. Для ее решения необходимо провести анализ принципов построения таких анализаторов (рисунок 1), существующих функциональных возможностей (таблица 1), выделить направления их совершен-

ствования, сформировать предложения по совершенствованию методов тестирования протокольных стеков, осуществить верификацию предлагаемых решений.

Представленная на рисунке 1 классификация видов фаззинга позволяет выделить функциональные особенности, характерные как для программных, так и протокольных фаззеров. Для тестирования функциональных возможностей протокольных конструкций используются методы динамического изменения тестовых последовательностей – мутационные и генетические алгоритмы.

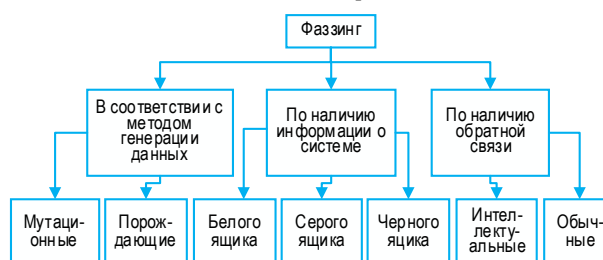


Рис. 1 Классификация видов фаззинга

Fig. 1. Classification of Fuzzing Methods

При тестировании протоколов в случае неизвестной структуры и правил взаимодействия – применяется методология «черного ящика» (BB – аббр. от англ. Black Box), при наличии информации о логической и процедурных характеристиках протокола – методы «серого» (GB – аббр. от англ. Grey Box) и «белого» (WB – аббр. от англ. White Box) ящика. Наличие обратной связи является ключевым признаком проверки функционирования протокола или выяснения степени влияния на протокол тестовой конструкции.

В работах [3, 4] выделены основные функциональные особенности, характерные для программных фаззеров, а также характеристики, связанные с обработкой протоколов разных типов, возможности формирования структуры пакета и генерации тестовых последовательностей, наличие исходного кода. Развитие выделенных программных средств идет с расширением их функционала, в том числе и в области работы с протокольными конструкциями, а также в направлении качества проведенных тестовых исследований, за счет применения современных математических методов: марковских цепей, машинного обучения, генетических алгоритмов.

В таблице 1 проведен анализ функциональных характеристик фаззеров и их классификация, в соответствии с введенными ранее на рисунке 1 признаками. Функциональные возможности фаззеров с возможностью описания структуры пакета в соответствии с вышеприведенной классификацией считаются характерным признаком порождающих типов.

ТАБЛИЦА 1. Анализ фаззеров в приложении к анализу протоколов

TABLE 1. Fuzzer Analysis in an Application to Protocol Analysis

Фаззер, тип [ссылка]	Генерация пакетов	Язык	Поддержка протоколов	Описание структуры пакета	Мутационные без описания структуры	Тип обратной связи
AFL, GB <a href="http://lcamtuf.coredump.cx/afl">[http://lcamtuf.coredump.cx/afl]</a>	-	C, C++	-	-	Нет	Анализ хода выполнения программы
Peach Fuzzer, WB <a href="https://www.peach.tech">[https://www.peach.tech]</a>	+	XML, C#	Не ограничен	+	Да	Анализ хода выполнения программы
Honggfuzz, GB <a href="https://github.com/google/honggfuzz">[https://github.com/google/honggfuzz]</a>	-	C	-	-	Нет	Анализ хода выполнения программы
Boofuzz, GB <a href="https://boofuzz.readthedocs.io/en/stable">[https://boofuzz.readthedocs.io/en/stable]</a>	+	Python	UDP, TCP	+	Нет	Анализ ответа
Sulley, GB <a href="http://www.fuzzing.org/wp-content/SulleyManual.pdf">[http://www.fuzzing.org/wp-content/SulleyManual.pdf]</a>	+	Python	UDP, TCP	+	Нет	Анализ ответа
Radamsa, BB <a href="https://gitlab.com/akihe/radamsa">[https://gitlab.com/akihe/radamsa]</a>	+	C	Не ограничен	-	Да	Нет
Cert BFF, GB <a href="https://github.com/CERTCC/certifuzz">[https://github.com/CERTCC/certifuzz]</a>	-	Python	-	+	Да	Анализ хода выполнения программы
Pulsar, GB [7]	+	Python	Не ограничен	+	Да	Анализ ответа
AFLNet, GB <a href="https://github.com/aflnet/aflnet">[https://github.com/aflnet/aflnet]</a>	-	C	Не ограничен	-	Да	Анализ хода выполнения программы
Netzob, GB <a href="https://github.com/netzob/netzob">[https://github.com/netzob/netzob]</a>	+	Python	Не ограничен	+	Да	Анализ ответа
ProtoFuzz, GB <a href="https://github.com/trailofbits/protofuzz">[https://github.com/trailofbits/protofuzz]</a>	-	Python	-	+	Нет	Анализ тестируемого объекта
Doona, GB <a href="https://github.com/wireghoul/doona">[https://github.com/wireghoul/doona]</a>	+	Perl	Не ограничен	+	Нет	Наличие ответа

