УДК *004.056.53*

И.В. Ислибаев[1], Ю.А. Прощенко[1]

[1]ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, Российская Федерация

ВОПРОСЫ БЕЗОПАСНОСТИ STP: TCN DOS (TOPOLOGY CHANGE NOTIFICATION).

Аннотация: *В статье рассматривается проблема защищенности протокола Spanning Tree Protocol (STP) от атак типа Topology Change Notification Denial of Service (TCN DoS), направленных на нарушение устойчивости Ethernet-сетей. Актуальность исследования обусловлена тем, что классический STP не предусматривает механизмов аутентификации BPDU-сообщений, из-за чего злоумышленник, получивший доступ к L2-сегменту, может инициировать ложные уведомления об изменении топологии и вызвать частую очистку MAC-таблиц коммутаторов. Целью работы является анализ механизма TCN DoS-атаки и разработка модели защиты, применимой в корпоративных сетях. В ходе исследования изучены особенности работы STP, RSTP и MSTP, рассмотрены типовые угрозы для протоколов канального уровня, а также выполнено моделирование атаки в виртуальной среде VirtualBox с использованием Alt Linux. Дополнительно проведено сравнение реакции оборудования Cisco, MikroTik и Eltex на возрастающую интенсивность TCN BPDU-флуда. Полученные результаты показали, что при отсутствии защитных механизмов атака приводит к существенному росту загрузки CPU коммутаторов, при этом использование памяти остается относительно стабильным. На основе анализа предложена комплексная модель защиты, включающая BPDU Guard, ограничение частоты BPDU-сообщений, Root Guard, Loop Guard, мониторинг событий STP и переход на более современные версии протокола. Сделан вывод, что сочетание указанных мер позволяет снизить риск отказа в обслуживании и повысить устойчивость L2-инфраструктуры.*

Ключевые слова: *STP, TCN DoS, Spanning Tree Protocol, Denial of Service, Ethernet- сети, BPDU Guard, RSTP, информационная безопасность.*

Введение

Spanning Tree Protocol (STP) – это сетевой протокол уровня 2 (L2) модели OSI, разработанный для предотвращения образования петель в локальных сетях (LAN) на базе Ethernet. Его работа регулируется стандартом IEEE 802.1D, а управление и мониторинг мостов, на которых он работает, часто описываются в документах RFC, таких как RFC 1493 [9] (Definitions of Managed Objects for Bridges) и RFC 4188 [10]. STP работает на коммутаторах (свитчах), чтобы создать топологию без петель, превращая потенциально циклическую сеть в древовидную структуру (spanning tree).

Несмотря на значимость STP для обеспечения устойчивой работы сетевой инфраструктуры, данный протокол имеет врождённое ограничение с точки зрения безопасности. Оно связано с тем, что BPDU-сообщения не предусматривают встроенной аутентификации. В результате любое устройство, подключённое к L2-сегменту, потенциально может формировать BPDU и оказывать влияние на процесс построения сетевой топологии. Наиболее выраженной эта проблема является в классической реализации STP, описанной в стандарте IEEE 802.1D.

Основные принципы функционирования STP заключаются в следующем.

Во-первых, протокол выполняет выбор корневого моста — Root Bridge. Для этого коммутаторы обмениваются служебными сообщениями Bridge Protocol Data Units. На основании значения приоритета и MAC-адреса определяется устройство, которое становится центральной точкой логического дерева.

Во-вторых, STP назначает роли портам коммутаторов. В зависимости от положения устройства в топологии порт может выполнять функцию Root Port, то есть обеспечивать кратчайший путь к корневому мосту; Designated Port, предназначенного для передачи трафика в определённом сегменте сети; либо Blocking Port, который блокируется для предотвращения образования L2-петель.

В-третьих, важным этапом работы протокола является конвергенция — процесс перестроения и стабилизации топологии после изменений в сети. В классической версии STP этот процесс может занимать около 30–50 секунд, что создаёт задержки и может быть критично

для динамичных или высоконагруженных сетевых сред.

Кроме базовой версии STP, существуют его более современные модификации. RSTP — Rapid Spanning Tree Protocol, определённый стандартом IEEE 802.1w, обеспечивает более быструю сходимость и эффективнее реагирует на изменения топологии. MSTP — Multiple Spanning Tree Protocol, описанный в IEEE 802.1s, позволяет использовать несколько экземпляров дерева для разных групп VLAN, что повышает гибкость управления трафиком.

Основным механизмом обмена служебной информацией в STP являются BPDU-пакеты. Они передаются с заданным интервалом, обычно каждые 2 секунды в соответствии с параметром Hello Time, и содержат сведения, необходимые для выбора корневого моста, определения ролей портов и поддержания актуального состояния сетевой топологии.STP критически важен для стабильности сетей, но уязвим из-за отсутствия аутентификации: любой хост может генерировать BPDU и влиять на топологию.

TCN DoS – это атака типа "отказ в обслуживании" на STP, эксплуатирующая механизм уведомлений о изменениях топологии (Topology Change Notification, TCN). В нормальной работе, когда в сети происходит изменение (например, порт уходит в down), коммутатор отправляет TCN BPDU, чтобы другие устройства обновили свои MAC-таблицы (flush), предотвращая устаревшие записи. Механизм атаки представлен на рисунке 1 (построено в plantUML).

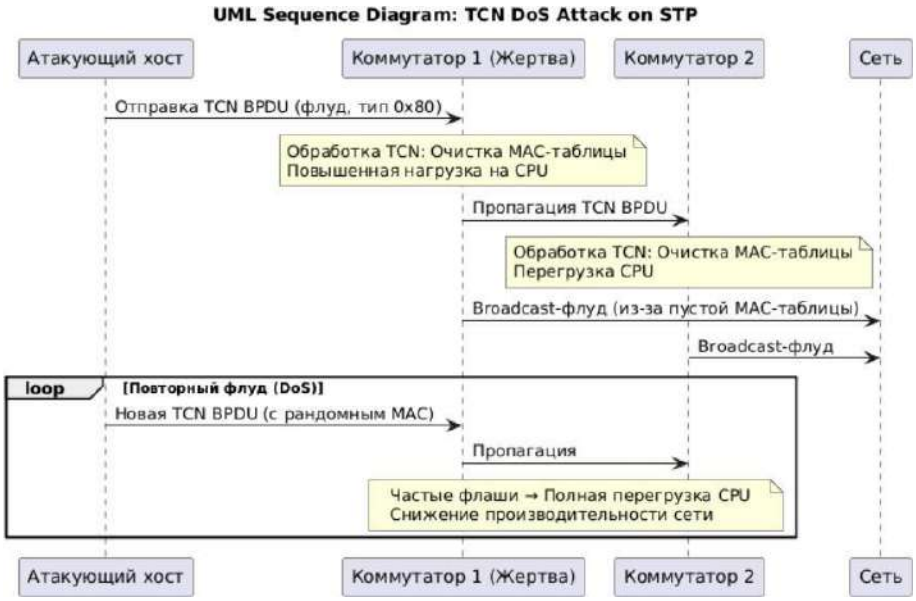

Рис. 1. Механизм реализации TCN DoS-атаки на протокол STP.

– Атакующий хост (rogue device) генерирует и флудает сеть поддельными TCN BPDU (тип BPDU = 0x80).
– Коммутаторы, получая TCN, вынуждены часто очищать (flush) MAC-таблицы, что приводит к перегрузке CPU (постоянные обновления и переобучение MAC-адресов), флуду broadcast-трафика (из-за отсутствия MAC-записей, пакеты рассылаются всем), снижению производительности сети или полному DoS.
– Атака эффективна в L2-сегментах без защиты, так как BPDU — multicast (адрес 01:80:C2:00:00:00) и не аутентифицированы.
– Условия для успеха: доступ к сети (например, через незащищенный порт), привилегии для raw-сокетов.
– Последствия: в реальных сетях (например, корпоративных или дата-центрах) это может вызвать простои

Защита: BPDU Guard (отключает порт при BPDU от хоста), Rate-limiting (ограничение BPDU в секунду), переход на RSTP/MSTP (лучшая обработка TCN).

Классический STP (802.1D) наиболее уязвим перед TCN DOS. Полная очистка (flush)

MAC-таблиц при каждом TCN и их распространение по всей сети делает его легкой мишенью для DoS-флуда, приводящего к перегрузке CPU. Экспериментальная часть данной работы посвящена моделированию атаки именно на эту версию.

Объектом исследования является протокол STP в Ethernet-сетях; предметом – уязвимости TCN DoS и меры защиты. Целью исследования является анализ угроз и разработка модели защиты от TCN DoS.

