Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. каф. ЭВМ

\_\_\_\_\_\_\_\_\_\_ Д.И. Самаль

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

ПРОГРАММНОЕ СРЕДСТВО ТЕСТИРОВАНИЯ НАДЕЖНОСТИ СЕТЕВЫХ РЕСУРСОВ

БГУИР ДП 1–40 02 01 01 189 ПЗ

Студент М.П. Трафимук

Руководитель Н.Н. Иванов

Консультанты:

от кафедры ЭВМ Н.Н. Иванов

по экономической части В.И. Шкода

Нормоконтролер А.С. Сидорович

Рецензент

МИНСК 2017

РЕФЕРАТ

Дипломный проект предоставлен следующим образом. Электронные носители: 1 компакт-диск. Чертежный материал: 6 листов формата А1. Пояснительная записка: 57 страниц, 5 рисунков, 3 таблицы, 8 литературных источников, 4 приложения.

Ключевые слова: нагрузочное тестирование, сетевое приложение.

Объектом исследования и разработки являются наиболее распространенные атаки типа «отказ в обслуживании» и методы защиты от них.

Целью проекта является разработка программного средства, которое позволяет протестировать защиту ресурса от атак типа «отказ в обслуживании».

При разработке программного средства тестирования надежности сетевых ресурсов была использована среда разработки Microsoft Visual Studio 2012, фреймворк .NET, библиотека WinPCap.

Областью практического применения программного средства являются сетевые ресурсы. Разрабатываемый программный продукт подразумевает собой некоторое решение, которое выполняет нагрузочное тестирование сетевого ресурса в аспекте использования сети. Системный администратор запускает приложение на одном или нескольких компьютерах. После проверяет доступность ресурса из произвольного источника. В случае если ресурс недоступен или обрабатывает запросы слишком долго – защита недостаточна.

Разработанный программный продукт можно считать экономически эффективным, и он полностью оправдывает вложенные в него средства.

Дипломный проект является завершенным, поставленная задача решена в полной мере, присутствует возможность дальнейшего развития программы и увеличение её функционала.

Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: КСиС. Кафедра: ЭВМ.

Специальность: 40 02 01 «Вычислительные машины, системы и сети».

Специализация: 40 02 01-01 «Проектирование и применение локальных компьютерных сетей».

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Д.И. Самаль

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ

по дипломному проекту студента

Трафимука Максима Павловича

**1** Тема проекта: «Программное средство тестирования надежности сетевых ресурсов» – утверждена приказом по университету от 13 апреля 2017 г. № 627-с.

**2** Срок сдачи студентом законченного проекта: 1 июня 2017 г.

**3** Исходные данные к проекту:

**3.1** Операционная система: Windows.

**3.2** Среда разработки: Microsoft Visual Studio.

**3.3** Языки программирования: C, C++.

**4** Содержание пояснительной записки (перечень подлежащих разработке

вопросов):

Введение 1. Обзор литературы. 2. Системное проектирование.   
3. Функциональное проектирование. 4. Разработка программных модулей.   
5. Программа и методика испытаний. 6. Руководство пользователя. 7. Технико-экономическое обоснование эффективности разработки и реализации Программное средство тестирования надежности сетевых ресурсов. Заключение. Список использованных источников. Приложения.

**5** Перечень графического материала (с точным указанием обязательных

чертежей):

**5.1** Вводный плакат. Плакат.

**5.2** Программное средство тестирования надежности сетевых ресурсов. Схема структурная.

**5.3** Программное средство тестирования надежности сетевых ресурсов. Схема программы.

**5.4** Программное средство тестирования надежности сетевых ресурсов. Диаграмма последовательности.

**5.5** Классификация типов DDoS-атак. Плакат.

**5.5** Заключительный плакат. Плакат.

**6** Содержание задания по экономической части: «Технико-экономическое обоснование эффективности разработки и реализации программного средства тестирования надежности сетевых ресурсов».

ЗАДАНИЕ ВЫДАЛ В.И. Шкода

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов  дипломного проекта | Объем  этапа,  % | Срок выполнения этапа | Примечания |
| Подбор и изучение литературы | 10 | 02.02 – 25.03 |  |
| Структурное проектирование | 20 | 25.03 – 14.04 |  |
| Функциональное проектирование | 20 | 14.04 – 23.04 |  |
| Разработка программных модулей | 20 | 23.04 – 04.05 |  |
| Программа и методика испытаний | 10 | 05.05 – 09.05 |  |
| Технико-экономическое обоснование | 10 | 09.05 – 18.05 |  |
| Оформление пояснительной записки | 10 | 18.05 – 25.05 |  |

Дата выдачи задания: 02 февраля 2017г.

Руководитель Н.Н. Иванов

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_

УСЛОВНЫЕ СОКРАЩЕНИЯ

DoS (Denial of Service) – отказ в обслуживании

OSI (Open Systems Interconnection Basic Reference Model) – базовая эталонная модель взаимодействия открытых систем

CUDA (Compute Unified Device Architecture) – унифицированная программно-аппаратная архитектура

TCP (Transmission Control Protocol) – протокол управлений передачей

UDP (User Datagram Protocol) – протокол пользовательских датаграмм

HTTP (Hyper Text Transfer Protocol) – протокол передачи гипертекста

ICMP (Internet Control Message Protocol) – протокол межсетевых управляющих сообщений

IP (Internet Protocol) – межсетевой протокол

DNS (Domain Name System) – система доменных имен

SSL (Secure Sockets Layer) – уровень защищенных сокетов

RUDY (Are You Dead Yet) – ты еще мертв

FFT (Fast Fourier Transform) – быстрое преобразование Фурье

BLAS (Basic Linear Algebra Subprograms) – базовые подпрограммы линейной алгебры

API (Application Programing Interface) – интерфейс программирования приложений

SDK (Software Development Kit) – комплект средств разработки

GPU (Graphics Processing Unit) – графическое процессорное устройство

CPU (Central Processing Unit) – центральное процессорное устройство

SIMD (Single Instruction Multiple Data) – одна инструкция, множество данных

PPPoE (Point-to-Point Protocol over Ethernet) – протокол канального уровня передачи кадров PPP по Ethernet

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc483629348)

[1 ОБЗОР ЛИТЕРАТУРЫ 9](#_Toc483629349)

[**1.1** Классификация DDoS-атак по уровням модели OSI 9](#_Toc483629350)

[**1.2** Виды DDoS-атак 10](#_Toc483629351)

[**1.3** Методы защиты от основных видов DDoS-атак 13](#_Toc483629352)

[**1.4** Методы усиления атак 15](#_Toc483629353)

[**1.5** Основные рекомендации по защите от DDoS-атак 17](#_Toc483629354)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 18](#_Toc483629355)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 20](#_Toc483629356)

[**3.1** Модуль атакующего процесса 20](#_Toc483629357)

[**3.2** Модуль интерфейса 28](#_Toc483629358)

[**3.3** Модуль определения идентификатора текущей PPPoE сессии 30](#_Toc483629359)

[**3.4** Модуль мониторинга 31](#_Toc483629360)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 32](#_Toc483629361)

[**4.1** Модуль атакующего процесса 32](#_Toc483629362)

[**4.2** Модуль интерфейса 36](#_Toc483629363)

[**4.3** Модуль определения идентификатора текущей PPPoE сессии 38](#_Toc483629364)

[**4.4** Модуль мониторинга 39](#_Toc483629365)

[5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 40](#_Toc483629366)

[**6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 46](#_Toc483629367)

[**6.1** Требования к аппаратному обеспечению 46](#_Toc483629368)

[**6.2** Руководство по установке системы 46](#_Toc483629369)

[**6.3** Руководство по использованию программного средства 46](#_Toc483629370)

[7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВА ТЕСТИРОВАНИЯ НАДЕЖНОСТИ СЕТЕВЫХ РЕСУРСОВ 51](#_Toc483629371)

[**7.1** Характеристика программного продукта 51](#_Toc483629372)

[**7.2** Расчет сметы затрат и отпускной цены программного средства 51](#_Toc483629373)

[**7.3** Расчет экономического эффекта продажи программного продута 55](#_Toc483629374)

[**7.4** Расчет показателей экономической эффективности разработки и реализации программного продукта 56](#_Toc483629375)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc483629376)

[ПРИЛОЖЕНИЕ А 60](#_Toc483629377)

[ПРИЛОЖЕНИЕ Б 61](#_Toc483629378)

# ВВЕДЕНИЕ

Современные тенденции развития информационных технологий предполагают максимальную виртуализацию и абстрагирование от конкретных рабочих станций и все больший переход к облачным и сетевым системам хранения и обработки данных. Эти решения сопряжены с меньшим риском потери информации и большей вычислительной мощностью таких систем, а также с их независимостью от географического местоположения пользователя и мощности его личного устройства, взаимодействующего с системой.

Основополагающими принципами систем хранения данных и центров обработки данных являются: безопасность, надежность и доступность. Для обеспечения надежности используются дублирующие компоненты, для обеспечения безопасности различные виды аутентификации, а для обеспечения доступности многие организации не предпринимают практически ничего, что является серьезной проблемой, потому что этот аспект является самым легкодоступным для внешних вредоносных действий.

В данной работе рассматривается только один аспект, а именно доступность.