Мутационные фаззеры на основе известных структурных конструкций пакетов порождают новые тестовые последовательности, применяемые далее в работе.

По наличию обратной связи программные средства разделяются по способу контроля хода выполнения программы следующими функциональными возможностями:

- "Анализ хода выполнения программы" и "Анализ ответа";
- "Анализ тестируемого объекта" и "Наличие ответа".

Из таблицы видно, что часть существующих решений разработана не только для анализа ПО, но и для анализа протоколов.

Программы без возможности генерации пакета могут осуществлять частичное тестирование уязвимостей телекоммуникационных протоколов. Например, во время анализа исходного кода могут быть затронуты участки, отвечающие за обработку трафика, что порождает отправку или получение пакета.

Большая часть телекоммуникационных фаззеров сосредоточена на тестировании высокоуровневых текстовых протоколов и назвать их универсальными нельзя. Таким образом для тестирования протоколов и групп протоколов лучшими функциональными характеристиками обладают: Radamsa, Pulsar, Netzob. Выделенные средства для тестирования вне зависимости от применяемого протокола обладают возможностью формирования сетевых пакетов и мутационным подходом к изменению тестовых данных. Фаззеры с обратной связью (в виде анализа хода выполнения программы) имеют ограничения, связанные с универсальностью применения. В случае наличия обратной связи для каждой разновидности платформы требуется разработка агента для обратной связи, что усложняет реализацию.

Фаззеры Pulsar и Netzob формируют данные на основе изменения графа модели взаимодействия протокола. Эти программы анализируют входной трафик, формируют изменения графа модели взаимодействия протокола с помощью математиче-

ских методов кластеризации, машинного обучения и марковских моделей. Основное внимание в данных фаззерах уделено моделированию процедурной характеристики работы протокола, а не изменению структуры пакетов.

Инструмент для генерации и мутации входных данных – Radamsa – предназначен исключительно для работы с входными данными и не предназначен для получения ответов от сервера. Однако он сам по себе не имеет функциональности для отправки пакетов или взаимодействия с серверами.

Таким образом, из результатов анализа следует, что универсального средства тестирования протоколов, не требующего знания о структуре протокола, не существует, что говорит об актуальности задачи формирования фаззера для протокола или группы протоколов. Направление развития программных фаззеров является применение математических методов для снижения рутинных функций инженера, а также разработка способов автоматизации процесса тестирования, что также может учтено при развитии решений, предназначенных для проверки протокольных конструкций.

### Фаззинг протоколов

Проанализированные функциональные возможности средств фаззинга протоколов, представленные в таблице 1, позволяют дополнить уже существующую классификацию (рисунок 1) новыми классификационными признаками, представленными на рисунке 2.



