УДК 004.7 DOI: 10.17587/it.25.131-142

И. С. Петров, аспирант, e-mail: ipetrov@cs.msu.ru,
Московский государственный университет им. М. В. Ломоносова

Системы обнаружения скомпрометированных коммутаторов в программно-конфигурируемых сетях

Скомпрометированные коммутаторы являются серьезной угрозой безопасности сети, так как они могут использоваться злоумышленником для проведения широкого спектра атак на сеть. В статье представлен сравнительный анализ существующих систем обнаружения скомпрометированных коммутаторов в программно-конфигурируемых сетях, отмечены достоинства и недостатки каждой из них.

Ключевые слова: *программно-конфигурируемые сети, системы обнаружения вторжений, скомпрометированные коммутаторы*

Введение

Программно-конфигурируемые сети[1] (ПКС), по мнению ведущих производителей сетевого оборудования, являются одним из самых перспективных направлений сетевой индустрии на данный момент [1].

В ПКС уровни управления и передачи данных разделяются за счет переноса функций управления на отдельное устройство — контроллер. На сетевых устройствах — коммутаторах — используются специальные таблицы потоков, в которых контроллер задает правила для маршрутизации и/или модификации пакетов.

Одним из важных элементов концепции ПКС является протокол OpenFlow [2] для программирования и управления сетевыми устройствами. Протокол OpenFlow предполагает использование на коммутаторах специальных таблиц маршрутизации, в которые контроллер загружает правила для маршрутизации и/или модификации пакетов, проходящих через коммутатор. Такие таблицы называются таблицами потоков.

С появлением нового вида компьютерных сетей появляются и новые виды уязвимостей, которые могут привести не только к материальным потерям, но и к утрате репутации и имиджа компаний, использующих подобные сети. Следовательно, во избежание негативных последствий компьютерных атак необходимо проводить анализ защищенности протоколов ПКС и разрабатывать механизмы защиты таких сетей от атак различных типов.

Одной из возможных угроз безопасности сети является наличие в ПКС скомпрометированных коммутаторов. Под скомпрометированным коммутатором понимается некоторый коммутатор ПКС, управляемой контроллером, который в действительности находится под контролем злоумышленника. Возможность компрометации коммутатора обусловлена тем, что сетевое оборудование может иметь уязвимости [3, 4], которые могут быть использованы злоумышленником для получения контроля над коммутатором.

Несмотря на то что все функции управления сетью вынесены на контроллер, компрометация коммутатора является серьезной угрозой безопасности всей сети, так как атакующий может использовать подконтрольные ему коммутаторы для проведения широкого спектра атак как на контур данных, так и на контур управления [4, 5].

Сложность задачи обнаружения скомпрометированного коммутатора заключается в том, что подобный коммутатор невозможно обнаружить средствами аутентификации устройства. Это обусловлено тем, что, получая контроль над коммутатором, злоумышленник получает доступ к криптографическим ключам, находящимся

[1] Англ. аналог — Software Defined Network.

в памяти коммутатора. Далее, используя эти ключи, злоумышленник может провести процедуру аутентификации скомпрометированного коммутатора, а любой коммутатор, прошедший процедуру аутентификации, будет считаться легитимным с точки зрения контроллера.

На данный момент существует множество систем обнаружения скомпрометированных коммутаторов [6—15]. Такие системы используют различные механизмы для проверки корректности работы контура данных. К таким механизмам относятся анализ сетевой статистики и тестирование сети с помощью специально созданных пакетов. Также системы обнаружения используют тот факт, что у контроллера ПКС имеется полная информация о топологии сети, свойствах потоков, которые проходят через эту сеть, и правилах обработки сетевого трафика, установленных на коммутаторы.

В статье представлен сравнительный анализ существующих систем обнаружения скомпрометированных коммутаторов в ПКС, отмечены достоинства и недостатки каждой из них. Необходимо отметить, что в обзоре рассматриваются только системы обнаружения коммутаторов, которые уже были скомпрометированы и использованы для различных атак на контур данных. Системы обнаружения процесса компрометации не рассмотрены, так как эта область информационной безопасности уже является хорошо исследованной [16].

1. Критерии сравнения

Ниже сформулированы требования к системе обнаружения скомпрометированных коммутаторов в ПКС, основанные на функциональности OpenFlow-коммутатора, которые мы будем использовать в качестве критериев сравнения систем в этом обзоре.

K1. Обнаружение двух и более скомпрометированных коммутаторов

Система обнаружения должна иметь возможность распознать ситуацию, когда злоумышленник захватил несколько коммутаторов в сети и организовал их согласованную работу так, чтобы избежать обнаружения.

K2. Верификация данных, поступающих от каждого коммутатора

Система обнаружения должна распознавать скоординированные действия скомпрометированных коммутаторов, при которых на контроллер поступает статистика, искаженная таким образом, чтобы скрыть факт наличия атаки, либо заставить систему обнаружения подозревать легитимный коммутатор.

K3. Разграничение вредоносного и легитимного сброса пакетов

Важным критерием сравнения является возможность системы отличать вредоносный сброс пакета на коммутаторе вследствие атаки от легитимного сброса пакета в силу либо наличия в памяти коммутатора соответствующего правила обработки пакетов, либо из-за перегрузки легитимного коммутатора. Если система не способна различать эти случаи сброса пакетов, то обнаружение скомпрометированных коммутаторов будет сопровождаться большим числом ошибок второго рода, когда сброс пакетов из-за перегрузки в сети будет восприниматься как атака.

K4. Отсутствие требования модификаций контура данных

Применение системы обнаружения скомпрометированных коммутаторов не должно требовать изменений в существующих протоколах и логике работы коммутаторов в контуре данных. Изменения в логике работы коммутаторов могут быть дорогостоящими и в некоторых ситуациях являются неприемлемым решением. Поэтому для того чтобы система могла быть применима в реальной сети, необходимо, чтобы она не требовала внесения изменений в существующие протоколы и логику работы коммутаторов.

K5. Обнаружение атаки вне зависимости от ее длительности

Этот критерий описывает возможность системы обнаруживать атаки, время которых незначительно по сравнению со временем жизни потока данных в сети. Например, к таким атакам могут относиться кратковременный сброс пакетов, кратковременная DoS-атака на некоторого пользователя или атака на определенные пакеты некоторого потока в сети.