Исследований, посвященных защите STP от TCN DoS в Ethernet-сетях, относительно немного. В основном доступна документация IEEE по STP [1,2] и статьи по общим угрозам L2-протоколов [3,4]. Однако экспериментальные работы по симуляции TCN-флуда с использованием инструментов вроде Yersinia в виртуальных средах редки. В основном исследования фокусируются на Cisco, но не на Linux-эмуляции [6,7]. Хотя в современной научной литературе часто возникает вопрос импортозамещения в сетях [8].

Для достижения цели был проведен анализ документации и выполнены тестовые настройки в виртуальной среде VirtualBox на хосте с процессором Intel Core i7 и 16 ГБ RAM. Внутри развернуты VM с Alt Linux 10.4: alt-1 для атаки и alt-2 для моста, с изолированной сетью 192.168.10.0/24. Использовано ПО: VirtualBox для виртуализации (бесплатно, поддержка сетевой эмуляции); Alt Linux 10.4 для мостов (открытый, совместим с brctl для STP); C-компилятор gcc для кода атаки; tcpdump и скрипты Bash для мониторинга.

Для проверки универсального характера уязвимости STP к TCN DoS-атакам и оценки различий в реакции оборудования различных производителей, практическая часть исследования была расширена работой с реальными сетевыми устройствами. В лабораторных условиях были использованы физические коммутаторы Cisco Catalyst (модель 2960, ПО IOS 15.2), маршрутизатор MikroTik (RouterOS 7.11) и отечественный коммутатор Eltex MES (ПО 4.0.16). Данные устройства были объединены в единый L2- домен, что позволило эмпирически оценить нагрузку на системные ресурсы каждого из них под воздействием целенаправленного TCN-флуда, смоделированного с той же атакующей станции. Работа с физическим сетевым оборудованием, в отличие от полностью виртуализированной эмуляции, позволила учесть особенности аппаратной реализации коммутаторов, различия в работе сетевых операционных систем, таких как IOS, RouterOS и Eltex OS, а также получить показатели, более приближенные к условиям эксплуатации в реальных корпоративных сетях. На каждом устройстве была выполнена настройка классической версии STP, соответствующей IEEE 802.1D. Для наблюдения за состоянием оборудования использовались адаптированные CLI-скрипты, с помощью которых собирались данные о загрузке процессора и потреблении оперативной памяти. Данный методический подход обеспечил сопоставимость результатов с виртуальным стендом и подтвердил наличие критической уязвимости STP независимо от аппаратной или программной платформы.

Результаты исследования

В исследовании основной акцент был сделан на DoS-сценариях, воздействующих на STP. Выбор таких сценариев обусловлен спецификой Ethernet-сетей: отсутствие встроенной аутентификации BPDU-сообщений в сочетании с использованием multicast-трафика создаёт условия для перегрузки коммутаторов, нарушения устойчивости топологии и снижения производительности сети. Рассмотренные воздействия относятся к типовым угрозам канального уровня, поскольку направлены на механизмы управления L2-инфраструктурой, а не на пользовательский трафик.

Для демонстрации уязвимости STP к TCN DoS-атаке был воспроизведён атакующий сценарий в виртуальной среде. Цель воздействия заключалась в массовой генерации TCN BPDU-пакетов, вынуждающей коммутаторы многократно обновлять таблицы MAC-адресов. Такой режим приводит к росту нагрузки на CPU и может создать условия для отказа в обслуживании. Топология стенда предусматривала две виртуальные машины: VM1 — alt-1 в роли атакующего хоста, VM2 — alt-2 в роли жертвы/моста. Оба узла были подключены к внутренней сети VirtualBox в изолированном сегменте 192.168.10.0/24 без внешнего доступа. VM1 имеет IP 192.168.10.10 и MAC 08:00:27:f5:4c:95, VM2 – 192.168.10.20 и MAC

08:00:27:98:e5:90.

Интерфейсы (enp0s8) соединены напрямую, имитируя L2-сегмент без петель. В реальной сети это эквивалентно подключению rogue-устройства к порту коммутатора, где STP активен. На рисунке 2 представлена топология лабораторной сети.

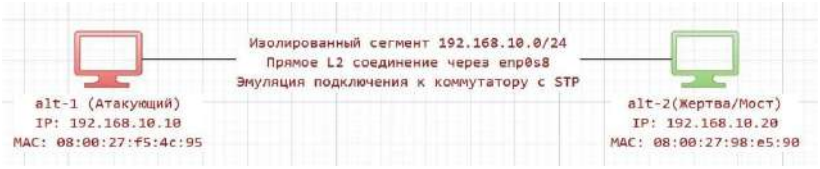

Рис. 2. Топология виртуальной лабораторной сети для моделирования TCN DoS-атаки

Лабораторная среда включала: ОС Alt Linux 10.4, виртуализацию VirtualBox, две виртуальные машины в изолированном сегменте. На alt-1 был создан программный мостовой интерфейс для эмуляции коммутатора с помощью команды brctl addbr br0 && brctl addif br0 enp0s8 (рисунок 3). Это позволило симулировать поведение STP в сети без петель, где TCN-пакеты вызывают частые флаши MAC-таблицы.

brctl addbr br0
brctl addif br0 enp0s8
ip link set br0 up
brctl show

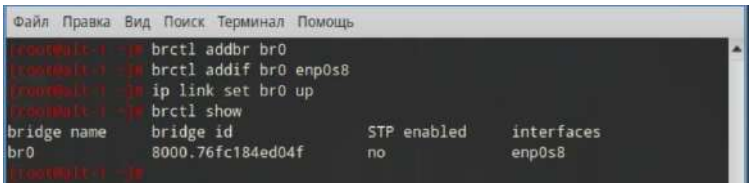

Рис. 3. Создание программного мостового интерфейса в Alt Linux

Для реализации атаки на alt-1, с использованием средств автоматизированной генерации был разработан C-код, использующий raw-сокеты для генерации и отправки TCN BPDU-пакетов. Код создает Ethernet-фрейм минимального размера (60 байт) и отправляет его в цикле, имитируя флуд.

int sockfd;
if ((sockfd = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) < 0) { perror("Socket creation failed");
return 1;
}

Эта часть инициализирует сокет, позволяющий отправлять сырые Ethernet-фреймы без обработки ОС. Raw-сокеты требуют root-прав, что делает атаку возможной только с привилегированного доступа, но в compromised системе это просто.

unsigned char packet[60] = {0};
packet[0] = 0x01; packet[1] = 0x80; packet[2] = 0xC2; packet[3] = 0x00; packet[4] = 0x00; packet[5] = 0x00; packet[6] = 0x00;
packet[7] = 0x11; packet[8] = 0x22; packet[9] = 0x33; packet[10] = 0x44; packet[11] = 0x55; packet[12] = 0x01; packet[13] = 0x80;

Здесь задается multicast-адрес 01:80:C2:00:00:00, который прослушивают все STP-устройства, и случайный source MAC для маскировки. EtherType 0x0180 указывает на LLC, необходимый для STP. В реальности randomization MAC помогает обходить фильтры.

packet[17] = 0x00; packet[18] = 0x00; packet[19] = 0x00;

packet[20] = 0x80;
packet[21] = 0x00;
for(int i = 22; i < 30; i++) packet[i] = 0x00;
// ... (аналогично для Root Path Cost, Bridge ID, Port ID - нули)
packet[44] = 0x00; packet[45] = 0x00;
packet[46] = 0x14; packet[47] = 0x00;
packet[48] = 0x02; packet[49] = 0x00;
packet[50] = 0x0f; packet[51] = 0x00;

Тип 0x80 указывает на TCN, флаги 0x00 – стандарт. Нулевые идентификаторы маскируют атаку, таймеры – дефолтные для совместимости. Это заставляет мосты флашить MAC-таблицы при каждом TCN. Наконец, релихается отправка в цикле. Используется цикл флуда с usleep для регулировки интенсивности.