Снижение доступности ресурса может подорвать доверие и заставить понести серьезные убытки как организации предоставляющей услуги, так и клиентам этой организации. Следует заметить, что убытки появляются именно из-за невозможности получить доступ к информации, если смотреть с точки зрения клиента, или же предоставить эту информацию, если смотреть на проблему с точки зрения поставщика услуг. Некоторые причины снижения доступности, организованные злоумышленниками или соперниками, могут принести серьезные проблемы, но могут быть обезврежены тем или иным способом.

Семейство причин снижения доступности, организованные злоумышленниками или соперниками называется Distributed Denial of Service (DDoS), распределенная атака типа «отказ в обслуживании». Иными словами, когда злоумышленник загружает канал доступа к ресурсу клиенты не могут воспользоваться этим ресурсом, таким образом нарушается принцип доступности.

За годы развития глобальной сети было много прецедентов разной известности, когда группа злоумышленников причиняла серьезные убытки простым ограничением доступности. Самыми известными являются такие как Operation Payback в 2010 году длившаяся несколько месяцев, атака на EveryDNS, атака на крупнейший хостинг GoDaddy 10 ноября 2012 года. В результате этих атак пострадали миллионы сайтов и еще большее количество пользователей. Так как эти атаки караются Уголовным Кодексом, то их участники неизвестны, зачастую известны только страны из которых они производились. Найденных участников всего 2: Алекс Тапанарис и некий Jeroenz0r, шестнадцатилетний школьник из Нидерландов. Оба участвовали в Operation Payback, но так как мероприятие это было децентрализованным невозможно сказать насколько важны они были для атаки.

Целью этого дипломного проекта является разработка программного продукта, позволяющего эмулировать атаки на сетевые ресурсы для получения необходимого опыта системными администраторами и проверки надежности уже выстроенной защиты. В ходе создания проекта необходимо учитывать, что от масштабных атак нет надежной защиты, и для них нужно большое количество участников, таким образом на ресурсы, не являющиеся всемирно востребованными такие атаки не производятся. Также в качестве отрицательного эффекта объема всемирной сети является ее статичность, таким образом эмуляции самых распространенных видом атак достаточно для подготовки к 90% угроз доступности.

В соответствии с поставленной целью были определены следующие задачи:

* + выбор платформы и средств для разработки;
  + имитация наиболее распространенных видов атак семейства DDoS;
  + реализация предсказуемых методов усиления атак;
  + тестирование надежности сетевых ресурсов в части доступности;
  + реализация модуля мониторинга доступности ресурса во время атаки;

# 1 ОБЗОР ЛИТЕРАТУРЫ

Для достижения поставленных целей и эмуляции атак типа «отказ в обслуживании» необходимо рассмотреть основные типы существующих атак, способы их усиления, а так же различные виды защиты от них. Так как разрабатываемое приложение направлено на эмуляцию атак, а не на защиту от них, то анатомию угрозы надо рассмотреть в деталях. Далее рассмотрены самые часто встречающиеся виды атак типа «отказ в обслуживании», а так же способы их усиления и способы защиты от этих атак.

Наилучшей классификацией DDoS-атак является их разделение по уровням модели OSI, на которых они проводятся.

## **1.1** Классификация DDoS-атак по уровням модели OSI

*7-й уровень OSI*: посредством протоколов HTTP, FTP, POP3, SMTP производятся атаки, приводящие к чрезмерному потреблению системных ресурсов у атакуемого сервера. К таковым относятся HTTP-флуд, PDF GET, HTTP GET и HTTP POST атаки.

*6-й уровень OSI*: посредством протокола SSL производятся атаки, полностью загружающие этот протокол и не позволяющие клиентам получить доступ к информации посредством этого протокола. К таковым относятся подложные SSL-запросы и иные атаки под SSL.

*5-й уровень OSI*: посредством протоколов Telnet и PAP отрезает возможность удаленного доступа системного администратора к устройству, но не является DDoS-атакой, поэтому не рассматривается в рамках данного проекта. К таковым относятся атаки на протокол Telnet.

*4-й уровень OSI*: посредством протоколов TCP и UDP перегружается канал доступа в сеть для атакуемого устройства или же заполняется количество возможных подключений. К таковым относятся Fraggle атака, SYN-флуд, приумножение DNS, Low and Slow атаки.

*3-й уровень OSI*: посредством протоколов ICMP и IP перегружается канал доступа атакуемого устройства в сеть. К таковым относятся ping-флуд и ICMP-флуд.

*2-й уровень OSI*: посредством протоколов 802.3 и 802.5 перегружается ближайший к атакующему свитч или роутер, что приводит к блокированию всех портов. Используется только для локальных атак, поэтому в данном проекте не рассматривается. К атакам этого уровня относится только MAC-флуд.

*1-й уровень OSI*: на этом уровне возможно только физическое разрушение линий связи и сетевых устройств, которое по очевидным причинам не рассматривается в проекте.

Следует рассмотреть основные виды атак и защиты от них в соответствии с приведенной классификацией.

## **1.2** Виды DDoS-атак

На 7 уровне модели OSI:

*HTTP-флуд*

Для атаки на сервер в качестве аналога ping-флуда на высоком уровне модели OSI обычно применяется HTTP-флуд. Атакующий шлёт маленький по объёму HTTP-пакет, с содержимым, на которое сервер должен ответить пакетом, размер которого в сотни раз больше. Даже если канал сервера в десятки раз шире канала атакующего, то все равно есть большой шанс насытить полосу пропускания жертвы. А для того, чтобы ответные HTTP-пакеты не вызвали отказ в обслуживании у атакующего, он каждый раз подменяет свой ip-адрес на ip-адреса узлов в сети[1].

На 6 уровне модели OSI:

*Атаки под SSL* – подразумевается, что атакующий может маскировать свои деструктивные действия внутри SSL-трафика, что значительно усложняет противодействие. Протокол SSL работает поверх TCP/IP, обеспечивая безопасность обмена информацией для пользователей. Можно говорить об атаках на сам процесс установки SSL-взаимодействия (SSL handshake), отправка «мусорных» пакетов серверу или злоупотребление функциями согласования ключевой информации и так далее. Так как обработка пакетов SSL является достаточно ресурсозатратным [5] действием этот тип атак нагружает в основном не сеть, а систему, которая обрабатывает эти пакеты, существующие же разновидности защиты только усиливают нагрузку на систему не обеспечивая ее безопасности, таким образом наверняка перегружая цель. Но атаки такого типа достаточно редки и сложны и на данном этапе развития информационных технологий практически не используются.

На 4 уровне модели OSI:

##### *Атака Fraggle (UDP-флуд)*

Атака Fraggle (осколочная граната) является полным аналогом Smurf-атаки, но вместо протокола ICMP используется протокол UDP, поэтому её ещё называют UDP-флуд. Принцип действия этой атаки такой же как у ICMP-флуда. Если все же служба echo отключена, что будет достаточным для защиты от предыдущего типа атак, то будут сгенерированы ICMP-сообщения, что также приведёт к насыщению полосы, тем самым сведя защиту от предыдущего способа на нет и снова приводя ресурс к недоступности на протяжении всей атаки [2].

*Приумножение DNS-запросов*

Метод приумножения DNS-запросов позволяет поставить на службу атакующему огромное количество ресурсов сети Интернет. DNS — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене. Одним из флагов DNS-запроса является требование рекурсии, что в контексте DNS означает поиск адреса на других серверах если на этом его нет, таким образом сформировав рекурсивный запрос на получение полной таблицы имен и адресов каждый доступный неправильно сконфигурированный сервер отправит цели достаточно большой объем данных [4], что перегрузит полозу пропускания на очень долгий срок, сложные системы с использованием данного типа атак могут создавать потоки данных до 300 Гбит/с по единичной цели. Таким способом в 2013 году была произведена одна из мощнейших атак в истории. Парадоксальным в данном типе атак является тот факт, что для защиты от такого типа атак был создан протокол DNSSec, в большинстве случаев только увеличивающий размер пакетов и, соответственно, нагрузку на целевое устройство [4].

##### *Атака с помощью переполнения пакетами SYN (SYN-флуд)*

До появления атаки Smurf была широко распространена атака с помощью переполнения пакетами SYN, также известная под названием SYN-флуд. Для описания её действия необходимо знать принцип установки соединения на основе тройного рукопожатия. Если 2 устройства хотят установить между собой TCP соединение, после которого они смогут обмениваться между собой данными, то на установку соединения выделяется некоторое количество ресурсов, этим и пользуется этот тип DoS-атаки. Отправив несколько сотен ложных запросов, можно израсходовать все ресурсы системы, отведённые на установление соединения, таким образом легитимные клиенты не смогут получить доступ к атакуемому ресурсу. Рассмотрим подробнее, как это происходит. Атакующий отправляет SYN пакет цели, но предварительно меняет свой IP-адрес на несуществующий. Затем, устройство-цель отправляет ответ SYN/ACK на несуществующий IP-адрес и переходит в состояние SYN-RECEIVED. Так как сообщение SYN/ACK не дойдет до какого-либо устройства, то цель никогда не получит пакет с флагом ACK. Данное потенциальное соединение будет помещено в очередь. Из очереди по умолчанию оно выйдет только по истечении 75 секунд. В этом и состоит принцип атаки: отправка сразу большого количества пакетов SYN на компьютер жертвы с небольшим интервалом, чтобы полностью исчерпать ресурсы системы [3], выделенные на подключение новых пользователей. Определить источник нападения очень непросто, так как атакующий постоянно меняет исходный IP-адрес.