K6. Независимость от используемых в сети алгоритмов и политик маршрутизации

Этот критерий предполагает возможность системы работать при использовании в сети разнообразных алгоритмов и политик маршрутизации. Поскольку архитектура ПКС дает большую свободу в управлении сетевым трафиком, то заранее предвидеть сложность алгоритмов и политик маршрутизации невозможно. Также система не должна зависеть от применяемых в сети механизмов маршрутизации и оптимизации потоков, таких как агрегация потоков, балансировка нагрузки, перенаправление трафика в случае изменений в топологии из-за ошибок в сети [17].

K7. Отсутствие влияния на атаки

Процедура обнаружения атаки может влиять на состояние сети, например, устанавливать новые правила маршрутизации, генериро-

вать новые служебные пакеты. Поэтому могут возникнуть ситуации, когда процедура обнаружения может повлиять на саму атаку, проводимую в сети. Из-за такого влияния атака может прекратиться, и, таким образом, она не будет обнаружена системой.

2. Системы обнаружения скомпрометированных коммутаторов

2.1. ATPG

Система ATPG[2] [6] — это система, предназначенная для обнаружения ошибок в работе контура данных. Под ошибками понимается некорректное исполнение коммутатором установленных на него правил маршрутизации. Система ATPG обнаруживает ошибки с помощью независимого и полного тестирования всех правил маршрутизации, установленных в сети.

Тестирование правил маршрутизации проводится с помощью так называемых тестовых пакетов — специальных пакетов, посылаемых в сеть системой, для которых заранее известен маршрут. Система устанавливает в сеть дополнительные правила маршрутизации, которые должны отправлять тестовые пакеты обратно на контроллер в конце их маршрута. Таким образом, система ATPG может проверить, что пакеты действительно прошли предполагаемым маршрутом и не были изменены в процессе.

Так как большое число тестовых пакетов может снизить производительность сети, то система ATPG формирует минимальное множество пакетов, которое будет обработано всеми правилами маршрутизации в сети. Минимальность формируемого множества доказана авторами в работе [6].

Для этого система формирует таблицу достижимости между всеми портами в сети. Эта таблица описывает всевозможные маршруты, которые пакеты могут пройти в сети. Также для каждого маршрута известен набор заголовков пакетов, которые могут пройти по этому маршруту.

Для каждого маршрута система случайно выбирает один пакет из множества, соответствующего этому маршруту. В результате формируется множество пакетов $\mathcal{P}$ такое, что для каждого правила маршрутизации найдется пакет $p \in \mathcal{P}$, который обрабатывается этим правилом. Если разные пакеты из этого множества обрабатываются одним и тем же правилом маршрутизации, то алгоритм выбирает минимальное подмножество пакетов, которое также обрабатывается всеми правилами маршрутизации. Иными словами, решается задача покрытия множества подмножествами. Так как известно, что эта задача NP-сложна [18], то для ее решения используется аппроксимационный алгоритм со сложностью $O(n^2)$.

Одним из недостатков этой системы является тот факт, что обнаружение ошибок работы сети с помощью тестовых пакетов может приводить к большим ошибкам второго рода, которые могут возникать вследствие сброса пакетов, происходящего из-за перегрузок в сети. Поскольку в системе не предусмотрено никаких механизмов анализа перегрузок, возникающих в сети, система не сможет отличить вредоносный сброс пакетов атакующим от легитимного сброса из-за перегрузки.

Подход на основе использования тестовых пакетов также может не обнаружить атаки, время жизни которых меньше интервала между тестированием сети. Также система может не обнаружить сложные атаки, которые осуществляются с помощью нескольких скомпрометированных коммутаторов.

Таким образом, система ATPG не удовлетворяет критериям ***K3*** и ***K5***.

2.2. FADE

Система FADE[3] [7] анализирует сетевую статистику, предоставляемую счетчиками правил маршрутизации, и проверяет ее согласованность. Под согласованностью сетевой статистики понимается следующее: разность значений счетчиков правил маршрутизации, обрабатывающих один и тот же поток, должна быть в пределах заранее определенного порогового значения.

Для анализа сетевой статистики система FADE выбирает минимальный набор потоков, которые описывают поведение счетчиков всех правил маршрутизации, установленных в сети. Для нахождения таких потоков система FADE строит граф зависимостей правил маршрутизации в соответствии с топологией и установленными в сети правилами маршрутизации. Граф зависимостей правил маршрутизации — это ориентированный граф, вершинами которого являются правила маршрутизации, а ребрами — возможные переходы пакетов между правилами. Таким образом, каждый поток в сети представлен некоторым путем в графе зависимостей правил маршрутизации.

[2] Automatic Test Packet Generation.

[3] Forwarding Anomaly Detect Environment.

Авторы статьи [7] предполагают наличие в сети агрегирующих правил маршрутизации — правил, которые обрабатывают сразу несколько потоков. Такие правила описываются вершинами графа с несколькими входящими ребрами и одним исходящим.

Из-за наличия таких вершин следует, что потоки, описываемые путями в графе зависимостей правил, объединяются в деревья, и этот граф представляет собой лес, где потоки направлены от листьев к корням деревьев. Везде далее, если не оговорено противное, под термином граф мы будем понимать граф зависимостей правил маршрутизации.

Система FADE выбирает минимальный набор путей, которые покрывают все вершины в графе. Такие пути соответствуют потокам в сети, обработка которых затрагивает все правила маршрутизации. Для каждого таким образом выбранного потока система FADE генерирует набор дополнительных правил маршрутизации и устанавливает его на коммутаторы, обрабатывающие этот поток. Дополнительные правила маршрутизации, установленные системой FADE, используются для сбора сетевой статистики с коммутаторов.

Для того чтобы устанавливаемые системой FADE правила маршрутизации обрабатывали только пакеты соответствующих им потоков, система FADE помечает пакеты выделенными VLAN-метками. На каждом таком правиле маршрутизации установлено временное ограничение — hard-timeout, после истечения которого правило удаляется, и на контроллер отправляется статистика о числе пакетов, обработанных этим правилом.

После сбора статистики система FADE сравнивает значения счетчиков правил маршрутизации и делает вывод о наличии аномалии маршрутизации в сети, если эти значения отличаются более чем на заранее определенную пороговую величину.