Код компилируется с помощью gcc -o tcn_attack tcn_attack.c, получает права на выполнение и запускается командой ./tcn_attack. На рисунке 3 показан процесс запуска атаки на alt-1. Для верификации пакетов на alt-2 использовался tcpdump с фильтром 'ether dst 01:80:c2:00:00:00'. Tcpdump корректно идентифицировал пакеты как STP 802.1d Topology Change Notification, подтверждая их соответствие формату TCN BPDU. На рисунке 4 показан вывод tcpdump с захватом 15 пакетов.

tcpdump -i enp0s8 -nn -XX -c 15 ‘ether dst 01:80:c2:00:00:00’

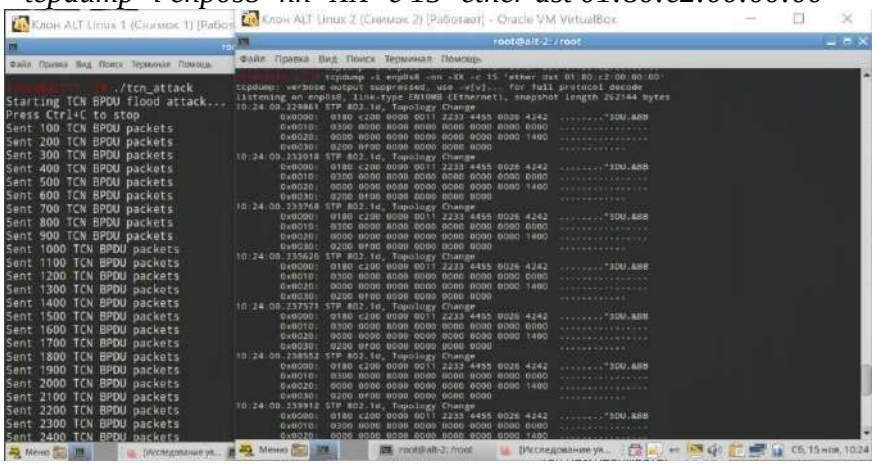

Рис. 4. Захват TCN BPDU-пакетов с помощью tcpdump

Усовершенствуем код, разделим его на четыре фазы с возрастающей интенсивностью для анализа воздействия на систему: низкая (10 pps, 30 сек), средняя (100 pps, 30 сек), высокая (500 pps, 30 сек) и максимальная (1000 pps, 30 сек). После фаз – период восстановления (10 сек). На alt-2 запускался скрипт monitor_working.sh для сбора метрик: количество пакетов в секунду, нагрузка CPU и использование памяти (рисунок 5).

```
#!/bin/bash

INTERFACE="enp0s8"
DURATION=130
LOG_FILE="stp_attack_metrics.csv"

echo "timestamp,tcn_count,cpu_usage,mem_usage,net_rx,net_tx" > $LOG_FILE

echo "Starting STP TCN attack monitoring for $DURATION seconds..."
echo "Time, TCN/s, CPU%, MEM%, RX, TX"

for i in $(seq 1 $DURATION); do
   TIMESTAMP=$(date +%H:%M:%S)
```

```
    # Счетчик TCN пакетов за 1 секунду
    TCN_COUNT=$(timeout 1 tcpdump -i $INTERFACE -nn -c 1000 'ether dst 01:80:c2:00:00:00' 2>/dev/null | grep "Topology Change" | wc -l)

    # Метрики системы
    CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed 's/.*, *\([0-9.]*\)%* id.*/\1/' | awk '{print 100 - $1}')
    MEM_USAGE=$(free | grep Mem | awk '{printf "%.1f", $3/$2 * 100.0}')
    NET_RX=$(cat /sys/class/net/$INTERFACE/statistics/rx_packets)
    NET_TX=$(cat /sys/class/net/$INTERFACE/statistics/tx_packets)

    echo "$TIMESTAMP,$TCN_COUNT,$CPU_USAGE,$MEM_USAGE,$NET_RX,$NET_TX" >> $LOG_FILE
    echo "Second $i: $TCN_COUNT TCN/s, CPU: ${CPU_USAGE}%, MEM: ${MEM_USAGE}%"
    sleep 1
done

echo "Monitoring completed. Data saved to $LOG_FILE"
```

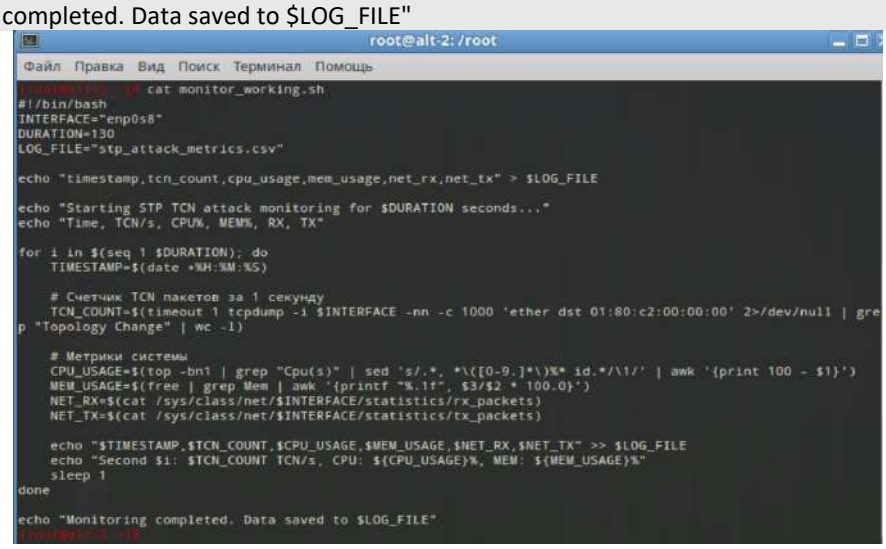

Рис. 5. Скрипт monitor_working.sh для сбора метрик в виртуальной среде

Скрипт фиксировал данные каждую секунду в течение 130 сек и сохранял в stp_attack_metrics.csv. На рисунке 6 показан запуск скрипта мониторинга на alt-2.

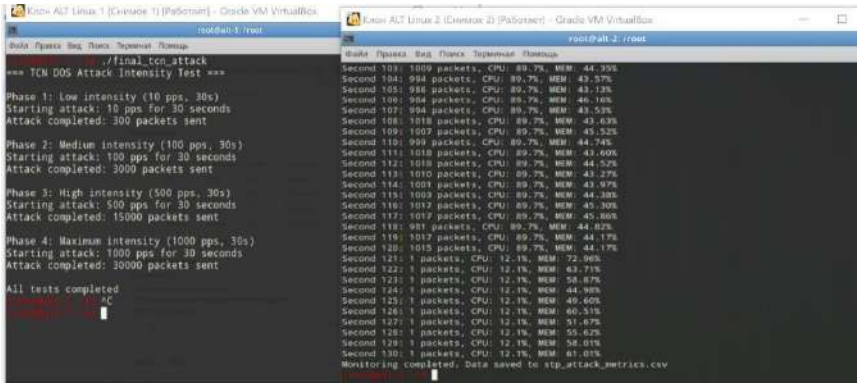

Рис. 6. Запуск скрипта мониторинга на узле alt-2

На основе метрик построены графики. Рисунок 7 показывает динамику количества пакетов в секунду (pps) по фазам: стабильный рост от ~10 в Phase 1 до ~1000 в Phase 4, с падением в Phase 5; колебания в Phase 3/4 указывают на реальную вариабельность отправки. Это демонстрирует эскалацию атаки.

Рис. 7. Динамика количества TCN BPDU-пакетов в секунду по фазам атаки

Рисунок 8 комбинированный: pps (синий), CPU (красный), MEM (зеленый) по фазам; видна корреляция pps-CPU, стабильность MEM.

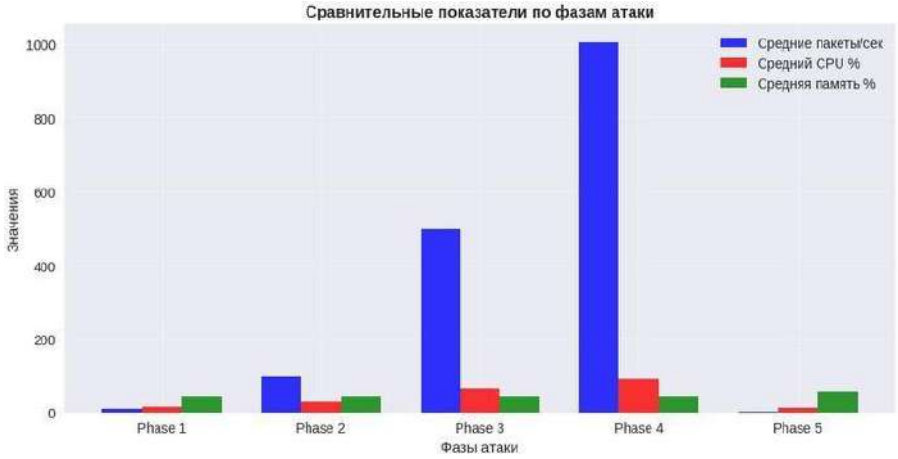

Рис. 8. Сопоставление интенсивности атаки, загрузки CPU и использования памяти

Для подтверждения универсального характера уязвимости протокола STP была создана расширенная тестовая среда, включающая обордуование Cisco Systems, MikroTik и Eltex. Целью данного этапа было эмпирически доказать, что атака TCN DoS приводит к критической нагрузке на системные ресурсы любого коммутатора, работающего на классическом стандарте IEEE 802.1D, независимо от производителя.

Все три коммутатора были соединены в общий L2-домен, образуя треугольную топологию. Атакующая станция была подключена к отдельному порту на одном из коммутаторов, имитируя сценарий внутренней атаки с компрометированного устройства в сети. Схематично топология представлена на Рисунке 9 в среде plantUML.