***Медленные атаки малого объема, так называемые Low and Slow.***

Этот вариант представляет большую опасность в силу малой заметности и продолжительного времени нарастания зловредного воздействия.

*RUDY*

Принцип этой атаки базируется на стандартном поведении протокола HTTP при обработке POST-запросов. На многих сайтах, таких как интернет-магазины, социальные сети, интернет-банки есть возможность аутентификации по логину и паролю. Когда обычный пользователь заполняет веб-форму, на сайт отправляется всего несколько пакетов, и сессия с веб-сервером закрывается, ресурсы сервера освобождаются, после чего он готов обслуживать запросы других пользователей. Атакующий, использующий тип атаки RUDY, действует иначе. Отправляемые на веб-сервер данные разбиваются на множество пакетов, каждый из которых содержит лишь один байт данных. Запросы на сервер отправляются со случайным небольшим интервалом, что не дает возможность серверу закрыть сессию, так как передача данных еще не завершена. Несколько тысяч таких запросов в течение небольшого промежутка времени приводят к тому, что сервер не может отвечать на запросы легитимных клиентов, потому что все ресурсы заняты сессиями атакующего. Не требуется большого объема трафика, чтобы вывести ресурс из строя. Все запросы абсолютно легитимны, здесь имитируется поведение системы с медленным каналом связи. Цель атакующего достигнута – работать с сайтом невозможно. Подобная уязвимость характерна практически для любого сайта.

*Атака SlowLoris.*

Так же, как и предыдущая, базируется на стандартном поведении протокола HTTP при обработке запросов. Для закрытия HTTP-сессии необходимо прислать соответствующую последовательность. Так и поступает обычный пользователь. Правильный запрос на получение данных с веб-сервера (Get Request) обычно состоит из одного пакета и завершается специальной последовательностью в конце для разрыва сессии. Суть атаки заключается в следующем: атакующий генерирует множественные подключения к целевому веб-серверу, при этом соединения не закрываются, поскольку в запросе будет отсутствовать соответствующая последовательность символов. В результате ресурсы сервера будут тратиться на поддержку соединений атакующего, и легитимные пользователи не смогут подключиться, тем самым цель атаки достигнута – работать с сайтом невозможно. Для этой атаки также не требуется большого объема трафика, чтобы вывести ресурс из строя. Все запросы абсолютно легитимны, именно поэтому стандартным средствам борьбы с DDoS крайне сложно идентифицировать такую атаку и противостоять ей. Атаки такого типа являются аналогами SYN-флуда для протокола HTTP [6].

На 3 уровне модели OSI:

##### *Ping-флуд*

Это самый примитивный вид DoS-атаки. Насыщение полосы пропускания можно осуществить с помощью обычных ping-запросов только в том случае, если канал атакующего намного шире канала компьютера-жертвы. Но такая атака бесполезна против сервера или любого иного сетевого ресурса, предоставляющего свои услуги большому количеству пользователей, так как он обладает довольно широкой полосой пропускания.

*Smurf-атака (ICMP-флуд)*

Атака Smurf или ICMP-флуд — один из самых опасных видов DoS-атак, так как у компьютера-цели после такой атаки произойдет отказ в обслуживании практически со 100 % гарантией. Атакующий использует широковещательную рассылку для проверки работающих узлов в системе, отправляя ping-запрос. В одиночку вывести из строя компьютер-цель таким образом невозможно, поэтому требуется усиливающая сеть. В ней по широковещательному адресу атакующий отправляет поддельный ICMP пакет, в котором адрес атакующего меняется на адрес жертвы. Все узлы пришлют ей ответный пакет. Поэтому ICMP-пакет, отправленный через усиливающую сеть, содержащую 250 узлов, будет усилен в 250 раз, при отправке огромного количества пакетов даже незначительного усиления достаточно чтобы сделать цель недоступной, а значит при значительном усилении у цели не обеспеченной должной защитой нет шансов восстановить доступность до конца атаки. Поэтому для такой атаки обычно выбирается большая сеть, чтобы у компьютера-жертвы не было никаких шансов.

По аналогии с рассмотренными атаками следует рассмотреть защиту от этих атак в порядке классификации приведенной ранее.

**1.3** Методы защиты от основных видов DDoS-атак

На 7 уровне модели OSI:

##### *HTTP-флуд*

Для защиты от HTTP-флуда необходимо увеличить одновременное количество максимальных подключений к базе данных сервера, а так же настроить серверный кэш для более быстрой обработки.

На 6 уровне модели OSI:

*Атаки под SSL*

Так как этот протокол создавался для защищенной передачи данных, то атаки с его использованием крайне сложны и на данный момент неизвестно ни одной проводившейся не в академических целях. Существующие виды защиты от этих атак только усугубляют ситуацию при правильно поставленной атаке, но для ее проведения необходимы слишком большие ресурсы, поэтому она не используется, хотя теоретически может стать серьезной проблемой.

На 4 уровне модели OSI:

##### *Атака Fraggle (UDP-флуд)*

Так как UDP-пакеты отсылаются на различные UDP-сервисы, то достаточно просто отключить их от внешнего мира и установить ограничение на количество соединений к DNS-серверу, что так же является единственной защитой на стороне атакуемого при атаке с помощью приумножения DNS-запросов [2].

*Приумножение DNS-запросов*

Помимо вышеописанного метода защиты, который является единственным, но не очень хорошим, необходима правильная настройка существующих в глобальной сети DNS-серверов, так как для такой атаки необходим неправильно настроенный сервер. Текущее же положение дел оставляет желать лучшего, так как огромное количество небольших, а так же несколько достаточно крупных DNS-серверов по-прежнему имеют неправильную конфигурацию. Компания Cloudflare ежемесячно публикует статистику в которой можно увидеть, что наибольшее количество таких серверов находятся в Корее и Японии (по 800-900 штук у организаций-провайдеров). Защиты же от крупномасштабных атак не существует вообще до правильной перенастройки всех серверов.

##### *Атака с помощью переполнения пакетами SYN (SYN-флуд)*

Защита строится на отключении очереди «полуоткрытых» TCP-соединений, так же необходимо включение механизма TCP cookies. На подходе к серверу должна быть настроена система анализа трафика, которая будет проверять открытые соединения каждую минуту, так как практически во всех системах стандартные настройки этих параметров выставлены на отметке в 2 часа. Если подключен легитимный клиент, то ответ будет получен практически сразу же.

***Медленные атаки малого объема, так называемые Low and Slow.***

Для всех атак этого типа достаточно тех же способов, которые подходят для защиты от HTTP и SYN-флуда [6]. А так как атаки практически не нагружают канал всегда есть достаточное количество времени для того чтобы определить производится ли атака или же это клиенты с малой скоростью доступа. Если становится известно, что это атака достаточно блокировать соединения с теми клиентами у которых долго открыто соединение и невысок трафик.

На 3 уровне модели OSI:

*Smurf-атака (ICMP-флуд)*

##### Для того, чтобы защититься от ICMP-флуда нужно отключить ответы на запросы ICMP ECHO, причем сделать это можно как в настройках роутера, обеспечивающего подключение с глобальной сети, так и в настройках брэндмауэра на отдельных устройствах.

*Ping-флуд*

Лучшей защитой от ping-флуда является та же защита что и от ICMP-флуда, так как ping производится посредством протокола ICMP[1].

Так же есть общие рекомендации для защиты от DDoS-атак: дублирование каналов связи и серверных мощностей, а так же их разделение и рассредоточивание; предварительно выстроенная защита; фильтрация трафика; уклонение от атак, например смена доменного имени, IP-адреса, временное изменение записи DNS-серверов; использование аппаратно-программных комплексов защиты от DDoS-атак, например SecureSphere или Периметр, а так же в случае невозможности защиты или отражения атаки – встречная атака [8].

Для успешности любой атаки и, соответственно, наиболее качественной проверки выстроенной защиты необходимо рассмотреть методы усиления атак.

## **1.4** Методы усиления атак

В качестве методов усиления атак, так же, как и для любого алгоритма, выполняющегося в современных операционных системах, может использоваться многопоточность и многоядерность.

Наиболее простое использование многоядерности подразумевает перекладывание работы по распределению ядер на операционную систему, что возможно только в случае создания нескольких процессов. Таким образом необходимо создать несколько процессов с общими ресурсами для наилучшего взаимодействия и синхронизации. Этого можно достичь за счет использования разделенной памяти или пайпов. В рамках поставленной задачи является наилучшим использование разделенной памяти и специальной нотации позволяющей сократить количество ресурсов используемых в качестве накладных расходов для взаимодействия между процессами.

Для многопоточности в рамках поставленной задачи можно выделить намного больше целевых областей. Ее можно использовать для усиления рассылки пакетов, для ускорения сборки пакетов, для ускорения генерации данных.

Усиление рассылки пакетов с помощью многопоточности можно достигнуть за счет создания большого количества потоков, которые практически одновременно будут через разные порты посылать пакеты. Одновременно посылка пакетов не может быть произведена вследствие того, что сетевое устройство занимается каждым пакетом отдельно, соответственно не может одновременно обрабатывать более одного. Так же можно запускать одновременно несколько типов атак в рамках одного процесса, так как в основном типы атак отличаются только содержимым пакета, в качестве исключения можно выделить атаки с использованием приумножения DNS запросов, так как там целью рассылки является не атакуемое устройство или его сеть, а DNS сервера.