Одним из недостатков системы FADE является тот факт, что она не учитывает возможность наличия в сети легитимных сбросов пакетов из-за перегрузок. Также система применима только в том случае, когда на контроллере реализована простая логика маршрутизации, не использующая агрегацию потоков, групповую адресацию, балансировку нагрузки и т. д. Учитывается возможность наличия агрегирующих правил, обрабатывающих сразу несколько потоков, но затем система эти потоки не различает. Не рассматривается групповая маршрутизация, изменение заголовков пакетов и деагрегация потоков, используемая для балансировки нагрузки на коммутаторы.

В статье [7] предполагается, что одна атака нацелена на одно определенное правило маршрутизации, т. е. не учитывается возможность наличия в сети нескольких скомпрометированных коммутаторов.

Также существует возможность, что во время анализа сетевой статистики система FADE своими действиями может повлиять на атаку и, таким образом, ее не обнаружить. Например, если дополнительные правила маршрутизации устанавливаются на скомпрометированный коммутатор, то эти правила могут начать обрабатывать пакеты вместо вредоносных правил, и атака не будет обнаружена. При этом после анализа сетевой статистики, когда дополнительные правила будут удалены с коммутаторов, атака может продолжиться.

Кроме того, в статье не описано, каким образом система реагирует на изменения в сети.

Таким образом, система FADE не удовлетворяет критериям ***K1***, ***K3***, ***K6*** и ***K7***.

2.3. FlowMon

Система FlowMon [8] предназначена для обнаружения скомпрометированных коммутаторов с помощью анализа сетевой статистики, получаемой с портов коммутаторов, и сообщений о новых потоках в сети. Эта система позволяет обнаружить два типа вредоносного поведения коммутатора: нелегитимный сброс пакетов и нарушение политики маршрутизации. Под сбросом пакетов подразумевается, что на скомпрометированный коммутатор устанавливают вредоносное правило, которое сбрасывает пакеты всех или части потоков, обрабатываемых скомпрометированным коммутатором. Под нарушением политики маршрутизации понимается отправка пакетов на порты, не предусмотренные правилами, которые установил контроллер.

Для обнаружения вредоносного сброса пакетов система FlowMon анализирует значения различных счетчиков на портах коммутаторов, которые соответствуют числу переданных, полученных и сброшенных пакетов. Система выбирает некоторый коммутатор S_k для проверки, запрашивает статистику с его портов.

Коммутатор S_k считается скомпрометированным, если коэффициент сброса пакетов превышает заранее определенное пороговое значение θ, т. е. выполняется следующая формула:

$$\left| \frac{\sum_{\forall i \in P_k} T_k^i - \sum_{\forall i \in P_k} R_k^i}{\sum_{\forall i \in P_k} T_k^i + \sum_{\forall i \in P_k} R_k^i} \right| > \theta,$$

где T_k^i — число пакетов, переданных с порта i коммутатора S_k, R_k^i — число пакетов, полученных на порту i коммутатора S_k, P_k — множество портов коммутатора S_k.

Также система FlowMon проверяет наличие вредоносного сброса трафика на физических линиях. Система проверяет соответствие числа пакетов, отправленных и полученных на портах, соединенных одной и той же физической линией. Если некоторые пакеты были сброшены на физической линии, то статистика на портах будет отличаться и нижеприведенное отношение не будет выполняться:

$$|T_{ij} + T_{ji} - R_{ij} - R_{ji}| > \alpha|T_{ij} + T_{ji}|,$$

где T_{ij} — число пакетов, переданных с коммутатора S_i на коммутатор S_j, R_{ij} — число пакетов, полученных на коммутаторе S_i с коммутатора S_j, а α — это заранее определенное пороговое значение.

Для обнаружения некорректной коммутации пакетов система FlowMon проверяет сообщения обо всех новых потоках, появившихся в сети. Проверка проводится путем анализа таблицы маршрутизации предыдущего коммутатора, т. е. коммутатора, с которого был получен пакет нового потока. Контроллер проверяет, на какой порт должен был быть отправлен этот пакет. Если такой порт отличается от порта, на котором был получен пакет, то делается вывод, что коммутатор некорректно коммутирует пакеты.

Серьезным недостатком системы FlowMon является то, что при ее построении используется слишком простая модель атакующего, которая не позволяет проконтролировать подделку статистики скомпрометированным коммутатором и которая не учитывает возможность наличия в сети нескольких скомпрометированных коммутаторов.

Из-за того, что система поочередно проверяет различные коммутаторы, она не способна обнаружить кратковременные атаки, которые выполняются на коммутаторе, не участвующем в проверке в данный момент.

Не учитываются также различные способы сокрытия атак. Например, атакующий может целенаправленно коммутировать пакеты на коммутаторы, на которых уже присутствуют вредоносные правила обработки подобных пакетов. Таким образом, статистика об этом пакете никогда не попадет на контроллер. Подобная ситуация может появиться, когда злоумышленник изменяет заголовки новых потоков на заголовки существующих.

Также система не учитывает возможность наличия на коммутаторе легитимных правил, которые сбрасывают трафик. Такими правилами, например, могут быть правила, установленные межсетевым экраном.

Таким образом, система FlowMon не удовлетворяет критериям ***K1***, ***K2***, ***K3*** и ***K5***.

2.4. FDWD

Система FDWD[4] [9] — это система для обнаружения скомпрометированных коммутаторов в ПКС, которая использует тестовые пакеты для проверки того факта, что коммутатор не выполняет установленные на него правила маршрутизации. Применяются два алгоритма обнаружения: Forwarding Detection и Weighting Detection.

Алгоритм Forwarding Detection использует тестовые пакеты для проверки корректности выполнения правил маршрутизации на коммутаторах. Алгоритм работает следующим образом. Случайно выбирается коммутатор S, и на нем случайно выбирается правило маршрутизации f с полем match M. Далее алгоритм генерирует новые правила маршрутизации f' с полем match $M' = M \cup M_t$, где M_t — поле match, не пересекающееся с полем M, такое что в сети нет пакетов с заголовками, удовлетворяющими этому полю. Эти правила маршрутизации устанавливаются на коммутаторы, непосредственно соединенные с коммутатором S.

После этого алгоритм генерирует тестовый пакет p, заголовок которого удовлетворяет полю M', и отправляет его на коммутатор S. Теперь контроллер может проверить, на какой коммутатор будет отправлен пакет p. Если правило f должно отправлять пакеты на коммутатор S', то при корректном поведении коммутатора S пакет p должен быть обработан правилом f' на коммутаторе S'. В противном случае коммутатор S будет считаться скомпрометированным. Также некорректным поведением будет считаться изменение пакета p.