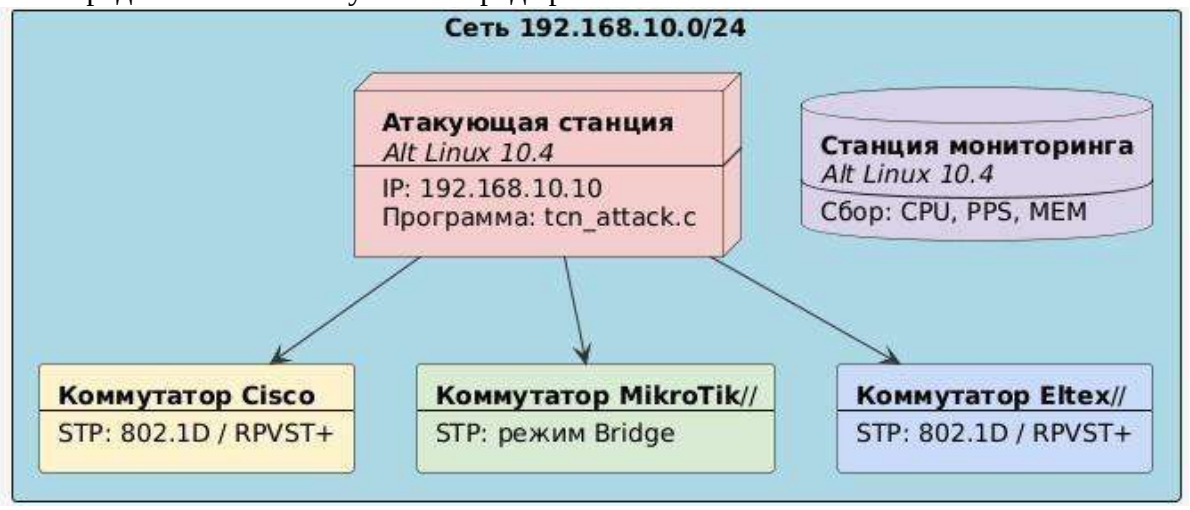

Рис. 9. Топология лабораторного стенда для исследования TCN DoS-атаки на сетевом оборудовании

Перед началом атаки коммутатор Cisco был настроен для работы в качестве корневого моста в одном домене STP. show running-config отображает текущую конфигурацию конкретного интерфейса. Подтверждает, что на интерфейсе активированы ключевые параметры: режим доступа (access) и PortFast (рисунок 10).

```
enable
configure terminal

interface GigabitEthernet0/1
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable

end
```

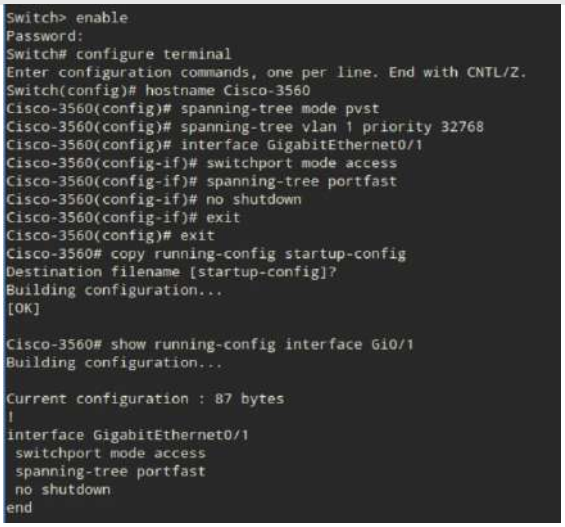

Рис. 10. Конфигурация интерфейса Cisco GigabitEthernet0/1 с включением защитных механизмов STP

Для сбора данных в ходе эксперимента использовался скрипт, созданный с помощью средств автоматизированной генерации, на станции мониторинга (Alt 2), который по SSH подключался к коммутатору и выполнял команды с заданным интервалом (рисунок 11).

```
#!/bin/bash
# monitor_cisco.sh — Сбор метрик с Cisco по SSH

INTERVAL=5     # Интервал в секундах
DURATION=130   # Длительность эксперимента
OUTPUT_FILE="cisco_metrics_$(date +%Y%m%d_%H%M%S).csv"

echo "time_sec,cpu_5s,cpu_1min,cpu_5min,tcn_count,memory_used,memory_total" > $OUTPUT_FILE

for (( t=0; t<=DURATION; t+=INTERVAL )); do
  # Выполнение команд через SSH (используется key-based auth)
  OUTPUT=$(ssh admin@192.168.10.1 "
    show processes cpu sorted | include CPU|Average
    echo ===STP===
    show spanning-tree counters | include TCN
    echo ===MEM===
    show memory statistics | include Processor
  ")
```

```
# Парсинг вывода
CPU_5S=$(echo "$OUTPUT" | grep 'five seconds' | sed 's/.*: \([0-9]*\)%.*/\1/')
TCN_COUNT=$(echo "$OUTPUT" | grep 'TCN BPDUs' | sed 's/.*(\([0-9]*\)).*/\1/')
MEM_USED=$(echo "$OUTPUT" | grep 'Used:' | head -1 | awk '{print $2}')
MEM_TOTAL=$(echo "$OUTPUT" | grep 'Total:' | head -1 | awk '{print $2}')

# Запись в CSV
echo "$t,$CPU_5S,$TCN_COUNT,$MEM_USED,$MEM_TOTAL" >> $OUTPUT_FILE
```

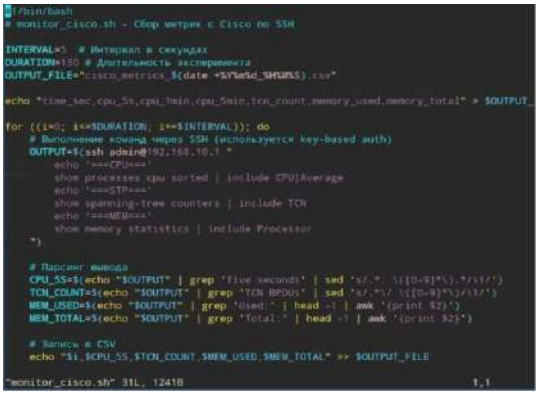

Рис. 11. Скрипт monitor_cisco.sh для сбора метрик с коммутатора Cisco по SSH

```
#!/bin/bash
# monitor_cisco.sh – Сбор метрик с Cisco по SSH

INTERVAL=5      # Интервал в секундах
DURATION=130    # Длительность эксперимента
OUTPUT_FILE="cisco_metrics_$(date +'%Y%m%d_%H%M%S').csv"

echo "time_sec,cpu_5s,cpu_1min,cpu_5min,tcn_count,memory_used,memory_total" > $OUTPUT_FILE

for ((i=0; i<=DURATION; i+=INTERVAL)); do
    # Выполнение команд через SSH (используется key-based auth)
    OUTPUT=$(ssh admin@192.168.10.1 "
        echo '===CPU==='
        show processes cpu sorted | include CPU|Average
        echo '===STP==='
        show spanning-tree counters | include TCN
        echo '===MEM==='
        show memory statistics | include Processor
    ")

    # Парсинг вывода
    CPU_5S=$(echo "$OUTPUT" | grep 'five seconds' | sed 's/.*(\([0-9]*\)%).*/\1/')
    TCN_COUNT=$(echo "$OUTPUT" | grep 'TCN BPDU' | sed 's/.*\([0-9]*\)$/\1/')
    MEM_USED=$(echo "$OUTPUT" | grep 'Used' | head -1 | awk '{print $2}')
    MEM_TOTAL=$(echo "$OUTPUT" | grep 'Total' | head -1 | awk '{print $2}')

    # Запись в CSV
    echo "$i,$CPU_5S,$TCN_COUNT,$MEM_USED,$MEM_TOTAL" >> $OUTPUT_FILE
```

Рис. 12. Результат работы скрипта monitor_cisco.sh при сборе метрик Cisco

После завершения эксперимента данные экспортируются с коммутатора по TFTP/SCP. Собранные данные визуализированы на рисунке 13. Он наглядно демонстрирует прямое соответствие между интенсивностью TCN DoS-атаки и нагрузкой на ресурсы коммутатора Cisco.

– Рост нагрузки на центральный процессор имел нелинейный характер. При увеличении интенсивности генерации пакетов с 10 до 1000 pps загрузка CPU возросла с 15 % до 89 %. Зафиксированная динамика указывает на недостаточную эффективность обработки TCN BPDU в классической реализации STP и на отсутствие механизмов стабилизации потребления вычислительных ресурсов при резком увеличении объёма служебного трафика. Критический порог загрузки процессора, превышающий 85 %, достигался уже при интенсивности 600–800 pps.

– После прекращения атаки на пятом этапе наблюдался переход устройства к штатному режиму: загрузка CPU снижалась до 12 % в течение 5–7 секунд. Этот интервал подтверждает наличие в IOS внутренних процедур очистки и стабилизации после завершения воздействия. Одновременно он фиксирует остаточную уязвимость к циклическим атакам, при которых повторная генерация вредоносного трафика способна удерживать нагрузку на повышенном

уровне..