Рис. 2. Классификация видов фаззинга в соответствии с целевой областью телекоммуникаций

Fig. 2. Classification of Fuzzing Types According to the Target Area of Telecommunications

В зависимости от уровня модели взаимодействия открытых систем (ЭМВОС), на котором находится объект тестирования, принято выделять фаззеры протоколов и веб-приложений.

Под фаззерами протоколов принято понимать программы, взаимодействующие на соответствующих уровнях стека коммуникационных протоколов. В настоящий момент существуют следующие подходы к их реализации: в фаззере протоколов Reach Fuzzer, в специальном файле конфигурации заранее описывается структура пакета и обозначаются типы данных для каждого из них. В ходе тестирования создаются пакеты согласно шаблону. В данном решении в качестве метрики тестирования используются коды завершения программы, получаемые с помощью дебаггера на сервере.

Схема работы фаззера показана на рисунке 3, особое внимание следует обратить на то, что данный инструмент используется преимущественно для работы протоколами прикладных уровней, а транспортировку тестовых конструкций осуществляют с помощью протоколов TCP, UDP. Это ограничивает сферу его применения протоколами прикладного уровня.

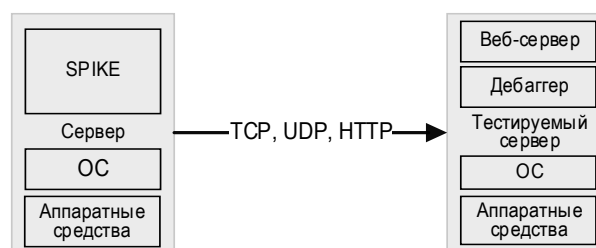


Рис. 3. Схема работы генерационного фаззера

Fig. 3. Schematic Diagram of the Generating Fuzzer Operation

С точки зрения протокольных программ для тестирования отдельные исследования [7] выделяют методы генерации пакетов, применяемые в AFLNet, которые могут быть реализованы в программах тестирования протоколов. В фаззере протоколов AFLNet применяется метод генерации программного фаззера AFL, где с помощью средств операционной системы имитируется получение тестируемым приложением пакетов. Таким образом, фаззер протоколов AFLNet имитирует функции протокольного фаззера на основе функции программного фаззера.

При генерации данных мутационным фаззером выделяются следующие простейшие операции, порождающие изменение входных данных – мутацию: вставка (случайного) символа; удаление символа; инверсия бита в представлении символа; случайный выбор мутации.

Помимо операций над битами, рассмотренные действия могут быть применены к другим единицам данных, например, в программном средстве AFL, помимо вышеописанных действий над битами, аналогичные операции применяются и к байтам.



Как и в AFL, в качестве метрики для фаззинга используется покрытие кода. Чем большая часть кода затронута в ходе тестирования, тем более безопасной считается реализации протокола в программе. Схема реализации процесса тестирования представлена на рисунке 4.

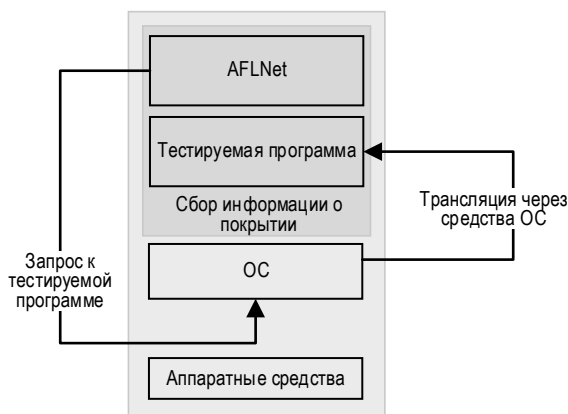


Рис. 4 Схема работы мутационного фаззера с трансляцией средствами ПО

Fig. 4. Scheme of Operation of Mutation Fuzzers with Translation by Software Means

Особенностью фаззера AFLNet является то, что создаваемые для тестирования пакеты с помощью средств ОС транслируются на вход тестируемой программы, установленной на той же ЭВМ. При реализации данных методов в качестве объекта тестирования всегда рассматривается программа, функционирующая на 7 уровне ЭМВОС.