Алгоритм Weighting Detection проверяет специальные групповые правила g протокола OpenFlow, которые отправляют пакеты на разные порты в заранее заданном соотношении. Проверка заключается в том, что контроллер отправляет на тестируемый коммутатор несколько специальных пакетов для того, чтобы затем проверить, что пакеты были отправлены на порты в правильном соотношении, заданном правилом g. Проверка происходит так же, как и

[4] Forwarding Detection and Weighting Detection.

в алгоритме Forwarding Detection — с помощью специальных правил, заранее установленных на соседние с тестируемым коммутаторы.

Недостатком описанной выше системы является тот факт, что она не учитывает возможность наличия в сети нескольких скомпрометированных коммутаторов, которые могут кооперироваться для того, чтобы избежать обнаружения. Например, если несколько соседних коммутаторов скомпрометированы, то они могут передавать контроллеру неверные сведения о прохождении тестовых пакетов в сети, таким образом скрывая наличие атаки.

Так как система проверяет одно правило за раз, то она может не обнаружить кратковременные атаки.

Из-за того, что система устанавливает дополнительные правила, она может повлиять на атаку, проводимую в сети. Например, система может установить правило, имеющее больший приоритет, чем у вредоносного правила.

Также система не учитывает наличие в сети сбросов пакетов из-за перегрузок.

Таким образом, система FDWD не удовлетворяет критериям ***K1***, ***K3***, ***K5*** и ***K7***.

2.5. MLPC

Система MLPC[5] [10] также использует тестовые пакеты и анализ сетевой статистики для обнаружения скомпрометированных коммутаторов. В статье [10] предполагается, что скомпрометированные коммутаторы могут выборочно сбрасывать, генерировать, изменять или некорректно коммутировать пакеты. Также предполагается, что в сети могут быть несколько скомпрометированных коммутаторов, которые могут кооперироваться между собой для того, чтобы избежать обнаружения.

Для проверки корректной работы сети система вычисляет минимальный набор тестовых пакетов, обработка которых охватывает все правила на коммутаторах в сети. Для нахождения такого набора пакетов система использует информацию об установленных в сети правилах маршрутизации и строит граф зависимостей правил маршрутизации.

Далее система использует топологическую сортировку [19] и модифицированную версию алгоритма Хопкрофта — Карпа [20] для нахождения минимального покрытия графа путями, причем учитываются только пути, соответствующие реальным маршрутам пакетов в сети. Назовем такой путь реализуемым. Для каждого реализуемого пути генерируется тестовый пакет, который должен пройти по всем правилам маршрутизации всех коммутаторов, соответствующим этому пути. В конце каждого пути устанавливается специальное правило маршрутизации, которое должно вернуть тестовые пакеты на контроллер для анализа. Для того чтобы процедура анализа сети не изменяла логику обработки пакетов, тестовые пакеты помечаются уникальной меткой (например, VLAN-меткой, не используемой в сети).

Если тестовый пакет был сброшен или модифицирован, контроллер уменьшает степень доверия к конкретному пути, соответствующему этому пакету. Когда степень доверия уменьшается ниже допустимого порога, путь помечается как подозрительный. Для определения местонахождения скомпрометированного коммутатора контроллер разбивает подозрительный путь на две части и повторяет процедуру тестирования рекурсивно, пока длина подозрительного пути не будет равна 1.

Для того чтобы обнаружить скомпрометированные коммутаторы S, которые отправляют некорректную информацию, контроллер использует коммутаторы, соединенные с S физическими линиями. Система проверяет правило сохранения потока на входе и выходе коммутатора: число пакетов, отправленных на коммутатор, должно быть равно числу пакетов, полученных от этого коммутатора.

Для обнаружения несоответствий в сетевой статистике контроллер устанавливает на соседние коммутаторы дополнительные правила, которые считают число пакетов, отправленных на тестируемый коммутатор, и число пакетов, полученных от тестируемого коммутатора.

Далее проверяется правило сохранения потока. Если найдено несоответствие, то уровень доверия к коммутатору снижается. Если уровень доверия снизится ниже допустимого порога, то коммутатор считается скомпрометированным.

Важно отметить, что подобная проверка статистики может быть неспособна обнаружить точное местоположение скомпрометированного коммутатора. Например, скомпрометированный коммутатор может предоставлять контроллеру специально сформированную сетевую статистику так, чтобы снижать уровень доверия системы к некоторому легитимному коммутатору. Также в ситуации, когда легитимный коммутатор окружен скомпрометированными, его уровень доверия может снижаться быстрее, чем у граничащих с ним коммутаторов.

Система не учитывает случай, когда скомпрометированные коммутаторы могут быть со-

[5] Minimal Legal Path Cover.

седними, т. е. соединены физической линией. В таком случае один из них может скрывать от контроллера информацию о потоке, выходящем с другого коммутатора. Также система не отличает легитимный сброс пакетов от вредоносного и не рассматривает агрегацию потоков, балансировку нагрузки и другие сложные механизмы маршрутизации.

Из-за того, что система проводит выборочные проверки правил маршрутизации с помощью тестовых пакетов и анализа сетевой статистики, она может пропустить кратковременные атаки.

Также из-за установки правил система влияет на состояние сети и, как следствие, влияет на проводимые в сети атаки.

Таким образом, система MLPC не удовлетворяет критериям ***K1***, ***K2***, ***K3***, ***K5***, ***K6*** и ***K7***.

2.6. PDMD

Система PDMD[6] [11] — это система обнаружения скомпрометированных коммутаторов путем проверки маршрутов, пройденных пакетами в сети.

Для выявления реальных маршрутов, пройденных пакетами, система устанавливает в сети правила маршрутизации, которые дублируют часть пакетов на контроллер. Выбор пакетов для отправки на контроллер проводится с помощью алгоритма Trajectory Sampling [21]. Этот алгоритм выбирает пакеты на основе решающей функции от метки пакета. Под меткой понимается набор полей заголовка пакета, которые уникально идентифицируют пакет в сети. Авторы статьи предполагают, что в сети не установлены правила маршрутизации, которые изменяют заголовки пакетов. Иными словами, значение решающей функции от одного и того же пакета одинаково на любом участке сети. Следовательно, выбранный для анализа пакет будет отправляться на контроллер на всех коммутаторах, на которых он был обработан.