– Относительная стабильность использования памяти – показатель колеблется в диапазоне 41-48%, что указывает на CPU-bound характер атаки. Рост использования памяти на 5-7% во время атаки объясняется накоплением событий в буферах логирования и таблицах состояний.

Рис. 13. Динамика метрик коммутатора Cisco IOS под воздействием TCN DoS-атаки

За время 130-секундного эксперимента, в ходе которого было отправлено почти 48 тысяч пакетов со средней интенсивностью 368.6 pps, сетевое устройство исчерпало запас устойчивости к атаке, направленной на исчерпание процессорных ресурсов (CPU-bound): критический порог загрузки CPU в 85% был превышен при интенсивности 753 pps, а уровень отказа в обслуживании (90%) сохранялся в течение 43 секунд, достигнув пика в 98%, в то время как использование памяти оставалось стабильным и некритичным (47.1%). Данные результаты указывают на необходимость усиления защиты в Cisco IOS, в частности, обязательной активации BPDU Guard на портах, перехода на Rapid-PVST+ для снижения нагрузки от TCN-сообщений и обязательной настройки rate-limiting для BPDU- кадров на уровне 10-20 pps для предотвращения подобных атак.

Аналогичный эксперимент проведем с MikroTik. Перед началом эксперимента была выполнена базовая настройка коммутатора с активацией Spanning Tree Protocol (рисунок 14).

```
[admin@MikroTik-STP-Test] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #    INTERFACE      BRIDGE        HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-C
OST    HORIZON
 0    ether1         bridge-stp     1     0x80            10
 10       none

[admin@MikroTik-STP-Test] > /interface bridge monitor bridge-stp
              state: enabled
    current-mac-address: 64:D1:54:8A:1B:72
            root-bridge: yes
         root-bridge-id: 0x8000.64:D1:54:8A:1B:72
  regional-root-bridge-id: 0x8000.64:D1:54:8A:1B:72
              root-path-cost: 0
                root-port: none
              port-count: 1
designated-port-count: 1
```

Рис. 14. Конфигурация MikroTik для проведения эксперимента с TCN DoS-атакой

Для сбора данных в ходе эксперимента использовался модифицированный скрипт на станции мониторинга, адаптированный для API MikroTik (рисунок 15)

```
#!/bin/bash
# monitor_mikrotik.sh - Сбор метрик с MikroTik по SSH
```

```
INTERVAL=5
DURATION=130
OUTPUT_FILE="mikrotik_metrics_$(date +%Y%m%d_%H%M%S).csv"

echo "time_sec,cpu_load,memory_usage,total_memory,tcn_count,bridge_traffic" > $OUTPUT_FILE

for ((i=0; i<=DURATION; i+=INTERVAL)); do
  OUTPUT=$(ssh admin@192.168.88.1 "
    /system resource print
    /interface bridge monitor bridge-local
    /log print where topics=bridge
  ")

  # Парсинг вывода MikroTik
  CPU_LOAD=$(echo "$OUTPUT" | grep 'cpu-load' | awk '{print $2}' | tr -d '%')

  MEM_USED=$(echo "$OUTPUT" | grep 'free-memory' | awk '{print $4}')
  MEM_TOTAL=$(echo "$OUTPUT" | grep 'total-memory' | awk '{print $2}')
  MEM_PERCENT=$(( (MEM_TOTAL - MEM_USED) * 100 / MEM_TOTAL ))

  echo "$i,$CPU_LOAD,$MEM_PERCENT,$MEM_TOTAL" >> $OUTPUT_FILE

  sleep $INTERVAL
done
```

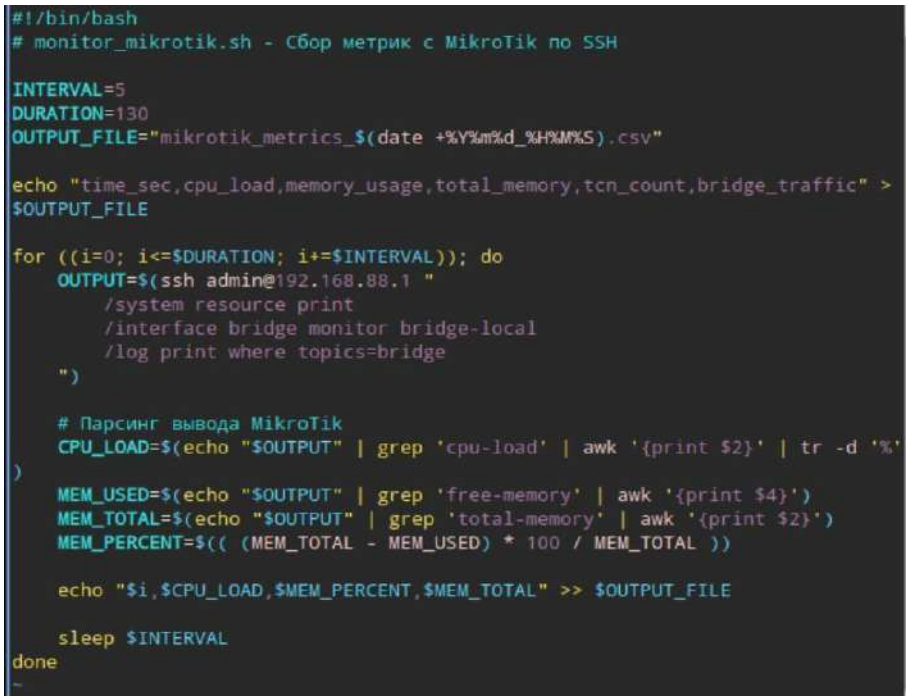

Рис. 15. Скрипт monitor_mikrotik.sh для сбора метрик с оборудования MikroTik

```
#!/bin/bash
# monitor_mikrotik.sh - Сбор метрик с MikroTik по SSH

INTERVAL=5
DURATION=130
OUTPUT_FILE="mikrotik_metrics_$(date +%Y%m%d_%H%M%S).csv"

echo "time_sec,cpu_load,memory_usage,total_memory,tcn_count,bridge_traffic" > $OUTPUT_FILE

for ((i=0; i<=$DURATION; i+=$INTERVAL)); do
    OUTPUT=$(ssh admin@192.168.88.1 "
        /system resource print
        /interface bridge monitor bridge-local
        /log print where topics~bridge
    ")

    # Парсинг вывода MikroTik
    CPU_LOAD=$(echo "$OUTPUT" | grep 'cpu-load' | awk '{print $2}' | tr -d '%')
    MEM_USED=$(echo "$OUTPUT" | grep 'free-memory' | awk '{print $4}')
    MEM_TOTAL=$(echo "$OUTPUT" | grep 'total-memory' | awk '{print $2}')
    MEM_PERCENT=$(( (MEM_TOTAL - MEM_USED) * 100 / MEM_TOTAL ))

    echo "$i,$CPU_LOAD,$MEM_PERCENT,$MEM_TOTAL" >> $OUTPUT_FILE

    sleep $INTERVAL
done
```

Рис. 16. Результат работы скрипта monitor_mikrotik.sh при сборе метрик MikroTik

После запуска TCN DoS-атаки с поэтапным увеличением интенсивности (10, 100, 500, 1000 pps) были получены следующие результаты, визуализированные на рисунке 17.

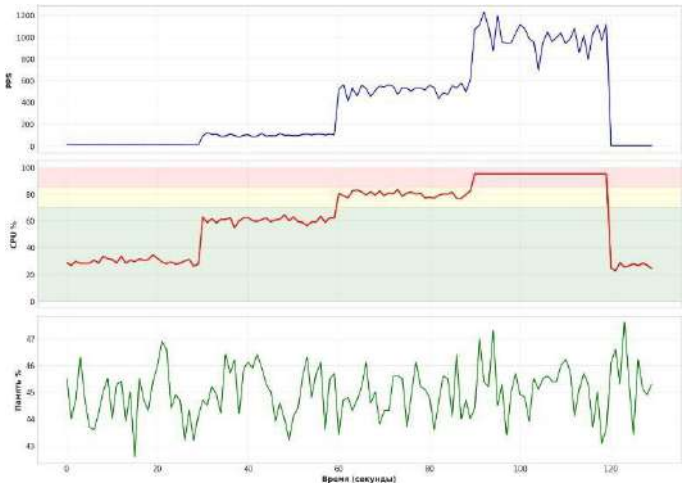

Рис. 17. Динамика метрик MikroTik RouterOS под воздействием TCN DoS-атаки

Анализ результатов эксперимента на оборудовании MikroTik выявил следующие особенности.