### Метод тестирования протоколов

В случае телекоммуникационного оборудования, тестирование может быть реализовано как при пересылке трафика, так и при обработке служебных (сигнальных) протоколов. При обработке полученного трафика ошибка может вызвана не только в программных модулях различных уровнях модели ЭМВОС, но и в системном ПО, например – в сетевых драйверах.

Для тестирования всех потенциально уязвимых протоколов сетевого взаимодействия, предлагается новый метод тестирования, предполагающий не только выявление ошибок в системном ПО, но и в конфигурации телекоммуникационного оборудования и иных компонентах обработки трафика, вместе формирующих сетевую коммуникационную инфраструктуру объекта КИИ.

При реализации данного подхода предлагается располагать устройство с фаззером в точке максимальной концентрации трафика в сетевом фрагменте или создавать такую точку, например, на основе технологии зеркалирования порта (Switch port analysis) и создавать множество (корпус) тестов из уже сгенерированного трафика, тем самым решая задачу формирования исходных параметров для тестирования.

Предполагается, что если пакеты, взятые за основу, сформированы согласно правилам (валидны), то и порожденные в ходе фаззинга пакеты с большей вероятностью не будут отброшены. Помимо этого, используя трафик системы в качестве исходного, нет необходимости производить его мониторинг для получения списка узлов и тестируемых служб. В этом случае достаточно лишь не изменять поля отправителя и получателя пакета, тогда сформированный трафик дойдет до получателя.

Для автоматизации процесса генерации тестовых фреймов и пакетов предполагается работать с заголовками пакетов в качестве базовых единиц. В связи с данными предложениями, предлагается модифицировать их структуру и организовать процесс генерации согласно рисунку 5. Получив исходное множество для генерации параметров (популяции), фаззер, согласно генетическому алгоритму, выбирает некоторое подмножество (популяции) и применяет к ним операцию объединения (кроссовера), объединяя их информацию за исключением заголовков, необходимых для продвижения пакета по сети (IP-адрес источника и назначения в примере на рисунке 5).

Определяется поле протокола, для которого будет выполнена операция мутации. На рисунке 5 мутация применена к заголовку смещения. В качестве пакетов для скрещивания выбирается IP-пакет С и IP-пакет D из множества IP-пакетов А, В, С, D. В качестве места объединения выбрано число 6, первые 6 параметров заголовка пакета С объединяются с оставшимися заголовками пакета D. Случайным образом принимается решение о необходимости мутации и, в качестве мутируемого заголовка, выбирается поле “Смещение” (рисунок 5).

Сгенерированный пакет отправляется тестируемой системе, причем для реализации данной операции необходимо выбирать программные средства, позволяющие создавать «неправильные» по спецификации пакеты (изменение порядка следования заголовков, несоответствие фактической длины пакеты и значения поля).

Эмпирической функцией эффективности предполагается выбрать время ответа, следовательно, чем сильнее «нагрузил» пакет систему, тем более он эффективен.

Уточним, что продемонстрированный метод применим не только к IP-пакетам (сетевому уровню), он и к единицам данных любого уровня, что демонстрируется на рисунке 6, где представлен процесс тестирования на основе единиц, передаваемых данных различного типа – пакетов и фрейма. В ходе операции кроссовера MAC-адрес назначения IP-пакета заменяется на адрес фрейма, а операции мутации подвергается поле инкапсулированного протокола.