Таким образом, система может получать из сети информацию о реальных маршрутах, пройденных пакетами. Далее, на основе информации об установленных в сети правилах маршрутизации система вычисляет теоретический маршрут, который каждый такой пакет должен был пройти в сети. Теоретический маршрут вычисляют по алгоритму достижимости из модели Header Space Analysis [22].

Далее система может сравнить реальный маршрут, пройденный выбранным пакетом, с теоретическим маршрутом. Если маршруты не совпадают, то делается вывод о наличии в сети скомпрометированного коммутатора.

Недостатком системы является тот факт, что при анализе маршрутов не учитывается, что в сети может находиться несколько скомпрометированных коммутаторов, и они могут отправлять на контроллер некорректную информацию о маршрутах пакетов, влияя на процедуру обнаружения. Например, такие коммутаторы могут не отправлять пакеты на контроллер, тем самым скрывая факт атаки. Также они могут изменять заголовки тестовых пакетов для того, чтобы в дальнейшем эти пакеты не были отправлены на контроллер другими легитимными коммутаторами.

В статье [11] предполагается, что в сети не может быть перегрузок, и все сброшенные пакеты были сброшены из-за атаки, а не из-за перегрузки. Также предполагается, что задержка обработки пакетов на коммутаторах и контроллере незначительна.

Из-за того, что алгоритм Trajectory Sampling [21] может не выбрать для анализа пакеты, относящиеся к некоторой проводимой в сети атаке, то эта атака не будет обнаружена. Такое может произойти, например, в случае наличия в сети кратковременной атаки.

Также необходимо отметить, что система рассматривает только простую логику маршрутизации — отдельные правила для различных потоков.

Из сказанного следует, что система PDMD не удовлетворяет критериям ***K1***, ***K2***, ***K3***, ***K5*** и ***K6***.

2.7. SPHINX

Система SPHINX [12] — это система обнаружения вторжений для ПКС. Она обнаруживает различные атаки, направленные на контур управления сетью, и проверяет соблюдение политики безопасности. Также эта система позволяет обнаруживать скомпрометированные коммутаторы, которые проводят различные атаки на контур данных.

Для обнаружения скомпрометированных коммутаторов система SPHINX анализирует устанавливаемые в сеть правила и строит модель сети — потоковый граф. Этот граф описывает маршруты различных потоков в сети. Потоковый граф используется для анализа сетевой статистики и обнаружения аномалий, таких как нелегитимный сброс пакетов.

Анализ сетевой статистики происходит следующим образом: для каждого потока в сети система запрашивает значения счетчиков правил, входящих в маршрут этого потока. Если

[6] Packet Drops and Misroutings Detector.

счетчики правил одного и того же потока отличаются больше, чем на заранее определенное пороговое значение, то система SPHINX сообщает о наличии в сети скомпрометированного коммутатора.

Недостатком системы SPHINX является то, что она не учитывает возможность использования в сети сложных механизмов маршрутизации, таких как групповая маршрутизация, агрегация потоков и балансировка нагрузки. Все потоки представляются маршрутами в сети, для которых установлены уникальные правила маршрутизации. Также не учитывается информация о пакетах, сброшенных из-за перегрузок в сети.

Таким образом, система SPHINX не удовлетворяет критериям ***K3*** и ***K6***.

2.8. RDDF

Система RDDF[7] [13] — это система обнаружения скомпрометированных коммутаторов, использующая тестовые пакеты для проверки работы правил маршрутизации в сети. В этой системе использован алгоритм оптимизации процедуры обнаружения, в котором один тестовый пакет может проверить более одного коммутатора. Оптимизация достигается за счет агрегации правил маршрутизации с совпадающими или непересекающимися полями match.

Как и система ATPG, эта система генерирует тестовые пакеты на основе информации о правилах маршрутизации. Для каждого тестового пакета эта система вычисляет маршрут, который этот пакет должен пройти, и сравнивает его с реальным маршрутом, пройденным пакетом.

Процедура обнаружения оптимизируется за счет объединения нескольких тестовых пакетов в один. Маршруты для тестовых пакетов представляются так называемыми деревьями агрегации (*aggregation trees*). Дерево агрегации — это дерево, в котором вершины соответствуют коммутаторам, а каждый путь от корня к листу описывает путь для тестового пакета в сети.

Дерево агрегации содержит:

- начальный коммутатор — коммутатор, который является стартовой точкой движения тестовых пакетов;
- промежуточные коммутаторы — коммутаторы, которые дублируют тестовые пакеты на различные порты;
- листовые коммутаторы — коммутаторы, отправляющие тестовые пакеты на контроллер.

Для каждого дерева агрегации в сети устанавливают правила маршрутизации из так называемой группы агрегации. Группой агрегации называется множество правил маршрутизации, которые удовлетворяют условиям:

- правила маршрутизации из одной и той же группы на различных коммутаторах имеют либо совпадающие, либо непересекающиеся поля *match*;
- тестовый пакет не должен проходить через один и тот же коммутатор дважды;
- правило маршрутизации *A* может принадлежать нескольким группам агрегации только в случае, если существует правило маршрутизации *B* на другом коммутаторе такое, что правила *A* и *B* имеют совпадающие поля *match*.

Для построения деревьев агрегации необходимо найти минимальное число групп агрегации при условии, что все правила маршрутизации в сети принадлежат хотя бы одной группе агрегации. Эта задача эквивалентна задаче нахождения минимального вершинного покрытия для ориентированного графа, которая, как доказано в работе [13], является NP-трудной.

Каждый тестовый пакет создается так, чтобы он мог быть обработан всеми правилами маршрутизации только из одной и той же группы агрегации. Такой тестовый пакет движется по дереву маршрутов, дублируется в промежуточных вершинах и отправляется на контроллер в листовых вершинах дерева. Если в сети нет скрытых правил, то все тестовые пакеты должны быть отправлены на контроллер из листовых вершин. Таким образом, контроллер может проверить, что каждый коммутатор выполняет все правила маршрутизации.

Эта система обладает тем же недостатком, что и любая система, использующая тестовые пакеты — система не способна отличить вредоносный сброс пакетов, появившийся вследствие атаки на сеть, от легитимного сброса из-за перегрузок в сети и не способна обнаружить кратковременные атаки.

Таким образом, система RDDF не удовлетворяет критериям ***K3*** и ***K5***.