Многопоточность для ускорения сборки пакета может быть достигнута за счет одновременного копирования в разные участки памяти данных предварительно помещенных в кэш-память CPU. Таким образом исключается конфликт при доступе к участку памяти в котором находится предварительно сгенерированный эталонный заголовок пакета и, в некоторых случаях, тело пакета. Для некоторых типов атак имеет смысл генерировать только шаблонный заголовок, а тело генерировать для каждого пакета отдельно, так как иначе защита от такой атаки будет слишком простой. Для атак Low and Slow нет смысла генерировать даже шаблонный заголовок, так как сервер не даст открыть большой количество подключений с одного адреса, но в данном случае для каждого потока должен быть свой заголовок который генерируется 1 раз и не меняется на протяжении всей работы. Соответственно лучшим решением будет выделить некоторое время перед началом атаки для генерации необходимых данных и открытия достаточного количества подключений.

Ускорение генерации данных помимо уже рассмотренных случаев может быть необходимо для генерации случайных данных, что во многих разработках было главным слабым местом. В некоторых вариантах реализации приложения было несколько различных генераторов случайных данных ни один из которых не давал необходимой производительности. В данном случае лучшим выходом будет использование GPU, например, такое решение как CUDA позволяет использовать не только арифметические, но и побитовые операции, а так же содержит библиотеки FFT, BLAS и CuRand [7]. Использование большого количества SIMD процессоров, находящихся там позволяет генерировать данные с ускорением в 100-1548 раз относительно CPU. Таким образом на большом количестве компьютеров использующих видеокарты NVidia более поздних версий чем GT 8600 будет существенный прирост в производительности. Библиотека CuRand помещает сгенерированные данные в оперативную память компьютера, таким образом нет необходимости оптимизировать передачу данных между памятью видеокарты и оперативной памятью.

## **1.5** Основные рекомендации по защите от DDoS-атак

Обзор и анализ основных методов атак и защиты от них, а так же общих рекомендаций по защите позволяет составить алгоритм противодействия атакам в зависимости от уровня на котором они проводятся.

Для атак на 7 уровне модели OSI главными рекомендациями являются статистический анализ входящего и исходящего трафика, а так же мониторинг существующих подключений, что позволит среагировать своевременно и заблокировать подключения и пакеты атакующего.

Атаки, проводимые на 6 уровне модели OSI отразить сложнее, но, учитывая их узкую специализацию, наилучшим выходом будет пользоваться протоколами, которые не подвержены таким атакам.

На 4 уровне модели OSI сконцентрировано больше всего атак, и защита от них наиболее сложна вследствие их массивности. Для защиты от них наилучшим выходом будут дублирующие каналы и распределенные системы, а так же программно-аппаратные комплексы защиты.

Атаки 3 уровня достаточно сильно устарели, поэтому не представляют опасности. Для защиты от них достаточно минимальной донастройки роутеров и файерволов.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

В приложении Б представлена структура разрабатываемой системы. Каждый отдельно реализованный блок имеет свои задачи. Блоки связаны между собой, что обеспечивает единство всей системы. Связь между блоками осуществляется в большей степени за счёт передачи информации.

Система была разделена на следующие блоки:

1. Блок пользовательского интерфейса.
2. Блок сборки и рассылки пакетов.
3. Блок получения дополнительной информации.
4. Блок мониторинга.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.188 C1.

Рассмотрим функции блоков системы.

*Блок пользовательского интерфейса* представлен оконным приложением. Данный блок предоставляет пользователю возможность управления системой, а именно выбор необходимых настроек, таких как тип атаки, величина усиления атаки. Также данный блок сообщает пользователю о ходе процесса работы приложения. Кроме того, по всплывающим подсказкам так же показан основной способ защиты от данного типа атак. Данный модуль разработан на основе средств Windows Forms и, соответственно, платформы .NET.

*Блок сборки и рассылки пакетов* формирует заголовки всех необходимых протоколов для равномерного распределения нагрузки на атакующую машину, а так же тело пакета. В зависимости от выбранных настроек атаки запускаются разные внутренние блоки, так как для разных типов атаки пакеты отличается почти полностью, то данный блок разделяется на несколько блоков для обработки и формирования каждого типа пакетов с использованием единого интерфейса для удобства разработки. Данный блок реализован на языке C++, так как стандартные библиотеки на этом языке дают доступ к заголовкам пакетов, позволяя реализовать данную задачу оптимальным образом. Кроме того, данный блок необходим для оптимизации алгоритма на этапе рассылки, обеспечивает параллелизм и необходимую мощность отправки для любого типа атаки, используя максимум доступных и подходящих для решения этой задачи ресурсов компьютера.

*Блок усиления получение дополнительной информации* занимается сбором информации вносимой в заголовки пакета в случае если эта информация не задана вручную. Этот блок позволяет получить данные о подключении, в частности идентификатор текущей PPPoE сессии, устройство которое подключено к глобальной сети Интернет и иные параметры, которые возможно получить только при использовании сторонних библиотек требующих отдельной установки. Данный блок реализован на языке C++ так как сторонние библиотека WinPCap, позволяющие получить доступ к входящим и исходящим пакетам и информации в них, реализованы на этом языке, а так же потому что язык C++ обеспечивает максимальное быстродействие с минимальными накладными затратами на данном этапе работы программного средства.

*Блок мониторинга* проверяет доступность ресурса для внешних источников, так как канал атакующего на высоких уровнях усиления занят собственными исходящими пакетами, что не позволяет адекватно оценивать результат и плодотворность атаки и взаимодействия с защитными системами ресурса. Результаты мониторинга выводятся в окно пользовательского интерфейса вместе с данными о ходе атаки на локальном компьютере. Это позволяет точно оценить качество и результативность проделанной работы. Реализован данный модуль на платформе .NET, так как она имеет встроенный функционал для выполнения возложенной на блок функции, а так же уже используется в проекте, поэтому не требует дополнительных внешних ресурсов.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных алгоритмов, используемых в работе, модулей программы и рассмотрим их зависимости. А также проанализируем все переменные и функции, которые входят в состав кода программы.

Обобщенный алгоритм программы можно увидеть на схеме работы системы ГУИР.400201.188 ПД. Последовательно выполняются следующие действия: считывается информация о типе, мощности и цели атаки процессом GUI\_t1; данные формируются в строку, являющуюся аргументами запуска процессов, производящих атаку attackProcess; параллельно запускается модуль мониторинга цели, если она находится в сети Интернет с названием Monitoring. Для проекта было выбрано направление наибольшей модульности, модули были разделены в зависимости от их требований к внешним ресурсам и библиотекам.

Из всех этих модулей наибольшую алгоритмическую сложность представляет процесс, производящий атаку, так как на нем лежит главная задача проекта и в нем собрано несколько алгоритмов атак. Для модуля была выбрана процедурная парадигма вследствие использования низкоуровневых библиотек для наибольшей скорости работы и наименьших требований к внешним ресурсам.

## **3.1** Модуль атакующего процесса

Обобщенный алгоритм всех атак следующий: запускается необходимое количество потоков для получения достаточной мощности, в каждом из которых собираются заголовки пакетов разных уровней модели OSI в соответствии с запущенной атакой, при необходимости собирается тело пакета, отправка этого пакета с незначительным интервалом через независимый для каждого потока порт до окончания атаки.

В соответствии с этим алгоритмом было выбрано деление на функции, а так же структуры для формирования необходимых заголовков.

Следующие структуры были необходимы для создания всех заголовков, используемых при атаках:

* IPHeader – заголовок протокола IPv4, самого низкого доступного к редактированию уровня модели OSI, используется во всех типах атак;
* ICMPHeader – заголовок протокола ICMP, используемого в атаках типа Smurf;
* UDPHeader – заголовок протокола UDP, который используется при атаках типа DNS amplification и Fraggle;
* TCPHeader – заголовок протокола TCP, который используется в HTTP-флуд атаках, а так же в атаках типа TCP-SYN и RUDY;
* PseudoIPHeader – псевдозаголовок протокола IP, необходимый для формирования правильной контрольной суммы в заголовке TCP;
* DNSHeader – заголовок протокола DNS, используемый в данном проекте для реализации DNS amplification атак;
* PPPoEHeader – заголовок для пакетов отправляемых по PPPoE соединению, как наиболее распространенному типу соединений между сетями, необходим для всех типов атак проводимых вне локальной сети;
* Question – вспомогательная структура для формирования DNS пакета.

СтруктураIPHeader состоит из следующих полей:

* ip\_header\_len – длина заголовка в 32-битных словах, как правило равен 5;
* ip\_version – версия протокола IP;
* ip\_tos – тип обслуживания, необходим для разделения трафика на класы обслуживания;
* ip\_total\_length – полный размер пакета, включая заголовок, может иметь значения от 20 до 65535;
* ip\_id – уникальный ID пакета;
* ip\_frag\_offset – смещение фрагмента в 8-байтных словах от начала фрагментации, используется только если пакет был фрагментирован;
* ip\_more\_fragment – флаг, показывающий что данный пакет не последний;
* ip\_dont\_fragment – флаг, показывающий что данный пакет не фрагментирован;
* ip\_reserved\_zero – зарезервированный 0;
* ip\_frag\_offset1 – продолжение поля ip\_frag\_offset;
* ip\_ttl – время жизни пакета;
* ip\_protocol – переменная показывающая данные какого протокола более высокого уровня находятся в этом пакете;
* ip\_checksum – контрольная сумма, показывающая правильность пакета;
* ip\_srcaddr – IP адрес отправителя пакета;
* ip\_destaddr – IP адрес назначения пакета.