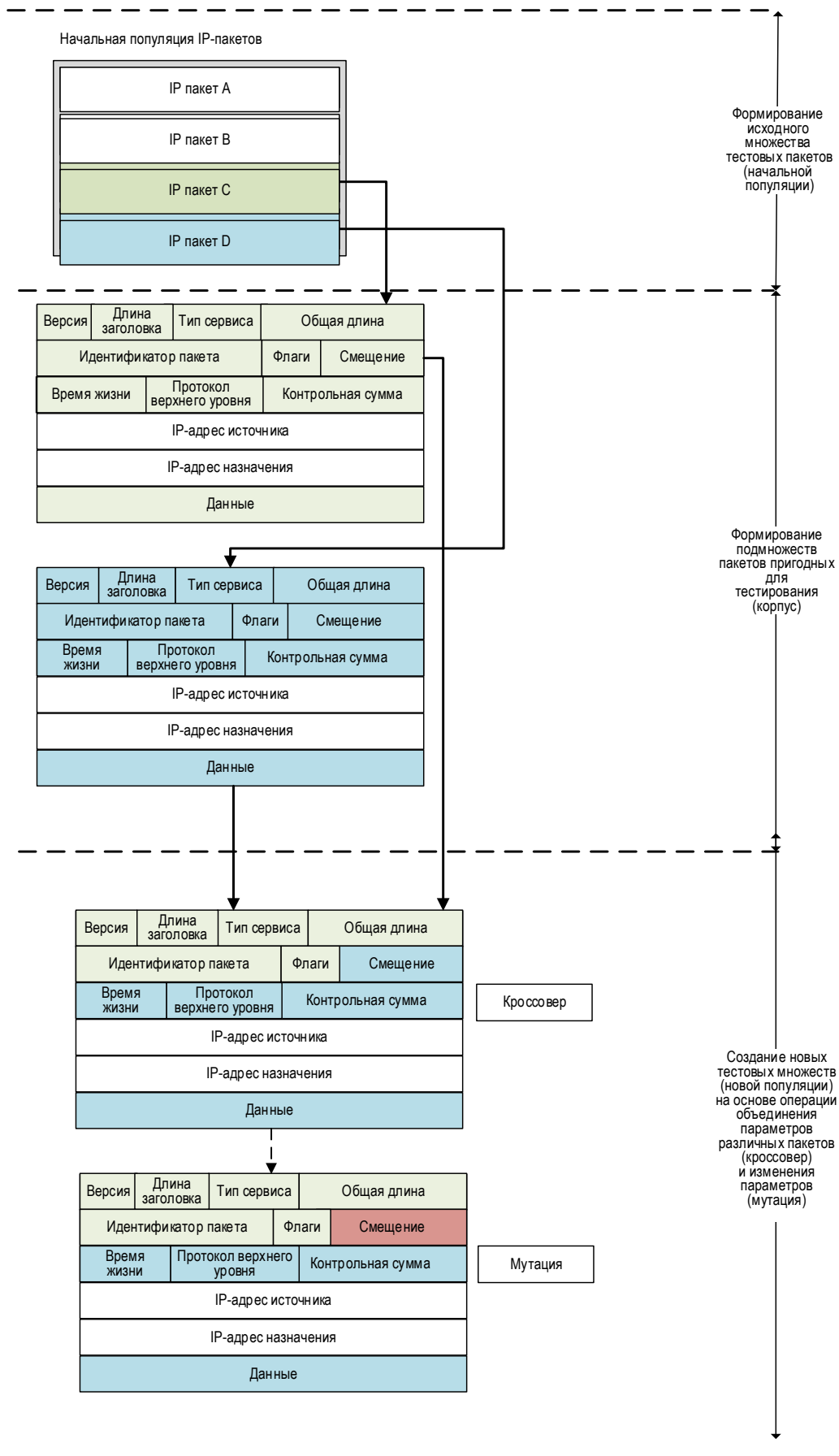


Рис. 5. Процесс генерации данных согласно эволюционному алгоритму на сетевом уровне

Fig. 5. The Process of Data Generation According to Evolutionary Algorithm at Network Layer

Предлагаемый на рисунке 6 метод тестирования протоколов или групп протоколов на основе генетического алгоритма, на примере протоколов VTP, IP.

Пусть из множества пакетов перехваченного в точке концентрации трафика случайным образом выбирается IP-пакет – D и VTP фрейм (рисунок 6). На следующем шаге выбирается случайное число размером меньшей единицы данных – VTP фрейма, на рисунке 6 выбрано число 1.