2.9. WedgeTail

Система WedgeTail [14] анализирует маршруты, пройденные пакетами в сети, для обнаружения скрытых правил маршрутизации. Эта система перехватывает OpenFlow-сообщения, передаваемые между контроллером и коммутаторами, и строит у себя виртуальную копию сети, которая затем используется для вычисления теоретических маршрутов движе-

[7] Rapid Detection of Disobedient Forwarding.

ния пакетов в сети. Теоретические маршруты вычисляются с помощью алгоритма вычисления достижимости из модели Header Space Analysis [22].

Для определения реальных маршрутов, пройденных пакетами, система использует систему NetSight [23]. NetSight — это решение для обнаружения аномалий в сети, которое позволяет ПКС-приложению получать информацию о пройденных пакетом коммутаторах. NetSight требует модификации контура данных для того, чтобы генерировать отчеты о пройденных пакетом коммутаторах.

После сбора информации о реальных маршрутах, пройденных пакетами, система WedgeTail сравнивает эти маршруты с теоретическими. Если реальные маршруты пакетов не являются подмножеством теоретических маршрутов, то некоторые коммутаторы на пути пакета являются скомпрометированными.

Для ускорения обнаружения скомпрометированных коммутаторов система WedgeTail упорядочивает коммутаторы на основе приоритета. Приоритет вычисляется на основании трафика, обрабатываемого коммутатором. Основная идея заключается в том, что проверка должна проводиться в первую очередь на тех коммутаторах, которые обрабатывают большее число пакетов. Для определения таких коммутаторов WedgeTail отслеживает траектории всех пакетов на всех портах и определяет наиболее часто задействованные коммутаторы.

Одним из недостатков системы WedgeTail является то, что для ее использования необходимо изменять логику работы коммутатора, потому что эта система использует NetSight для сбора информации о маршрутах, пройденных пакетами. Таким образом, ограничивается область применения этой системы в существующих сетях.

Система WedgeTail может не обнаружить кратковременные атаки из-за того, что проверка проводится поочередно на различных коммутаторах в сети. Также система WedgeTail не разграничивает легитимный и вредоносный сброс трафика.

Таким образом, система WedgeTail не удовлетворяет критериям ***K3***, ***K4*** и ***K5***.

2.10. DYNAPFV

DYNAPFV [15] — это система для проверки корректности обработки пакетов на коммутаторах. Первый коммутатор, обрабатывающий пакет, называется входным для этого пакета. Выходным коммутатором для пакета, называется коммутатор, который отправил этот пакет во вне сети, например, на некоторый хост. Проверка происходит с помощью сравнения числа пакетов, полученных на входных и выходных коммутаторах.

Система устанавливает на граничные коммутаторы правила маршрутизации, которые с помощью *packet-in*-сообщений отправляют пакеты на контроллер для анализа. Система выбирает случайные промежутки времени для сбора пакетов из различных потоков, т. е. на контроллер отправляется только часть пакетов, проходящих в сети.

Каждый пакет, пришедший на контроллер в *packet-in*-сообщении, сохраняется до того момента, пока этот пакет не придет в другом сообщении с выходного коммутатора, либо когда истечет время, равное максимальному времени приема-передачи (RTT) в сети. Таким образом, происходит проверка того, что пакеты не были модифицированы в процессе их обработки промежуточными коммутаторами.

Если отношение числа корректно обработанных пакетов (которые пришли и со входного и с выходного коммутаторов) к числу всех пакетов, отправленных на контроллер, превышает заранее заданное пороговое значение, то система сообщает об ошибке, и начинается поиск скомпрометированных коммутаторов.

Во время поиска скомпрометированных коммутаторов выбирается набор промежуточных коммутаторов между входным и выходным коммутаторами, на которых удаляются соответствующие потоковые правила так, чтобы эти коммутаторы начинали отправлять пакеты на контроллер.

Также система проверяет корректность обработки пакетов в случае, когда таймауты потоковых правил истекают. Система извлекает соответствующие значения сетевой статистики и проверяет корректность обработки пакетов, сравнивая значения статистики с разных коммутаторов. Если отношение числа пакетов, обработанных на некотором коммутаторе S, к числу пакетов, обработанных входным коммутатором, меньше заранее определенного порогового значения, то коммутатор, предшествующий S, считается скомпрометированным.

Для снижения нагрузки на управляющий канал в системе используется динамическая настройка вероятности сбора пакетов для различных потоков.

Недостатком системы DYNAPFV является тот факт, что рассматривается только одна процедура маршрутизации — динамическая установка правил на коммутатор в случае появления пакета, для которого нет правила об-

работки. Кроме того, не учитываются пакеты, сброшенные из-за перегрузки сети, т. е. наличие легитимно сброшенного трафика будет интерпретировано системой как атака на сеть.

Из-за того, что система DYNAPFV запрашивает большое число пакетов из сети, она может негативно повлиять на производительность контроллера и на загрузку управляющего канала.

Система DYNAPFV не обнаруживает атаки, когда в сети есть несколько скомпрометированных коммутаторов, которые передают на контроллер некорректные данные о пакетах, проходящих в сети.

Таким образом, система DYNAPFV не удовлетворяет критериям ***K1***, ***K2***, ***K3*** и ***K6***.

3. Сравнительный анализ

В таблице представлены сводные результаты анализа и сравнение систем обнаружения скомпрометированных ПКС коммутаторов, приведенные выше.

Сравнение систем обнаружения скомпрометированных коммутаторов

	K1	K2	K3	K4	K5	K6	K7
ATPG	✓	✓		✓		✓	✓
FADE		✓		✓	✓		
FlowMon				✓		✓	✓
FDWD		✓		✓		✓	
MLPC				✓			
PDMD				✓			✓
SPHINX	✓	✓		✓	✓		✓
RDDF	✓	✓		✓		✓	✓
WedgeTail	✓	✓				✓	✓
DYNAPFV				✓	✓		✓

Из этой таблицы хорошо видно, что существующие системы обнаружения скомпрометированных коммутаторов в ПКС используют три основных механизма обнаружения:

- анализ значений счетчиков правил маршрутизации;
- генерацию тестовых пакетов и проверку маршрутов, которые они проходят;
- сбор существующих пакетов и анализ их маршрутов.