Структура ICMPHeader состоит из следующих полей:

* byType – тип ICMP сообщения;
* byCode – код ICMP сообщения;
* nChecksumH – старший байт контрольной суммы;
* nChecksumL – младший байт контрольной суммы;
* nId – уникальный идентификатор пакета;
* nSequence – номер последовательности;
* data[] – данные.

Структура UDPHeader состоит из следующих полей:

* udph\_srcport – порт из которого был отправлен пакет, не является обязательным если на пакет не предусмотрено получение ответа;
* udph\_destport – порт назначения пакета;
* udph\_len – длина данных в пакете в байтах включая заголовок;
* udph\_chksum – контрольная сумма пакета, как правило не проверяется и не является обязательной вследствие особенностей протокола UDP.

Структура TCPHeader состоит из следующих полей:

* sourcePort – порт из которого был отправлен пакет;
* destinationPort – порт назначения пакета;
* th\_seq – порядковый номер пакета;
* th\_ack – порядковый номер предыдущего полученного пакета;
* th\_x2 – зарезервировано;
* th\_off – длина заголовка в 4-байтных словах, может быть от 5 до 15;
* th\_flags – поле флагов;
* th\_win – количество байт которое способен принять отправитель в качестве ответа на данный пакет;
* th\_sum – контрольная сумма, рассчитывается с учетом псевдозаголовка, так как протоколы более высокого уровня не имеют доступа к заголовкам протоколов более низких уровней;
* th\_urp – номер октета в котором заканчиваются важные данные, необходим только при установленном флаге отправки внеполосных данных;
* options[3] – опции, в данный момент почти всегда включают 2 байта NOP(no operation) резервирующих некоторое количество системных ресурсов для последующих пакетов;

Структура PseudoIPHeader состоит из следующих полей:

* sourceIP – IP адрес отправителя;
* destinationIP – IP адрес назначения;
* reservedZero – зарезервированный нуль;
* protocol – переменная показывающая данные какого протокола в этом пакете;
* TCPLength – длина пакета включая заголовок TCP.

Структура DNSHeader состоит из следующих полей:

* id – идентификационный номер пакета;
* rd – флаг показывающий необходимость рекурсии;
* tc – флаг показывающий что пакет обрезан, то есть для протокола UDP пакет превысил размер в 512 байт;
* aa – флаг показывающий что ответ получен из заслуживающего доверие источника;
* opcode – код операции;
* qr – флаг показывающий является ли этот пакет запросом к серверу или ответом;
* rcode – код ответа;
* cd – флаг показывающий что проверка отключена;
* ad – флаг показывающий аутентичность данных, игнорируется сервером;
* z – зарезервированный нуль;
* ra – флаг показывающий что рекурсия возможна, в запросах всегда устанавливается в 0;
* q\_count – количество вопросов в пакете, практически всегда равен 1;
* ans\_count – количество ответов, в запросе всегда устанавливается 0;
* auth\_count – количество авторитетных ответов, в запросе всегда устанавливается в 0;
* add\_count – количество дополнительной информации, в запросе всегда устанавливается в 0.

Структура PPPoEHeader состоит из следующих полей:

* typeVer – тип и версия PPPoE соединения, как правило устанавливается в 0x11;
* data – данные, как правило устанавливается в 0;
* sessionID – идентификационный номер текущей PPPoE сессии;
* length – длина данных в пакете в байтах включая поле PPPProtocol;
* PPPProtocol – протокол данных содержащихся в пакете, для IP равен 0х21.

Далее стоит подробно рассмотреть алгоритм каждой атаки и разработанные в соответствии с ним функции.

Для атаки типа *Smurf* алгоритм следующий:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным адресу жертвы и широковещательным адресом назначения, а так же с протоколом равным IPPROTO\_ICMP;
3. Создание ICMP заголовка с фиксированным содержимым длиной 1024 и заранее рассчитанной для него контрольной суммой для ускорения работы;
4. Копирование полученных данных в цельный массив байт;
5. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
6. Разрешение широковещательной рассылки и посылки данных в заголовке;
7. Бесконечно посылать этот пакет с небольшим интервалом.

Для атаки типа *Fraggle* алгоритм во многом схож:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным адресу жертвы и широковещательным адресом назначения, а так же с протоколом равным IPPROTO\_UDP;
3. Создание UDP заголовка с фиксированным содержимым длиной 480 байт, так как суммарный размер пакета не должен быть более 508 байт, для того чтобы все устройства могли его принять и обработать, а так же с исходящими и целевыми портами равными 7, что является портом эхо запроса;
4. Копирование полученных данных в цельный массив байт;
5. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
6. Разрешение широковещательной рассылки и посылки данных в заголовке;
7. Бесконечно посылать этот пакет с небольшим интервалом;

Для TCP-SYN атаки алгоритм следующий:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным произвольному адресу и адресом цели в качестве адреса назначения, а так же с протоколом равным IPPROTO\_TCP;
3. Создание TCP заголовка с исходящим портом, рассчитываемым из номера процесса и потока, целевым портом, заданным в аргументах командной строки и флагом SYN равным 1;
4. Создание псевдозаголовка IP для расчёта контрольной суммы с теми же данными что и заголовок IP;
5. Расчет контрольной суммы;
6. Копирование полученных данных в цельный массив байт;
7. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
8. Бесконечно посылать этот пакет с небольшим интервалом.

Алгоритм реализации DNS amplification атаки несколько сложнее:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным адресу жертвы и адресом назначения равным адресу неправильно сконфигурированного DNS сервера, а так же с протоколом равным IPPROTO\_UDP;
3. Создание UDP заголовка с целевым и исходящим портами равными 53, так как это общепринятый порт службы DNS;
4. Создание заголовка DNS с требованием рекурсии и одним вопросом;
5. Преобразование имени сайта к формату обрабатываемому DNS службами;
6. Создание окончания пакета с типом запроса ANY, который запрашивает все имеющиеся данные;
7. Копирование полученных данных в цельный массив байт;
8. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
9. Бесконечно посылать этот пакет с небольшим интервалом.

Для атаки типа HTTP flood был использован следующий алгоритм:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным произвольному адресу и адресом цели в качестве адреса назначения, а так же с протоколом равным IPPROTO\_TCP;
3. Создание TCP заголовка с исходящим портом, рассчитываемым из номера процесса и потока, целевым портом равным 80, как общепринятым для HTTP;
4. Создание псевдозаголовка IP для расчёта контрольной суммы с теми же данными что и заголовок IP;
5. Создание запроса HTTP
6. Расчет контрольной суммы;
7. Копирование полученных данных в цельный массив байт;
8. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
9. Бесконечно посылать этот пакет с небольшим интервалом.

Для TCP-SYN атаки алгоритм следующий:

1. При необходимости создание заголовка PPPoE с текущими значениями идентификатора сессии и размера пакета;
2. Заполнение заголовка IP с исходящим адресом равным произвольному адресу и адресом цели в качестве адреса назначения, а так же с протоколом равным IPPROTO\_TCP;
3. Создание TCP заголовка с исходящим портом, рассчитываемым из номера процесса и потока, целевым портом равным 80, как общепринятым для HTTP;
4. Создание псевдозаголовка IP для расчёта контрольной суммы с теми же данными что и заголовок IP;
5. Создание запроса HTTP с 1 байтом информации;
6. Расчет контрольной суммы;
7. Копирование полученных данных в цельный массив байт;
8. Создание сырого сокета и открытие порта в зависимости от порядкового номера процесса и потока;
9. Установка опции сокета SO\_KEEPALIVE;
10. Бесконечно посылать этот пакет с небольшим интервалом.

В соответствии с этими алгоритмами можно выделить следующие функции:

* generateIPHeader() – генерация заголовка IP;
* generateTCPHeader() – генерация заголовка TCP;
* generateUDPHeader() – генерация заголовка UDP;
* generateDNSHeader() – генерация заголовка DNS;
* generateICMPHeader() – генерация заголовка ICMP;
* completeDNSUDPPacket() – полная сборка DNS пакета включая заголовки и запрос;
* generatePPPoEHeader() – сгенерировать заголовок PPPoE;
* generatePseudoIPHeader() – сгенерировать псевдозаголовок IP для расчета контрольной суммы в заголовках более высоких уровней;
* Initialize() – инициализация WinSock;
* UnInitialize() – отключение WinSock;
* generateBroadcastIPAddress()–генерация широковещательного адреса из исходного;
* setBcastAndDataInHeader() – установка опций сокета разрешающих широковещательную рассылку и посылку данных в заголовке;
* generateCheckSum() – расчет контрольной суммы по входному потоку байт;
* changetoDnsNameFormat() – преобразование строки к формату необходимому для обработки DNS сервером;
* smurfAttackThread() – функция выполняющая атаку типа Smurf в отдельном потоке;
* fraggleAttackThread() – функция выполняющая атаку типа Fraggle в отдельном потоке;
* TCPSYNAttackThread() – функция выполняющая атаку типа TCP-SYN в отдельном потоке;
* RUDYAttackThread() – функция выполняющая атаку типа RUDY в отдельном потоке;
* HTTPFloodAttackThread() – функция выполняющая атаку типа HTTP Flood в отдельном потоке;
* DNSAmplificationAttackThread() – функция выполняющая атаку типа DNS amplification в отдельном потоке;
* startDNSAmplificationAttack() – запуск атаки типа DNS amplification в заданное количество потоков;
* startTCPSYNAttack() – запуск атаки типа TCP-SYN в заданное количество потоков;
* startSmurfAttack() – запуск атаки типа Smurf в заданное количество потоков;
* startRUDYAttack() – запуск атаки типа RUDY в заданное количество потоков;
* startHTTPFloodAttack() – запуск атаки типа HTTP Flood в заданное количество потоков;
* startFraggleAttack() – запуск атаки типа Fraggle в заданное количество потоков.