Следующим шагом производится процедура кроссовера, для этого от VTP фрейма выбирается только 1 заголовок (согласно ранее выбранному числу, выделено зеленым цветом на рисунке 6), остальные заголовки формируемой тестовой единицы данных копируются у IP-пакета – D).

При формировании нового пакета не могут быть взяты в качестве точки конкатенации адресные заголовки сетевого уровня для обеспечения доставки пакета существующей службе.

На следующем этапе случайным образом выбирается, будет ли применена мутация. На приведенном примере мутация применяется к случайно выбранному заголовку (инкапсулированный протокол), изменяя его значение. В качестве мутации могут быть применены как алгоритмы мутации, реализованные в программном средстве AFL, так и иные алгоритмы.

Алгоритм реализации универсального программного средства тестирования протоколов или групп протоколов, на основе генетических методов представлен на рисунке 7.

**Описание алгоритма генерации данных согласно эволюционному алгоритму**

Перехват трафика, формирования корпуса тестирования. В качестве точки перехвата видится необходимым выбирать точку сети с максимальной концентрацией трафика (коммутатор, маршрутизатор, межсетевой экран), либо формировать такую точку самостоятельно.

В ходе дальнейшего тестирования в исходном множестве тестовых пакетов будет присутствовать максимальное разнообразие реального трафика системы. При генерации данных наличие большого количества трафика позволит добиться увеличения числа тестируемых сервисов и устройств (при применении процедуры кроссовера или мутации, при этом поля заголовков источника и назначения не затрагиваются).

Выбор 2-х единиц данных – исходного множества для формирования тестовых пакетов, случайным образом или в соответствии с некоторой метрикой (например, временем ответа). В случае использования метрики для реализации генетического фаззера следует рассматривать функцию случайного выбора с заданными весами.