Одним из недостатков существующих систем обнаружения скомпрометированных коммутаторов заключается в том, что подобные системы рассматривают только самый простой механизм обработки пакетов в сети — динамическую установку отдельных правил маршрутизации для каждого потока в сети. Не учитывается групповая маршрутизация, агрегация правил и балансировка нагрузки. Таким образом, не учитывается тот факт, что ПКС позволяет реализовывать различные сложные механизмы маршрутизации и оптимизации потоков в сети. Сложные механизмы маршрутизации могут привести к очень непростым комбинациям потоковых правил, установленных на коммутаторах. Поэтому необходим алгоритм обнаружения скомпрометированных коммутаторов, который сможет работать при произвольной комбинации потоковых правил в сети.

Из таблицы также видно, что ни одна из существующих систем не способна отличить вредоносный сброс пакетов от легитимного сброса из-за перегрузок в сети. Поскольку перегрузки в сети — достаточно частое явление, то невозможность выявлять причину сброса пакета будет вызывать большое число ошибок второго рода.

Системы, использующие тестовые пакеты или сбор существующих пакетов, не могут различать типы сброса трафика из-за того, что протокол OpenFlow не предполагает отправки коммутатором на контроллер уведомления о каждом сброшенном пакете. Даже учитывая тот факт, что информация о числе сброшенных пакетов записана в статистике порта, у контроллера нет информации о том, что был сброшен именно тестовый пакет.

Также существующие системы, которые используют анализ сетевой статистики, не учитывают информацию о числе сброшенных пакетов, которая хранится в счетчиках портов.

Еще одной проблемой существующих систем, которые анализируют сетевую статистику, является уязвимость к сокрытию факта атаки, т. е. злоумышленник, сбрасывая некоторое число пакетов, может при этом генерировать такое же число новых пакетов для того, чтобы правило сохранения потока на коммутаторе выполнялось.

Кроме того, системы, применяющие анализ сетевой статистики, используют статистику только с граничных коммутаторов по отношению к проверяемому. Таким образом, если атакующий скомпрометировал несколько соседних коммутаторов, у него появляется возможность отправлять контроллеру некорректную сетевую статистику так, что атака не будет обнаружена.

Заключение

В данной статье проведен обзор и сравнение существующих систем обнаружения скомпрометированных ПКС коммутаторов. Сравнение проводилось по определенным критериям, которые охватывают различные аспекты систем обнаружения. Также в обзоре были выделены основные недостатки существующих систем. Необходимо отметить, что в обзоре рассмотрены только системы обнаружения скомпрометированных коммутаторов, которые уже были скомпрометированы и использованы для различных атак на контур данных. Системы обнаружения процесса компрометации не рассмотрены, так как эта область информационной безопасности уже является хорошо исследованной [16].

Из проведенного обзора средств обнаружения скомпрометированных коммутаторов в ПКС следует, что для надежного выявления скомпрометированных коммутаторов необходимо разработать систему, которая будет работать при условии использования в сети различных сложных механизмов маршрутизации и оставаться прозрачной для атакующего. Для этого система должна не зависеть от внутренней логики работы контроллера. Система, удовлетворяющая этим требованиям, должна строить представление сети на основе *OpenFlow* сообщений, отправленных контроллером на коммутаторы.

Важно, чтобы система обнаружения учитывала информацию о пакетах, сброшенных вследствие перегрузки в сети. Это необходимо для того, чтобы избежать ошибок второго рода во время перегрузок.

Список литературы

1. **Kreutz D., Ramos F. M., Verissimo P. E., Rothenberg C. E., Azodolmolky S., Uhlig S.** Software-defined networking: A comprehensive survey // Proceedings of the IEEE. 2015. Vol. 103, N. 1. P. 14—76.

2. **McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Turner J.** OpenFlow: enabling innovation in campus networks // ACM SIGCOMM Computer Communication Review. 2008. Vol. 38, N. 2. P. 69—74.

3. **Петров И. С.** Задача обнаружения скомпрометированных коммутаторов в SDN сетях // REDS: Телекоммуникационные устройства и системы. 2017. Т. 7, № 4. С. 515—518.

4. **Scott-Hayward S., O'Callaghan G., Sezer S.** SDN security: A survey // Future Networks and Services (SDN4FNS), 2013 IEEE SDN For. IEEE, 2013. P. 1—7.

5. **Шендяпин А. С., Петров И. С.** Исследование методов проведения атаки Man-in-the-Middle в программно-конфигурируемых сетях // Программные системы и инструменты. Тематический сборник / Под общей редакцией Р. Л. Смелянского. 2017. С. 73—82.

6. **Zeng H., Kazemian P., Varghese G., McKeown N.** Automatic test packet generation // Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012. P. 241—252.

7. **Pang C., Jiang Y., Li Q.** FADE: Detecting forwarding anomaly in software-defined networks // Communications (ICC), 2016 IEEE International Conference on. IEEE, 2016. P. 1—6.

8. **Kamisiński A., Fung C.** Flowmon: Detecting malicious switches in software-defined networks // Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense. ACM, 2015. P. 39—45.

9. **Chi P. W., Kuo C. T., Guo J. W., Lei C. L.** How to detect a compromised sdn switch // Network Softwarization (NetSoft). 2015 1st IEEE Conference on. IEEE, 2015. P. 1—6.

10. **Chao T. W., Ke Y. M., Chen B. H., Chen J. L., Hsieh C. J., Lee S. C., Hsiao H. C.** Securing data planes in software-defined networks // NetSoft Conference and Workshops (NetSoft), 2016 IEEE. IEEE, 2016. P. 465—470.

11. **Mohammadi A. A., Kazemian P., Pakravan M. R.** Detecting malicious packet drops and misroutings using Header Space Analysis // Telecommunications (IST), 2016 8th International Symposium on. IEEE, 2016. P. 521—526.

12. **Dhawan M., Poddar R., Mahajan K., Mann V.** SPHINX: Detecting Security Attacks in Software-Defined Networks // NDSS. 2015. Vol. 15. P. 8—11.

13. **Chiu Y. C., Lin P. C.** Rapid detection of disobedient forwarding on compromised OpenFlow switches // Computing, Networking and Communications (ICNC), 2017 International Conference on. IEEE, 2017. P. 672—677.

14. **Shaghaghi A., Kaafar M. A., Jha S.** Wedgetail: An intrusion prevention system for the data plane of software defined networks // Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 2017. P. 849—861.

15. **Li Q., Zou X., Huang Q., Zheng J., Lee P. P.** Dynamic Packet Forwarding Verification in SDN // IEEE Transactions on Dependable and Secure Computing. 2018.

16. **Intrusion** detection systems: A survey and taxonomy. Technical report, 2000. Vol. 99.