Так же в этом приложении есть несколько констант и глобальных переменных:

* targetIPAddress – IP адрес цели, является глобальной переменной так как задается из командной строки и в процессе работы не меняется, а так же нужен практически во всем проекте;
* targetPort – порт назначения, является глобальной переменной так как задается из командной строки и в процессе работы не меняется, а так же нужен практически во всем проекте;
* PPPoESessionID – идентификатор текущей PPPoE сессии, является глобальной переменной так как задается из командной строки и в процессе работы не меняется, а так же нужен практически во всем проекте, в случае если равен 0 игнорируется в процессе работы;
* procID – идентификатор данного процесса среди запущенных, необходим для использования ресурсов незанятых другими процессами производящими атаку;
* ICMPDATASIZE – размер данных включенных в ICMP пакет;
* UDPDAtaSIZE – размер данных отправляемых в UDP пакете;
* TH\_FIN – флаг TCP пакета означающий конец соединения, равен 0x01;
* TH\_SYN – флаг TCP пакета означающий инициализацию соединения, равен 0x02;
* TH\_RST – флаг TCP пакета означающий что соединение необходимо сбросить, равен 0x04;
* TH\_PUSH – флаг TCP пакета означающий что полученные ранее данные нужно передать в целевое приложение, равен 0x08;
* TH\_ACK – флаг TCP пакета означающий подтверждение установки соединения, равен 0x10;
* TH\_URG – флаг TCP пакета означающий что в пакете находятся внеполосные данные, равен 0x20;
* TH\_ECE – флаг TCP пакета означающий что данный узел способен на явное уведомление о перегрузках в сети, равен 0x40;
* TH\_CWR – флаг TCP пакета означающий что был получен пакет с флагом TH\_ECE и окно уменьшено, равен 0x80;
* TH\_FLAGS – установка всех вышеперечисленных флагов;
* DEFAULTPPPOETYPEVER – тип и версия PPPoE пакета по умолчанию, равен 0x11;
* DEFAULTPPPOEDATA – стандартные данные PPPoE включенные в заголовок PPPoE пакета, равен 0;
* DEFAULTPPPOEPROTOCOL – используемый по умолчанию протокол IP, равен 0х0021.

**3.2** Модуль интерфейса

Данный модуль выполняет сразу 2 задачи: сбор данных необходимых для запуска атаки и управление остальными модулями. Такие данные о атаке как адрес цели, порт цели, количество процессов, количество потоков, тип атаки, необходимость использования PPPoE соединения задаются в главном окне. Так же в главном окне краткие советы по защите от каждого типа атак.

В случае необходимости использования PPPoE соединения запускается модуль получающий идентификатор текущей PPPoE сессии. Далее запускается модуль мониторинга. После этого запускается необходимое количество модулей атаки и создается список этих процессов для дальнейшего управления ими.

Данный модуль реализован на .NET C++ как на наиболее распространенной и широкодоступной платформе реализации интерфейса. Но этот модуль отделен от остальных так как для запуска атаки интерфейс не обязателен и несущественен, а так же является сравнительно требовательным.

В единственном классе в данном приложении есть следующие поля:

* attackProcesses – список созданных процессов производящих атаку, необходим для остановки процессов по требованию, а так же просмотра информации о них в случае необходимости;
* selectedRadioButton – индекс выбранного в главном окне типа атаки;
* processCount – количество процессов для создания;
* threadsCount – количество потоков необходимое для создания в каждом атакующем процессе;
* targetPort – целевой порт отправляемый в каждый атакующий процесс;
* radioButton1 – пункт меню включающий тип атаки HTTP-флуд;
* radioButton2 – пункт меню включающий тип атаки RUDY;
* radioButton3 – пункт меню включающий тип атаки DNS усиление;
* radioButton4 – пункт меню включающий тип атаки Smurf;
* radioButton5 – пункт меню включающий тип атаки UDP-Fraggle;
* radioButton6 – пункт меню включающий тип атаки TCP-SYN;
* textBox1 – поле для ввода количества создаваемых процессов атаки;
* textBox2 – поле для ввода количества создаваемых потоков в каждом атакующем процессе;
* textBox3 – поле для ввода адреса цели или же его доменного имени;
* textBox4 – поле для ввода целевого порта;
* button1 – кнопка запускающая атаку;
* button2 – кнопка останавливающая атаку;
* progressBar1 – полоса загрузки;
* label1 – текстовое поле с пояснением для textBox1;
* label2 – текстовое поле с пояснением для textBox2;
* label3 – текстовое поле с пояснением для textBox3;
* label4 – текстовое поле с пояснением для textBox4;
* label5 – текстовое поле с советами по защите от каждого типа атак;
* checkBox1 – поле для разрешения или запрета на использование PPPoE соединения.

Так же в этом классе находятся следующие методы:

* InitializeComponent() – создание всех элементов управления и текстовых полей;
* button1\_Click() – обработчик нажатия на кнопку button1;
* runMonitoring() – запуск модуля мониторинга;
* getPPPoESessionID() – получение идентификатора текущей PPPoE сессии с помощью отдельного модуля;
* button2\_Click – обработчик нажатия на кнопку button2;
* radioButton1\_CheckedChanged() – обработчик нажатия на кнопку radioButton1;
* radioButton2\_CheckedChanged() – обработчик нажатия на кнопку radioButton2;
* radioButton3\_CheckedChanged() – обработчик нажатия на кнопку radioButton3;
* radioButton4\_CheckedChanged() – обработчик нажатия на кнопку radioButton4;
* radioButton5\_CheckedChanged() – обработчик нажатия на кнопку radioButton5;
* radioButton6\_CheckedChanged() – обработчик нажатия на кнопку radioButton6;

При запуске приложения все поля пустые и ожидают ввода, после нажатия кнопки button1 запускается метод button1\_Click, который запускает модуль получения идентификатора текущей PPPoE сессии в случае если checkBox1 был отмечен. После этого запускается модуль мониторинга и необходимое количество модулей атаки с аргументами командной строки, которые формируются в этом же методе. После этого кнопка button1 становится недоступна.

По нажатию на кнопку button2 запускается метод button2\_Click, который останавливает процессы если они еще не были остановлены и разрешает нажатие на кнопку button1 и, соответственно, старт новой атаки.

## **3.3** Модуль определения идентификатора текущей PPPoE сессии

Данный модуль реализован в процедурной парадигме чтобы минимизировать количество кода и ускорить его работу. Он был выделен так как он использует библиотеку WinPCap и не будет работать на компьютерах где этой библиотеки нет. Так же запуск этого модуля не нужен в случае, когда PPPoE соединение не используется. Таким образом функционал проекта по-прежнему сохраняется.

Данный модуль состоит из единственной функции, реализующей следующий алгоритм:

* Определение не виртуального устройства, подключенного в сети Интернет;
* Отправка минимального запроса к DNS серверу;
* Фильтр пакетов по параметрам PPPoE соединение и 53 порт;
* Чтение идентификатора текущей PPPoE сессии из отфильтрованного пакета;
* Отправка полученного числа в качестве возвращаемого значения.

Единственная функция этого модуля main() выполняет этот алгоритм и возвращает число находящееся в 16-17 байтах полученного пакета.

## **3.4** Модуль мониторинга

Данный модуль так же, как и модуль интерфейса был реализован с помощью технологии .NET WinForms и представляет собой встроенный браузер со страничкой, проверяющей доступность цели и время ее отклика в различные точки на планете.

Единственный класс Form1 содержит следующий метод:

* InitializeComponent – создание всех компонентов окна с предопределенными параметрами.

Так же в этом классе содержится одно поле:

* webBrowser1 – встроенный веб браузер открывающий страничку показывающую доступность ресурса находящегося в сети Интернет.

Реализация модуля мониторинга ресурса в пределах локальной сети не представляется возможной, так как для реализуемых типов атак задействованы те же системные ресурсы что и при мониторинге сетевого ресурса в произвольный момент времени.

# 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

При разработке данного проекта, осуществляющего тестирование надежности сетевых ресурсов в разрезе доступности, было выбрано направление доступности для любых пользователей. В рамках этого подразумевается наибольшая гибкость программы и приспособленность ее к любым условиям работы. Лучшим выходом в этой ситуации является разделение проекта на несколько программ в зависимости от требований к системным ресурсам чтобы в минимальной функциональности проект мог работать даже на операционной системе Windows без дополнительных библиотек или доступа в сеть Интернет.