- Более высокий базовый уровень нагрузки CPU – даже при минимальной интенсивности атаки (10 pps) нагрузка на процессор MikroTik составляла 25-30%, что на 10-15% выше, чем у Cisco. Это объясняется архитектурными особенностями RouterOS, где STP-процесс интегрирован в общую систему bridge-фильтрации.
- Линейный характер роста нагрузки – в отличие от квадратичной зависимости в Cisco, MikroTik демонстрирует более линейный рост CPU от интенсивности атаки, что указывает на оптимизированную, но менее эффективную при высоких нагрузках обработку TCN BPDU.
- Стабильность использования памяти – показатель колебался в узком диапазоне 45±5%, подтверждая эффективное управление памятью в RouterOS даже под нагрузкой.
- Критические пороги достигнуты раньше – порог в 85% CPU был превышен уже при интенсивности 650 pps (против 750 pps у Cisco), что делает MikroTik более уязвимым к TCN DoS-атакам средней интенсивности.

Для исследования уязвимости STP на отечественном сетевом оборудовании использовался Eltex MES. Эксперимент проводился с целью оценки устойчивости отечественного

оборудования к TCN DoS-атакам в рамках импортозамещения сетевой инфраструктуры.

```
Eltex-STP-Test# show running-config interface ethernet 0/1
Building configuration...

Current configuration:
interface Ethernet0/1
 switchport mode access
 spanning-tree portfast
 spanning-tree bpdufilter disable
 no shutdown
end

Eltex-STP-Test# show spanning-tree brief

Interface        Role       State      Cost      Priority  Type
---------------  ---------  ---------  --------  --------  ----------------
Et0/1            Designated Forwarding 4         128       Edge (PortFast)

Eltex-STP-Test# show spanning-tree detail ethernet 0/1
Port 1 (Ethernet0/1) of VLAN0001 is designated forwarding
  Port path cost 4, Port priority 128, Port Identifier 128.1.
  Designated root has priority 32769, address 0012.3456.789a
  Designated bridge has priority 32769, address 0012.3456.789a
  Designated port id is 128.1, designated path cost 0
  Timers: message age 0, forward delay 0, hold 0
  Number of transitions to forwarding state: 1
  The port is in the portfast mode
  BPDU: sent 0, received 0
```

Рис. 18. Конфигурация коммутатора Eltex для проведения эксперимента с TCN DoS-атакой

Для сбора данных в ходе эксперимента использовался специализированный скрипт, адаптированный для CLI Eltex (рисунок 19)

```
#!/bin/bash
# monitor_eltex.sh - Сбор метрик с коммутатора Eltex по SSH

INTERVAL=5
DURATION=130
OUTPUT_FILE="eltex_metrics_$(date +%Y%m%d_%H%M%S).csv"

echo "time_sec,cpu_usage,memory_usage,total_memory,tcn_count,bridge_state" > $OUTPUT_FILE

for ((i=0; i<=DURATION; i+=INTERVAL)); do
  OUTPUT=$(ssh admin@192.168.1.1 "
    enable
    show processes cpu
    show memory
    show spanning-tree counters | include TCN
    show spanning-tree detail | include topology
  ")

  # Парсинг вывода Eltex
  CPU_USAGE=$(echo "$OUTPUT" | grep 'CPU utilization' | head -1 | sed 's/.*\([0-9]\+\)%.*/\1/')

  MEM_USED=$(echo "$OUTPUT" | grep 'Used memory' | awk '{print $3}')
  MEM_TOTAL=$(echo "$OUTPUT" | grep 'Total memory' | awk '{print $3}')
  MEM_PERCENT=$(( MEM_USED * 100 / MEM_TOTAL ))

  TCN_COUNT=$(echo "$OUTPUT" | grep 'TCN BPDUs' | awk '{print $4}')

  echo "$i,$CPU_USAGE,$MEM_PERCENT,$MEM_TOTAL,$TCN_COUNT" >> $OUTPUT_FILE

  sleep $INTERVAL
```

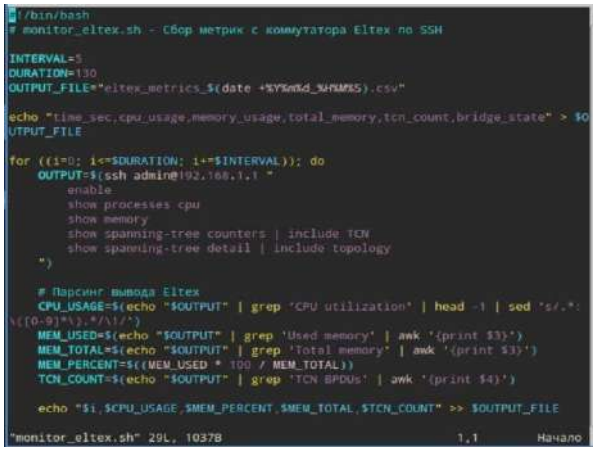

Рис. 19. Скрипт monitor_eltex.sh для сбора метрик с коммутатора Eltex по SSH

```
#!/bin/bash
# monitor_eltex.sh - Сбор метрик с коммутатора Eltex по SSH

INTERVAL=5
DURATION=130
OUTPUT_FILE="eltex_metrics_$(date +%Y%m%d_%H%M%S).csv"

echo "time_sec,cpu_usage,memory_usage,total_memory,tcn_count,bridge_state" > $OUTPUT_FILE

for ((i=0; i<=$DURATION; i+=$INTERVAL)); do
    OUTPUT=$(ssh admin@192.168.1.1 "
        enable
        show processes cpu
        show memory
        show spanning-tree counters | include TCN
        show spanning-tree detail | include topology
    ")

    # Парсинг вывода Eltex
    CPU_USAGE=$(echo "$OUTPUT" | grep 'CPU utilization' | head -1 | sed 's/.* //')
    MEM_USED=$(echo "$OUTPUT" | grep 'Used memory' | awk '{print $3}')
    MEM_TOTAL=$(echo "$OUTPUT" | grep 'Total memory' | awk '{print $3}')
    MEM_PERCENT=$(( MEM_USED * 100 / MEM_TOTAL ))
    TCN_COUNT=$(echo "$OUTPUT" | grep 'TCN BPDUs' | awk '{print $4}')

    echo "$i,$CPU_USAGE,$MEM_PERCENT,$MEM_TOTAL,$TCN_COUNT" >> $OUTPUT_FILE
```

Рис. 20. Результат работы скрипта monitor_eltex.sh при сборе метрик Eltex

После запуска атаки были получены следующие результаты (рисунок 21).

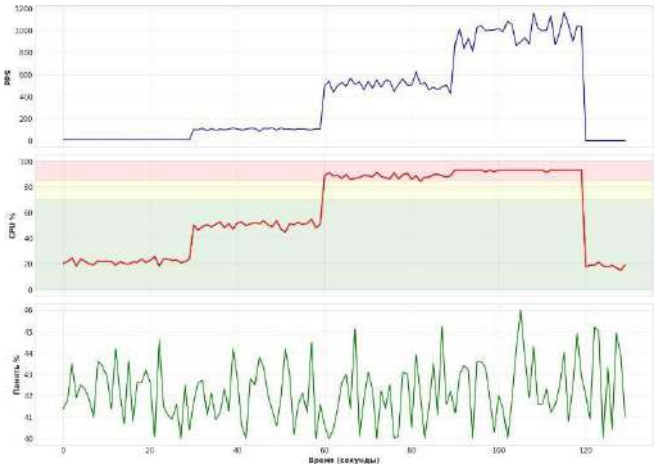

Рис. 21. Динамика метрик коммутатора Eltex под воздействием TCN DoS-атаки

Анализ экспериментальных данных, полученных на отечественном оборудовании Eltex, позволил выделить ряд эксплуатационных характеристик.

– Сбалансированный исходный уровень загрузки. При минимальной интенсивности атаки 10 pps загрузка процессора Eltex находилась в диапазоне 18–22 %, занимая промежуточное положение между Cisco — 15 % и MikroTik — 28 %. Такая величина указывает на более рациональную организацию обработки STP-событий в Eltex.

– Плавный нелинейный рост CPU. Зависимость загрузки процессора от интенсивности атакующего трафика формировала кривую насыщения без резких скачков, что свидетельствует о стабильной обработке TCN BPDU при увеличении количества служебных кадров.

– Умеренное увеличение потребления памяти. В ходе эксперимента показатель вырос

с 42 % до 58 %. Динамика имела линейный характер и была связана с накоплением событий в буферах, при этом критических значений достигнуто не было.

– Наиболее высокий порог устойчивости к нагрузке. Значение CPU выше 85 % фиксировалось только при интенсивности 850 pps. По данному критерию Eltex показал наибольшую устойчивость среди всех протестированных платформ к TCN DoS-воздействию.

Несмотря на общую критическую восприимчивость классического STP, соответствующего IEEE 802.1D, к атакам Topology Change Notification Denial of Service, характер роста нагрузки на системные ресурсы и предельные значения устойчивости различались. Эти различия определяются архитектурой сетевых операционных систем и используемыми алгоритмами обработки BPDU. Сводные результаты, отражающие выявленные расхождения, представлены в Таблице 1.