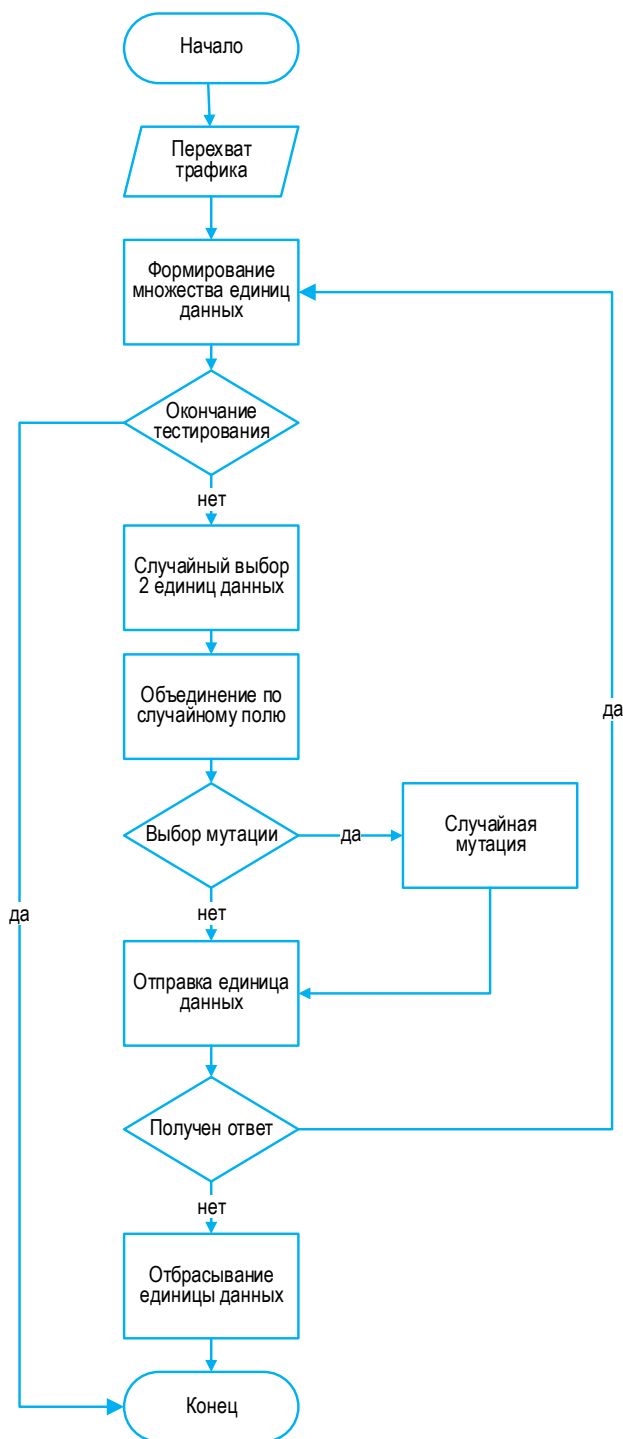


Рис. 7. Алгоритм генерации данных согласно эволюционному алгоритму

Fig. 7. Data Generation Algorithm According to Evolutionary Algorithm

Выбор номера случайного поля и объединение единиц данных соответственно. Реализация данной операции возможна как с учетом уровня протокола согласно ЭМВОС, так и без нее. Необходимо принять решение о объединении только общих заголовков или объединении без учета структуры каждой из единиц исходных данных.

Случайный выбор мутации. Мутация и решение о ее применении выбираются случайным образом. В качестве мутаций, в отличие от AFL, предлагается выбирать не универсальные операции над байтами, а основанную на типе данных генерацию потенциально некорректных значений. Примером реализации такого подхода является фреймворк Scapy (<https://scapy.readthedocs.io/en/latest/usage.html#fuzzin>).

Отправка сгенерированного пакета. В качестве потенциальной метрики эффективности фаззера, предлагаемого к реализации, рекомендуется выбрать время ответа сервера, которое, при задании известной задержке передачи, характеризует время обработки пакета сервером. Если же пакет вызовет отказ в обслуживании, данное явление будет возможно вычислить по отсутствию дальнейших ответов на запрос. При реализации тестирования методом черного ящика предлагаем периодически проверять состояние тестируемого сервиса или устройства посредством протокола ICMP.

Помещение пакета в корпус или отбрасывание. При генерации валидного пакета следует поместить его в корпус тестирования для увеличения потенциального разнообразия данных для генетического алгоритма. При использовании метрики предлагаем в качестве ее рассматривать значение времени ответа, нормированное относительно задержки передачи в сети.

Одним из направлений развития алгоритма является применение математического аппарата Марковских цепей для осуществления случайной (стохастической) выборки, осуществляемой по нормальному (равномерному) закону распределения. Такими элементами случайной выборки могут быть протоколы, группы протоколов, параметров протоколов для изменения (мутации), которые являются последовательностью дискретных случайных величин, выражение (1), где  $X_n$  – множество дискретных величин, протоколов, групп протоколов, параметров протоколов телекоммуникационных систем.

$$P(X_n = x_n | X_{n-1} = i_{n-1}, X_{n-2} = x_{n-2}, \dots, X_0 = x_0) = P(X_n = x_n | X_{n-1} = x_{n-1}). \quad (1)$$

Матрица переходных вероятностей формируется на первоначальном этапе исследований на основе нормального закона, равномерного закона распределения, в дальнейшем экспертным способом, при необходимости исследования известных векторов атак в соответствии с известными угрозами, например, в соответствии с банком данных угроз безопасности информации ФСТЭК. Предпочтительный вариант решения формируется на основе экспериментальных исследований.

## Заключение

В статье рассмотрены вопросы поиска уязвимостей в телекоммуникационных протоколах объектов КИИ. В ходе анализа установлено, что ближайшими аналогами таких средств является средства тестирования программ на уязвимости – программный фаззинг. Проведена классификация существующих средств и видов фаззинга, на ее основе установлены функциональные возможности существующих решений. Перспективное направление развития таких средств связано с применением математических методов для автоматизации процесса тестирования и поиска уязвимостей.