## **4.1** Модуль атакующего процесса

Самым сложным модулем, выполняющим непосредственно атаку является attackProcess.exe, среди используемых им библиотек все являются встроенными в любую операционную систему семейства Windows моложе 98. В частности, работа с потоками в нем производится с использованием WinAPI, работа с сетью с использованием WinSock2, все функции, использующие тот или иной функционал WinAPI, отделены друг от друга, что позволяет в случае необходимости в краткие сроки переработать данный проект для операционных систем семейства Unix. Исходные данные передаются в командной строке, что позволяет свести взаимодействие пользователя с программой к минимуму с одной стороны и дать программе взаимодействовать с любыми другими программами не используя платформозависимых решений с другой. Для этого необходимо знать разработанную нотацию, а так же способ перевода IP адреса из общепринятого формата XXX.XXX.XXX.XXX к формату используемому в WinSock2.

Нотация была выбрана максимально краткая и гибкая. В командную строку передается 6 чисел в следующем порядке:

* Уникальный идентификатор текущего процесса, если будет несколько процессов с одним идентификатором, то все, созданные после первого, не получат доступ к системным ресурсам используемым в процессе с таким идентификатором;
* Количество потоков для создания в этом процессе, IP адрес цели в формате используемом в WinSock;
* Целевой порт, если для атаки необходимого типа он не принципиален;
* Тип атаки, где 1 – запуск атаки типа RUDY, 2 – запуск атаки типа HTTP Flood, 3 – запуск атаки типа DNS amplification, 4 – запуск атаки типа Smurf, 5 – запуск атаки типа Fraggle, 6 – запуск атаки типа TCP-SYN;
* Идентификатор текущей PPPoE сессии если нужен доступ в сеть Интернет с использованием PPPoE соединения, в случае если необходима атака на ресурс в локальной сети это значение должно быть нулевым.

Идентификатор текущей PPPoE сессии можно получить используя сторонние программы, показывающие содержимое пакета, например WireShark, или разработанный в рамках проекта модуль определения идентификатора текущей PPPoE сессии sessionIDPPPDefiner.exe.

Для преобразования IP адреса из общепринятого формата ХХХ.ХХХ.ХХХ.ХХХ в формат, используемый в программе необходимо проделать следующие действия:

1. Перевести десятичное число в каждом октете в шестнадцатеричную систему счисления;
2. Выполнить конкатенацию полученных чисел в строку в порядке следования в исходной записи;
3. Перевести получившееся число в десятичную систему счисления;

Более краткий вариант этого алгоритма:

Если адрес записан в форме AAA.BBB.CCC.DDD вычислить значение выражения AAA\*256^3+BBB\*256^2+CCC\*256+DDD. Полученное число и будет искомым.

При поступлении в программу данные аргументы преобразуются в числовую форму, причем адрес в беззнаковое четырехбайтное целое число, а порт и идентификатор текущей PPPoE сессии в двухбайтовое целое число. Эти значения сохраняются в глобальных переменных по причинам, описанным в функциональном проектировании.

Далее в зависимости от типа атаки запускаются функция запускающая необходимый тип атаки в необходимое количество потоков, для наибольшего быстродействия используются WinAPI функция создания потоков CreateThread, но появляется платформозависимость. В случае переработки проекта под операционные системы семейства Unix эти функции необходимо заключить в директивы препроцессора #ifdef \_win32\_ и добавить директивы и код для необходимой операционной системы.

В функции выполняющиеся в потоке в качестве параметра передается их уникальный номер selfID с использованием которого позже вычисляется номер порта открываемого для текущего потока по формуле selfID+(procID\*1000), где procID это идентификатор текущего процесса. Таким образом лучшим можно считать создание процессов начиная с их идентификатора равного 1 и не более 64, для того чтобы не занять порты входящие в первую тысячу, зарезервированную под системные нужды и открыть заранее предопределенный порт, а не некоторый порт полученный с учетом переполнения двухбайтового целого беззнакового числа. Более того не стоит создавать количество процессов большее чем количество ядер в процессоре, чтобы не распылять мощность на накладные расходы, так же число потоков суммарно во всех процессах не должно быть более 50, так как большинство сетевых карт не имеет возможности обрабатывать создаваемое количество пакетов, таким образом крайне сильно нагружая компьютер. Оптимальной нагрузкой достаточной для стабильной работы будет создание и отправка 2000-5000 пакетов в секунду.

В каждом запущенном потоке выполняется функция реализующая требуемый тип атаки, в каждой такой функции есть создание и открытие сырого сокета с помощью функций библиотеки WinSock2 SOCKET(), htons() и inet\_addr(), что так же в случае переработки необходимо заключить в директивы препроцессора #ifdef \_win32\_ и добавить директивы и код для необходимой операционной системы. Но данный вариант работы является наиболее оптимальным с точки зрения системных ресурсов и затрачиваемого времени.

В зависимости от типа атаки в пакет добавляются различные заголовки и в соответствии с их содержанием в функции, собирающие их отправляются различные параметры. Для сборки заголовка IP в функцию передаются адреса которые необходимо внести в поля адреса отправки и адреса назначения, протокол и флаг фрагментации пакета. Для сборки заголовка UDP в функцию передаются порт назначения и порт отправки, а так же размер данных в пакете. Для сборки пакета TCP так же отправляются порты назначения и отправки, но вместо размера пакета передается флаг SYN. Для сборки заголовков DNS и ICMP в функции не передаются параметры, так как в них нет необходимости. Но для сборки полного пакета DNS в функцию помимо адресов назначения и источника передается идентификатор PPPoE сессии и переменная в которую будет записан итоговый размер пакета.

При сборке каждого пакета с использованием заголовков IP и TCP используется контрольная сумма, являющаяся двухбайтовым беззнаковым числом которая рассчитывается по всему пакету включая заголовки допуская что на момент расчета контрольная сумма текущего заголовка равна 0 по следующему алгоритму:

1. Сумма текущей контрольной суммы с последующими 2 байтами пакета;
2. Если контрольная сумма больше чем 65535, то отнять от нее 65535, добавить 1 и перейти к пункту 1;
3. Если пакет кончился – инвертировать значение текущей контрольной суммы иначе перейти к пункту 1;
4. Поместить результирующее число в необходимое место в заголовке пакета;

После оптимизации данный алгоритм был реализован следующим образом:

unsigned short generateCheckSum(unsigned short \*ptr,int nbytes)

{

register int nleft = nbytes;

register u\_short \*w = ptr;

register int sum = 0;

u\_short answer = 0;

while (nleft > 1) {

sum += \*w++;

nleft -= 2;

}

if (nleft == 1) {

\*(u\_char \*)(&answer) = \*(u\_char \*)w ;

sum += answer;

}

sum = (sum >> 16) + (sum & 0xffff);

sum += (sum >> 16);

answer = ~sum;

return(answer);

}

В случае Smurf атаки используется константная заранее рассчитанная контрольная сумма в заголовке ICMP пакета, так как содержимое этого пакета известно заранее и занимает сравнительно большое количество времени из-за его больших размеров. Для пакетов, использующих протокол UDP, то есть для Fraggle и DNS amplification атак, контрольная сумма не является обязательной, поэтому выставляется константным значением 0xFFFF.

Для формирования DNS пакета используемого в атаках типа DNS amplification необходимо формирование вопроса к серверу, то есть перевод адреса искомого ресурса в форму обрабатываемую DNS сервером, например, www.google.com преобразуется в 3www6google3com. Это преобразование реализуется следующей функцией:

void сhangetoDnsNameFormat(unsigned char\* dns,unsigned char\* host)

{

int lock = 0 , i;

strcat((char\*)host,".");

for(i = 0 ; i < strlen((char\*)host) ; i++)

{

if(host[i]=='.')

{

\*dns++ = i-lock;

for(;lock<i;lock++)

{

\*dns++=host[lock];

}

lock++;

}

}

\*dns++='\0';

}

Для посылки сформированного пакета используется функция sendto()так как она не требует установки соединения, что позволяет реализовывать подмену IP адреса в пакете, установку SYN флага вручную и иные необходимые при атаках параметры пакетов.

Так же в реализованных функциях помимо легкости переработки были добавлены возможности по добавлению новых типов атак не внося изменений в существующий код, например, отдельным параметром выделены некоторые флаги, такие как фрагментированность пакета, для реализации атаки переполняющей буфер принимающей стороны сходно с TCP-SYN атакой.

Таким образом разработан предельно гибкий и независимый модуль, который, тем не менее, может быть объединен в единый проект с другими модулями вне зависимости от того являются ли они отдельным приложениями или .bat скриптами. Так же этот модуль не требует никаких внешних библиотек и может быть запущен на «голой» операционной системе семейства Windows, причем если запускать атаки вручную, без использования модуля интерфейса, то можно запустить одновременно несколько типов атак или же атак на разные цели, однако в этом случае необходимо правильно рассчитать мощность атак чтобы не перегрузить процессор, память или сетевую карту.

## **4.2** Модуль интерфейса

Данный модуль был выделен в отдельное приложение так как для его запуска и работы необходима платформа .NET начиная от версии 2.0 и запуск этого модуля не является обязательным для запуска атаки, однако существенно повышает удобство, а так же содержит краткие советы по защите от реализованных типов атак для начинающих пользователей.

Данный модуль был реализован на WinForms, а не на более современном WPF также содержащимся в платформе .NET так как требует меньшее количество ресурсов и не требует более новых версий платформы .NET. Кроме того, сравнительная простота и удобство программирования вкупе с возможностью контролировать каждый аспект программы, является более важным критерием при выборе средств разработки, чем визуальная привлекательность, так как большая востребованность данного приложения достигается в первую очередь функциональным наполнением, а не приятным внешним видом. Так же вследствие модульности приложения можно разработать удобный интерфейс на любой платформе, которая способна отправлять команды в командную строку операционной системы.