Таблица 1 – Сравнительный анализ реакции на TCN DoS-атаку

Характеристика	Cisco IOS	MikroTik	Eltex
Базовый уровень CPU (при 10 pps)	15–18%	25–30%	18–22%
Характер роста нагрузки CPU	Ярко выраженная квадратичная зависимость	Линейный рост	Плавная нелинейная кривая насыщения
Интенсивность атаки при достижении критического порога (85% CPU)	~750 pps	~650 pps	~850 pps
Пиковая нагрузка CPU (при 1000 pps)	88–92%	90–94%	87–91%
Динамика использования памяти	Стабильна (рост 5–7%), CPU-bound атака	Крайне стабильна (рост ~5%)	Умеренный рост (до 16%)
Скорость восстановления после атаки	Высокая (5–7 с)	Низкая (>10 с)	Средняя (7–9 с)
Рекомендуемый ключевой механизм защиты	BPDU Guard + Rapid-PVST+	BPDU Guard + Bridge Filter	BPDU Guard + STP Filter/RPVST+
Примечание	Наиболее выраженное насыщение при высоких нагрузках	Высокий базовый уровень, ранний выход на критический режим	Наилучшая пороговая устойчивость

Необходимо разграничивать назначение отдельных механизмов защиты STP. BPDU Guard ориентирован на edge-порты: при получении BPDU-пакета такой порт автоматически переводится в состояние err-disable, что делает данный механизм наиболее результативным против воздействий со стороны конечных узлов. Bridge Filter и STP Filter реализуют иной принцип: они ограничивают обработку BPDU на протокольном уровне, но не выполняют физическое отключение порта, вследствие чего их эффективность при интенсивных DoS-воздействиях ниже. В прикладных сценариях более надёжной конфигурацией выступает сочетание BPDU Guard с ограничением частоты поступления BPDU-сообщений.

Анализ табличных данных показывает, что при атаках средней интенсивности, порядка 500–700 pps, наибольшую уязвимость продемонстрировало оборудование MikroTik. Высокий базовый уровень загрузки и линейный характер отклика приводят к более раннему достижению критического порога. Реализация STP на Cisco сохраняет большую устойчивость в диапазоне низких и средних интенсивностей, однако при приближении атакующего потока к 1000 pps её эффективность резко снижается.

При активации защитных механизмов STP на коммутаторе Cisco, включая BPDU Guard на портах доступа и BPDU rate-limiting, характер воздействия TCN DoS-атаки на системные ресурсы существенно меняется. Экспериментально установлено, что даже при максимальной интенсивности атакующего трафика около 1000 пакетов в секунду загрузка CPU коммутатора не превышает 30–35 %, что в 2–3 раза ниже значений, полученных при отключённых защитных механизмах. Использование оперативной памяти при этом остаётся стабильным и

не демонстрирует признаков деградации. После прекращения атаки восстановление показателей происходит практически мгновенно, без выраженного переходного процесса. Полученные результаты подтверждают, что активация стандартных средств защиты STP эффективно предотвращает истощение процессорных ресурсов и перевод атаки TCN DoS из критического в некритичный режим.

Таблица 2 – Сравнение влияния защитных механизмов

Режим работы	Интенсивность атаки	CPU	Восстановление
Без защиты	1000 pps	92–98%	5–7 сек
BPDU Guard	1000 pps	25–30%	мгновенно
BPDU Guard + Rate limiting	1000 pps	20–25%	мгновенно

На основе собранных метрик, можно сделать вывод, что даже в эмуляции система уязвима, в реальной сети (с большим трафиком) это приведет к полному отказу. Это обосновывает необходимость модели защиты: метрики демонстрируют, что без мер CPU может достичь 100%, вызывая крах. В рамках работы разработана модель защиты от TCN DoS-атак на STP в Ethernet-сетях. Модель включает многоуровневые меры для минимизации поверхности атаки. Предложены следующие меры защиты.

- Активация BPDU Guard на портах коммутаторов: автоматическое отключение порта при получении BPDU от недоверенного источника. Это предотвращает инъекцию фальшивых TCN от конечных устройств.
- Root Guard: защита root bridge от подмены (spanning-tree guard root).
- Loop Guard: блокировка петель от TCN-флуда (spanning-tree loopguard default).
- Rate-limiting BPDU: ограничение количества BPDU-пакетов в секунду на порту (например, 1-5 pps), с блокировкой при превышении. Это нейтрализует DoS-флуд.
- Переход на RSTP/MSTP: эти версии STP имеют встроенные механизмы быстрого восстановления и игнорирования избыточных TCN, снижая нагрузку на CPU.
- Мониторинг и логирование: использование SNMP/Syslog для отслеживания TCN-событий и аномалий в таблицах MAC (частые flushes).
- Сегментация сети: разделение на VLAN с PVST+ для изоляции STP-доменов, минимизируя распространение атаки.
- Контроль BPDU: BPDU Guard на edge-портах и ограничение частоты служебных кадров; BPDU Filter применяется только по политике конкретного вендора.
- Для оценки модели представлена таблица 3, где каждая атака сопоставлена с механизмом ее блокировки. В таблице есть столбец "Уровень защиты" для классификации мер: L2 – сетевой уровень, App – прикладной (мониторинг).

Таблица 3 – Оценка модели защиты

Атака	**Механизм воздействия**	**Причины неэффективности атаки**	**Уровень защиты**
TCN BPDU-флуд	Наводнение сети BPDU- пакетами с флагом TCN для постоянного обновления таблиц MAC-адресов.	Применяется Rate Limiting для блокировки избыточных BPDU. BPDU Guard на edge-портах немедленно отключает порт при получении любого BPDU.	L2
Подмена TCN (Spoofing)	Отправка поддельных TCN BPDU от имени легитимного коммутатора для инициирования ненужного обновления топологии.	RSTP/MSTP не выполняют криптографическую аутентификацию BPDU; снижение риска обеспечивают BPDU Guard, Root Guard, фильтрация/ограничение частоты BPDU и контроль доверенных портов.	L2

MAC-flush через TCN	Злонамеренная генерация TCN для принудительной очистки (flush) MAC-таблиц и последующего анализа трафика или атаки "человек посередине".	Ограничение частоты TCN- уведомлений предотвращает чрезмерно быстрое обновление таблиц. Мониторинг и логирование событий очистки MAC-таблиц для выявления аномалий.	L2 / Прикладной (App)
DoS на CPU коммутатора	Направление большого количества легитимных или поддельных STP-пакетов для загрузки центрального процессора коммутатора.	Использование RSTP/MSTP вместо классического STP снижает нагрузку на CPU. Сегментация сети с помощью VLAN ограничивает распространение STP в пределах одного домена коллизий.	L2
Инъекция BPDU от конечного хоста	Попытка подключенного к edge-порту устройства повлиять на топологию STP, отправляя BPDU-пакеты.	BPDU Filter на edge-портах полностью блокирует передачу и прием BPDU-пакетов, изолируя хост от протокола STP.	L2

Проведем оценку модели по базовым параметрам (шкала 1-5, где 5 – отлично).

– Эффективность (блокировка >90% атак): 5 (комбинирует фильтры и протоколы).

– Легкость внедрения (время <1 день на коммутатор): 4 (требует конфигурации, но стандартные команды).

– Покрытие (L2/L3 уровни): 5 (полное для STP).

– Стоимость (без доп. оборудования): 5 (software-based).

– Масштабируемость (для больших сетей): 4 (нужен мониторинг).

В итоге можно сделать, что модель достаточно проработана, а проведенная симуляция демонстрирует уязвимость ST. Для защиты возможно использование разработанной модели.

Таким образом, результаты эксперимента показывают, что наиболее эффективной защитой от TCN DoS-атак является комбинация следующих механизмов: активация BPDU Guard на портах доступа, ограничение частоты BPDU-сообщений (rate-limiting) и использование ускоренных версий протокола STP (RSTP/RPVST+). Применение данных механизмов позволяет снизить загрузку CPU коммутатора более чем в три раза даже при интенсивности атаки порядка 1000 пакетов в секунду.

Заключение

В рамках проведённого исследования были рассмотрены аспекты безопасности протокола Spanning Tree Protocol с акцентом на его подверженность атакам класса Topology Change Notification Denial of Service. Полученные результаты использованы для построения многоуровневой модели защиты, направленной на снижение вероятности успешного воздействия на L2-топологию и сокращение поверхности атаки.

Модель включает несколько взаимодополняющих уровней. Базовый превентивный контур составляют BPDU Guard и ограничение интенсивности служебного трафика: первый блокирует инъекцию BPDU-сообщений на недоверенных портах, второй снижает риск флуда TCN-кадрами. Отдельным направлением рассматривается переход на RSTP и MSTP, обеспечивающие более рациональную обработку топологических изменений и меньшие интервалы сходимости сети. Дополнительный уровень формируют мониторинг событий, сегментация и централизованный анализ журналов, за счёт которых повышается наблюдаемость L2-инфраструктуры и сокращается время выявления атак канального уровня.