## Список источников

1. Девянин П.Н., Тележников В.Ю., Хорошилов А.В. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем // Труды Института Системного Программирования РАН. 2021. Т. 33. № 5. С. 25–40.
2. Вареница В.В., Марков А.С., Савченко В.В. Практические аспекты выявления уязвимостей при проведении сертификационных испытаний программных средств защиты информации // Вопросы Кибербезопасности. 2021. № 5 (45). С. 36–44.



3. Манеев А.О., Спивак А.И. Стохастическое тестирование программного обеспечения для поиска уязвимостей // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21. № 6. С. 895–902.
4. Козачок А.В., Козачок В.И., Осипова Н.С., Пономарев Д.В. Обзор исследований по применению методов машинного обучения для повышения эффективности фаззинг-тестирования // Вестник Воронежского государственного университета. Серия: системный анализ и информационные технологии. 2021. №4. С. 83–106
5. Саттон М., Амини П., Грин А. Fuzzing: исследование уязвимостей методом грубой силы. Пер. с англ. М.: Символ-Плюс, 2017. 554 с.
6. Gascon H., Wressnegger C., Yamaguchi F., Arp D., Rieck K. PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols // Proceedings of the 11th International Conference on Security and Privacy in Communication Networks (SecureComm, Dallas, USA, 26–29 October 2015). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham: Springer, 2015. Vol. 164. PP. 330–347. DOI:10.1007/978-3-319-28865-9\_18
7. Шарков И.В., Падарян В.А., Хенкин П.В. Об особенностях фаззинг-тестирования сетевых интерфейсов в условиях отсутствия исходных текстов // Труды Института Системного Программирования РАН. 2021. Т. 33. № 4. С. 211–226.


## References


1. Devyanin P.N., Telezhnikov V.Yu., Khoroshilov A.V. Formation of methodology of safe system software development on the example of operating systems. *Proceedings of ISP RAS*. 2021;33(5):25–40.
2. Varenitsa V.V., Markov A.S., Savchenko V.V. Practical aspects of vulnerability detection during certification testing of information protection software. *Voprosy Kiberbezopasnosti*. 2021;5(45):36–44.
3. Maneev A.O., Spivak A.I. Stochastic Software Testing for Vulnerability Analysis. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics, Mechanics and Optics*. 2021;21(6):895–902.
4. Kozachok A.V., Kozachok V.I., Osipova N.S., Ponomarev D.V. Overview of Studies on the Application of Machine Learning Methods to Improve the Efficiency of Fusing Testing. *Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies*. 2021;4:83–106.
5. Sutton M., Amini P., Green A. *Fuzzing: vulnerability research by brute force method*. Translated from English. Moscow: Symbol-Plus Publ.; 2017. 554 p.
6. Gascon H., Wressnegger C., Yamaguchi F., Arp D., Rieck K. PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols. *Proceedings of the 11th International Conference on Security and Privacy in Communication Networks, SecureComm, 26–29 October 2015, Dallas, USA. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol.164*. Cham: Springer; 2015. p.330–347. DOI:10.1007/978-3-319-28865-9\_18
7. Sharkov I.V., Padarian V.A., Henkin P.V. Features of Fuzzing Network Interfaces Without Source Codes. *Proceedings of ISP RAS*. 2021;33(4):211–226.

Статья поступила в редакцию 21.11.2023; одобрена после рецензирования 29.11.2023; принята к публикации 06.12.2023.

The article was submitted 21.11.2023; approved after reviewing 29.11.2023; accepted for publication 06.12.2023.

## Информация об авторах:

**ВАСИНЕВ** | кандидат технических наук, доцент, сотрудник Академии ФСО России  
**Дмитрий Александрович** |  <https://orcid.org/0009-0004-7030-5421>

**СОЛОВЬЕВ** | сотрудник Академии ФСО России  
**Михаил Викторович** |  <https://orcid.org/0009-0006-6472-1167>