17. **Петров И. С., Смелянский Р. Л.** Минимизация группового трафика и обеспечение его отказоустойчивости в программно-конфигурируемых сетях // Известия Российской академии наук. Теория и системы управления. 2018. № 3. С. 64—75.

18. **Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.** Introduction to algorithms. Cambridge: MIT press, 2001. Vol. 6.

19. **Dilworth R. P.** A decomposition theorem for partially ordered sets // Annals of Mathematics. 1950. P. 161—166.

20. **Hopcroft J. E., Karp R. M.** An n^5/2 algorithm for maximum matchings in bipartite graphs // SIAM Journal on computing. 1973. Vol. 2, N. 4. P. 225—231.

21. **Duffield N. G., Grossglauser M.** Trajectory sampling for direct traffic observation // IEEE/ACM Transactions on Networking (ToN). 2001. Vol. 9, N. 3. P. 280—292.

22. **Kazemian P., Varghese G., McKeown N.** Header Space Analysis: Static Checking for Networks // NSDI. 2012. Vol. 12. P. 113—126.

23. **Handigol N., Heller B., Jeyakumar V., Mazières D., McKeown N.** I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks // NSDI. 2014. Vol. 14. P. 71—85.

I. S. Petrov, PhD Student, e-mail: ipetrov@cs.msu.ru,
Lomonosov Moscow State University, Moscow, Russian Federation

Systems for Compromised Switch Detection in Software-Defined Networking

Compromised network switches present a serious threat for the security of the network since they can be used by an attacker to perform a wide range of network attacks. This paper presents a survey and comparison of existing systems for compromised switch detection in software-defined networking, shows their main advantages and disadvantages.

Keywords: *software-defined networking, intrusion detection systems, compromised network switches*

DOI: 10.17587/it.25.131-142

References

1. **Kreutz D., Ramos F. M., Verissimo P. E., Rothenberg C. E., Azodolmolky S., Uhlig S.** Software-defined networking: A comprehensive survey, *Proceedings of the IEEE*, 2015, vol. 103, no. 1, pp. 14—76.

2. **McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Turner J.** OpenFlow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, 2008, vol. 38, no. 2, pp. 69—74.

3. **Petrov I. S.** *Zadacha obnaruzhenija skomprometirovannyh kommutatorov v SDN setjah* (The problem of detecting compromised switches in SDN), *REDS: Telekom Systems and Infrustructure*, 2017, vol. 7, no. 4, pp. 515—518 (in Russian).

4. **Scott-Hayward S., O'Callaghan G., Sezer S.** SDN security: A survey, *Future Networks and Services (SDN4FNS),* 2013 IEEE SDN For, IEEE, 2013, pp. 1—7.

5. **Shendiapin A. S., Petrov I. S.** *Issledovanie metodov provedenija ataki Man-in-the-Middle v programmno-konfiguriruemyh setjah* (Research of the Man-in-the-Middle in Software-Defined Netrowks), *Journal of Program Systems and Instruments*, Moscow, 2017, pp. 73—8 (in Russian).

6. **Zeng H., Kazemian P., Varghese G., McKeown N.** Automatic test packet generation, *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, ACM, 2012, pp. 241—252.

7. **Pang C., Jiang Y., Li Q.** FADE: Detecting forwarding anomaly in software-defined networks, *Communications (ICC), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1—6.

8. **Kamisiński A., Fung C.** Flowmon: Detecting malicious switches in software-defined networks, *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, ACM, 2015, pp. 39—45.

9. **Chi P. W., Kuo C. T., Guo J. W., Lei C. L.** How to detect a compromised sdn switch, *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, IEEE, 2015, pp. 1—6.

10. **Chao T. W., Ke Y. M., Chen B. H., Chen J. L., Hsieh C. J., Lee S. C., Hsiao H. C.** Securing data planes in software-defined networks, *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, IEEE, 2016, pp. 465—470.

11. **Mohammadi A. A., Kazemian P., Pakravan M. R.** Detecting malicious packet drops and misroutings using Header Space Analysis, *Telecommunications (IST), 2016 8th International Symposium on*, IEEE, 2016, pp. 521—526.

12. **Dhawan M., Poddar R., Mahajan K., Mann V.** SPHINX: Detecting Security Attacks in Software-Defined Networks, *NDSS*, 2015, vol. 15, pp. 8—11.

13. **Chiu Y. C., Lin P. C.** Rapid detection of disobedient forwarding on compromised OpenFlow switches, *Computing, Networking and Communications (ICNC), 2017 International Conference on*, IEEE, 2017, pp. 672—677.

14. **Shaghaghi A., Kaafar M. A., Jha S.** Wedgetail: An intrusion prevention system for the data plane of software defined networks, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ACM, 2017, C. 849—861.

15. **Li Q., Zou X., Huang Q., Zheng J., Lee P. P.** Dynamic Packet Forwarding Verification in SDN, *IEEE Transactions on Dependable and Secure Computing*, 2018.

16. **Axelsson S.** Intrusion detection systems: A survey and taxonomy, *Technical report*, 2000, vol. 99.

17. **Petrov I. S., Smeliansky R. L.** *Minimizacija gruppovogo trafika i obespechenie ego otkazoustojchivosti v programmno-konfiguriruemyh setjah* (Minimization of Multicast Traffic and Ensuring Its Fault Tolerance in Software-Defined Networks), *Journal of Computer and Systems Sciences International*, 2018, vol. 57, no. 3, pp. 407—419 (in Russian).

18. **Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.** Introduction to algorithms, Cambridge, MIT press, 2001, vol. 6.

19. **Dilworth R. P.** A decomposition theorem for partially ordered sets, *Annals of Mathematics*, 1950, pp. 161—166.

20. **Hopcroft J. E., Karp R. M.** An n^5/2 algorithm for maximum matchings in bipartite graphs, *SIAM Journal on computing*, 1973, vol. 2, no. 4, pp. 225—231.

21. **Duffield N. G., Grossglauser M.** Trajectory sampling for direct traffic observation, *IEEE/ACM Transactions on Networking (ToN),* 2001, vol. 9, no. 3, pp. 280—292.

22. **Kazemian P., Varghese G., McKeown N.** Header Space Analysis: Static Checking for Networks, *NSDI*, 2012, vol. 12, pp. 113—126.

23. **Handigol N., Heller B., Jeyakumar V., Mazières D., McKeown N.** I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks, *NSDI*, 2014, vol. 14, pp. 71—85.