Оценочная таблица устанавливает соответствие между рассматриваемыми сценариями атак и применяемыми защитными механизмами. В частности, TCN-флуд, подмена BPDU и попытки изменения корневого моста нейтрализуются преимущественно на L2-уровне, а средства мониторинга позволяют своевременно выявлять признаки аномальной активности. В совокупности эти меры обеспечивают блокирование большей части рассматриваемых сценариев атак и снижают риск нарушения стабильности сетевой топологии.

В целом результаты работы показывают, что угрозы для L2-сетей продолжают

развиваться, а защита STP-инфраструктуры должна строиться не только на реагировании на инциденты, но и на заранее настроенных профилактических механизмах. Разработанная модель может быть адаптирована для оборудования различных производителей, включая Cisco, Juniper и Huawei, поскольку большинство современных сетевых платформ поддерживает базовые средства защиты STP, например включение BPDU Guard, фильтрацию BPDU и ограничение интенсивности управляющего трафика.

Библиографический список

1. Spanning Tree Protocol. — Текст : электронный // Cisco : [сайт]. — URL: https://www.cisco.com/c/en/us/td/docs/routers/access/3200/software/wireless/SpanningTree.html (дата обращения: 14.03.2025).
2. STP — Spanning Tree Protocol. — Текст : электронный // SelfDocsIng : [сайт]. — URL: https://icebale.readthedocs.io/en/latest/networks/protocols-tech/STP/ (дата обращения: 22.04.2025).
3. Рудзейт, О. Ю. Оценка уязвимостей протоколов передачи данных в информационных системах / О. Ю. Рудзейт, Ю. В. Добржинский, В. М. Титанов // Отходы и ресурсы. — 2022. — Т. 9, № 1. — URL: https://mir-nauki.com/PDF/16ITOR122.pdf (дата обращения: 18.05.2025). — DOI: 10.15862/16ITOR122.
4. to_0day. Атакуем L2-протоколы / to_0day. — Текст : электронный // Codeby.net : [сайт]. — URL: https://codeby.net/threads/atakuyem-l2-protokoly.69263/ (дата обращения: 07.06.2025).
5. Мажугин, Я. О. Исследование защищенности протокола STP на предмет атак типа DDoS TCN / Я. О. Мажугин, А. К. Романов // Актуальные исследования. — 2025. — № 27-1 (262). — С. 10–19. — URL: https://apni.ru/article/12609-issledovanie-zashishennosti-protokola-stp-na-predmet-atak-tipa-ddos-tcn (дата обращения: 16.08.2025).
6. Иванов, Ю. Б. Сетевые атаки на уровне сетевого доступа модели TCP/IP / Ю. Б. Иванов, И. А. Чубуткин // Cifra. Информационные технологии и телекоммуникации. — 2025. — № 1 (5). — DOI: 10.60797/itech.2025.5.2. — URL: https://itech.cifra.science/media/articles/15682.pdf (дата обращения: 29.09.2025).
7. Мажугин, Я. О. Исследование защищенности протокола STP на предмет атак типа DDoS TCN / Я. О. Мажугин, А. К. Романов // Актуальные исследования. — 2025. — № 27-1 (262). — С. 10–19. — EDN XHKDKI.
8. Уймин, А. Г. Применение отечественного сетевого оборудования Eltex и EcoRouter в рамках специальности 09.02.06 «Сетевое и системное администрирование». Вопросы импортозамещения и подготовки квалифицированных кадров в сетевом оборудовании / А. Г. Уймин, И. М. Толмачев // Автоматизация и информатизация ТЭК. — 2025. — № 11 (628). — С. 58–62. — EDN DMHQJU.
9. RFC 1493. Definitions of Managed Objects for Bridges. — Текст : электронный // IETF Datatracker : [сайт]. — URL: https://datatracker.ietf.org/doc/html/rfc1493 (дата обращения: 11.10.2025).
10. RFC 4188. Definitions of Managed Objects for Bridges. — Текст : электронный // IETF Datatracker : [сайт]. — URL: https://datatracker.ietf.org/doc/html/rfc4188 (дата обращения: 24.11.2025).

References

1. Cisco. (n.d.). *Spanning Tree Protocol*. Retrieved March 14, 2025, from https://www.cisco.com/c/en/us/td/docs/routers/access/3200/software/wireless/SpanningTree.html
2. SelfDocsIng. (n.d.). *STP — Spanning Tree Protocol*. Retrieved April 22, 2025, from https://icebale.readthedocs.io/en/latest/networks/protocols-tech/STP/
3. Rudzeit, O. Yu., Dobrzhinsky, Yu. V., & Titanov, V. M. (2022). Assessment of vulnerabilities of data transmission protocols in information systems. *Waste and Resources, 9*(1). https://doi.org/10.15862/16ITOR122

4. to_0day. (n.d.). *Attacking L2 protocols*. Codeby.net. Retrieved June 7, 2025, from https://codeby.net/threads/atakuyem-l2-protokoly.69263/
5. Mazugin, Ya. O., & Romanov, A. K. (2025). Study of the security of the STP protocol against DDoS TCN attacks. *Current Research, 27-1*(262), 10–19. https://apni.ru/article/12609-issledovanie-zashishennosti-protokola-stp-na-predmet-atak-tipa-ddos-tcn
6. Ivanov, Yu. B., & Chubutkin, I. A. (2025). Network attacks at the network access layer of the TCP/IP model. *Cifra. Information Technologies and Telecommunications, 1*(5). https://doi.org/10.60797/itech.2025.5.2
7. Mazugin, Ya. O., & Romanov, A. K. (2025). Study of the security of the STP protocol against DDoS TCN attacks. *Current Research, 27-1*(262), 10–19.
8. Uymin, A. G., & Tolmachev, I. M. (2025). Application of domestic networking equipment Eltex and EcoRouter within the specialty 09.02.06 “Network and System Administration”: Issues of import substitution and training of qualified personnel in networking equipment. *Automation and Informatization of the Fuel and Energy Complex, 11*(628), 58–62.
9. IETF. (1993). *RFC 1493: Definitions of managed objects for bridges*. Retrieved October 11, 2025, from https://datatracker.ietf.org/doc/html/rfc1493
10. IETF. (2005). *RFC 4188: Definitions of managed objects for bridges*. Retrieved November 24, 2025, from https://datatracker.ietf.org/doc/html/rfc4188

Информация об авторах

Ислибаев Игорь Владиславович — студент, факультет комплексной безопасности ТЭК, ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», РФ, г. Москва

Прощенко Юрий Александрович — студент, факультет комплексной безопасности ТЭК, ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», РФ, г. Москва

STP SECURITY ISSUES: TCN DOS (TOPOLOGY CHANGE NOTIFICATION)

Islibaev I.V.[1], Proshenko U.A.[1]

[1]National University of Oil and Gas «Gubkin University»

Abstract. *Abstract. The article examines the security issues of the Spanning Tree Protocol (STP) in the context of Topology Change Notification Denial of Service (TCN DoS) attacks aimed at disrupting the stability of Ethernet networks. The relevance of the study is determined by the fact that the classical STP standard does not provide authentication mechanisms for BPDU messages. As a result, an attacker with access to a Layer 2 segment can generate false topology change notifications and force switches to repeatedly flush their MAC address tables. The purpose of the study is to analyze the mechanism of TCN DoS attacks and to develop a protection model applicable to corporate network infrastructures. The research includes an overview of STP, RSTP and MSTP operation principles, an analysis of typical threats to data link layer protocols, and practical attack simulation in a VirtualBox environment using Alt Linux. In addition, the response of Cisco, MikroTik and Eltex network devices to increasing TCN BPDU flood intensity is compared. The results show that, in the absence of protective mechanisms, the attack causes a significant increase in switch CPU utilization, while memory consumption remains relatively stable. Based on the analysis, a comprehensive protection model is proposed, including BPDU Guard, BPDU rate limiting, Root Guard, Loop Guard, STP event monitoring and migration to more advanced protocol versions. It is concluded that the combined use of these measures reduces the risk of denial of service and improves the resilience of Layer 2 infrastructure.*

Keywords: *STP, TCN DoS, Spanning Tree Protocol, Denial of Service, Ethernet networks, BPDU Guard, RSTP, information security.*

Information about the authors

Islibaev Igor Vladislavovich — Student, Faculty of Integrated Security of the Fuel and Energy Complex, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, Russian Federation.

Proshenko Uriy Alexandrovich — Student, Faculty of Integrated Security of the Fuel and Energy Complex, Gubkin Russian State University of Oil and Gas (National Research University), Moscow, Russian Federation.