Для выбора типа атаки был выбран элемент управления типа RadioButton, который позволяет выбрать только один элемент этого типа на текущей панели, для того чтобы при запуске атаки не было неоднозначности или расхождения ожиданий и реальности. Для включения же взаимодействия по PPPoE соединению был выбран CheckBox, который может быть выбран независимо от остальных элементов, для того чтобы каждый реализованный тип атаки можно было запускать как в локальной сети, так и в глобальную сеть Интернет.

По нажатию кнопки button1 проверяется установлен ли пункт разрешающий использование PPPoE соединения и в случае если да, то запускается модуль мониторинга и модуль определения идентификатора текущей PPPoE сессии. После этого запускается необходимое количество процессов атаки. Для этого используется класс Process как наиболее подходящий для данной цели. В пол этого класса заносится строка параметров командной строки, а так же указание что процессы нужно запускать от имени администратора, так как приложение использует сырые сокеты к которым нет доступа у пользователей и программ не обладающих привилегиями администратора. Кроме того, необходимо указать что искомая программа находится в том же каталоге что и модуль интерфейса, в противном случае будет произведена попытка запуска указанного приложения из каталога C:\windows\system32. После этого кнопка button2 становится активной и, соответственно, атаку можно остановить, а кнопка button1 становится неактивной чтобы не создавать конфликтов между атакующими процессами.

По нажатию кнопки button2 производятся обратные действия, то есть удаляются созданные процессы с небольшим интервалом необходимым на их удаление, запрещается кнопка button2 и разрешается кнопка button1.

Таким образом получается, что наибольшую гибкость проект проявляет при использовании опытным пользователем, так как через интерфейс можно одновременно запустить только один тип атаки. Действия же, выполняемые этим модулем максимально абстрагированы от платформы и остальных приложений что позволяет пользоваться проектом даже при отсутствии платформы .NET.

## **4.3** Модуль определения идентификатора текущей PPPoE сессии

Данный модуль был выделен в отдельное приложение так как использует библиотеку WinPCap, которая достаточно редко установлена на компьютере и необходимость ее установки может отпугнуть пользователя.

Алгоритм реализуемый в данном приложении достаточно прост чтобы быть умещенным в одну функцию без излишней громоздкости кода и состоит в следующем:

1. Инициализация WinPCap;
2. Нахождение всех сетевых устройств на текущем компьютере и доступных для обмена данными;
3. Поиск устройства по которому идет передача данных по PPPoE соединению, как правило это устройства компаний Broadcom или Atheros, соответственно из списка выбирается необходимое устройство;
4. Определение подключения данного устройства;
5. Подключение к данному устройству;
6. Сборка фильтра пакетов с параметрами «использование PPPoE» и «порт 53»;
7. Установка этого фильтра;
8. Отправка запроса к DNS серверу;
9. Поиск пакета с ответом от DNS сервера по установленному фильтру;
10. Отключение WinPCap;
11. Вернуть значение, находящееся в 16-17 байтах полученного пакета.

Программно этот алгоритм реализуется следующим образом:

1. Создание всех переменных;
2. pcap\_findalldevs\_ex("rpcap://",nil,alldevs,errbuf);
3. while (strcmp((cur->description),"Network adapter 'Microsoft' on local host")==0||strcmp((cur->description),"Network adapter 'Oracle' on local host")==0 ||strcmp((cur->description),"Network adapter 'MS NDIS 6.0 LoopBack Driver' on local host")==0) cur = cur->next;
4. pcap\_lookupnet(dev, &net, &mask, errbuf);
5. handle = pcap\_open\_live(dev, BUFSIZ, 1, 1000, errbuf);
6. pcap\_compile(handle, &fp, filter\_exp, 0, net);
7. pcap\_setfilter(handle, &fp);
8. system("nslookup google.by");
9. packet = pcap\_next(handle, &header);
10. pcap\_close(handle);
11. return packet[16]\*256+packet[17];

Таким образом модуль получает идентификатор текущей PPPoE сессии, а в случае если не удалось найти нужный пакет в программу-родитель возращается нуль, таким образом запуская атаку без использования PPPoE соединения.

## **4.4** Модуль мониторинга

Данный модуль был выделен в отдельное приложение несмотря на то что реализован на .NET WinForms так же как и модуль интерфейса потому что требует подключения к интернету и работе в несколько потоков, что требует излишних затрат на реализацию в качестве отдельного блока в модуле интерфейса. Кроме того, его использование необязательно и может быть контрпродуктивным в случае высокой мощности запущенной атаки, так как требует часть ресурсов сетевой карты которая нагружена до предела атакующими пакетами. Лучшим вариантом мониторинга является использование отдельного устройства с независимым соединением с сетью в которой производится атака, будь то локальная сеть или глобальная сеть Интернет.

Вследствие вышеперечисленных причин модуль состоит из одного элемента WebBrowser с предопределенной стартовой страницей, чего достаточно в рамках поставленной задачи.

# 5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование программного обеспечения – процесс исследования программного обеспечения с целью получения информации о качестве продукта. Тестирование является одним из важных этапов разработки, поскольку при написании программного кода невозможно предусмотреть все случаи, и ошибки в работе программы неизбежны. Кроме того, возможно появление ошибок в случае внесения дополнительного функционала в программу даже в уже отлаженном рабочем коде.

Тестирование программы направлено на доказательства её соответствия заявленным требованиям. Таким образом, основной целью тестирования является выделение несоответствия функционирования программы заявленным требованиям и устранение их, для повышения качества программного продукта.

Тестирование программы проводилось в два этапа:

* поэтапное тестирование отдельно каждого модуля в процессе написания программного кода;
* полное тестирование программы после окончания процесса написания программного кода.

Оба этапа являются достаточно важными, ни один из них невозможно исключить. Например, без модульного тестирования, при анализе работы программы в целом, будет происходить достаточное количество сбоев, выявить и локализовать которые может оказаться достаточно сложным заданием, в то время как при анализе работы одного модуля неисправность оказывается достаточно очевидной. И обратный случай, работоспособность каждого компонента в отдельности не гарантирует корректное поведение всей программы в целом.

Тестирование программы проводилось на следующих компьютерах:

* Intel Core I7-4702HQ 4 ядра по 3,4 ГГц, оперативная память 8Гб, видеокарта nVidia GT 750M. Операционная система Windows 8.1 Professional x64;
* Intel Core i5-4200 4 ядра по 2,49 ГГц, оперативная память 8 ГБ, видеокарта nVidia GeForce GTX 950M. Операционная система Windows 8.1 x64.

Тестирование каждого модуля проводилось как отдельно, так и в сочетании с другими модулями. Каждая функция, добавленная в любой модуль тестировалась преимущественно мануально, однако в различных сочетаниях с другими модулями или без оных.

Как говорилось ранее в модули были разделены в зависимости от необходимых внешних ресурсов, соответственно тестирование проводилось на операционных системах разной конфигурации и с различными установленными библиотеками и платформами.

Для обнаружения ошибок, которые возможно могли появиться на этапе программирования, проводилось функциональное тестирование. Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям. В зависимости от степени доступа к коду системы можно выделить два типа функциональных испытания: тестирование black box (черный ящик) – проведение тестирования без доступа к коду системы и тестирование white box (белый ящик) – тестирование с доступом к коду системы. Тестирование функциональности может проводиться в двух аспектах: требования и бизнес-процессы. Тестирование в перспективе «требования» используют спецификацию функциональных требований к системе как основу для дизайна тестовых случаев (Test Cases). В этом случае необходимо сделать список того, что будет тестироваться, а что нет, приоритезировать требования на основе рисков, а на основе этого приоритезировать тестовые сценарии. Это позволяет сфокусироваться и не упустить при тестировании наиболее важный функционал. Тестирование в перспективе «бизнес-процессы» используют знание этих самых бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этой перспективе тестовые сценарии, как правило, основываются на случаях использования системы (Use cases). Тестирование уже готового программного модуля проводилось сразу несколькими пользователями на разных компьютерах с разной производительностью. Функциональные тесты, проведенные над каждым программным модулем, представлены в таблице 5.1.

Как видно из скриншота модуля интерфейса (см рисунок 5.1) полей в которые можно вводить неверные значения 4. Кроме того 2 кнопки которые так же должны вести себя корректно при вводе некорректных значений.

В процессе разработки был выбран следующий способ защиты от большинства ошибок неправильного ввода: разрешено нажатие только на 1 кнопку в единичный момент времени, таким образом нельзя запустить атаку в процессе атаки или же остановить атаку, когда она не проводится, а так же кнопка «Start» не реагирует на нажатие в случае неправильного ввода.

Модуль атакующего процесса взаимодействует с пользователем и окружением только при запуске и остановке атаки, следовательно, главными проблемами помимо некорректного ввода является результативность непосредственно атак.

Для модуля мониторинга никаких тестов кроме выполнения его основной и единственной задачи нет и, вследствие отсутствия полей ввода и взаимодействия с пользователем кроме единственного, присутствующего на странице, быть не может.

У модуля определения идентификатора текущей PPPoE сессии так же может быть только одна проблема вследствие того, что он не взаимодействует с пользователем.

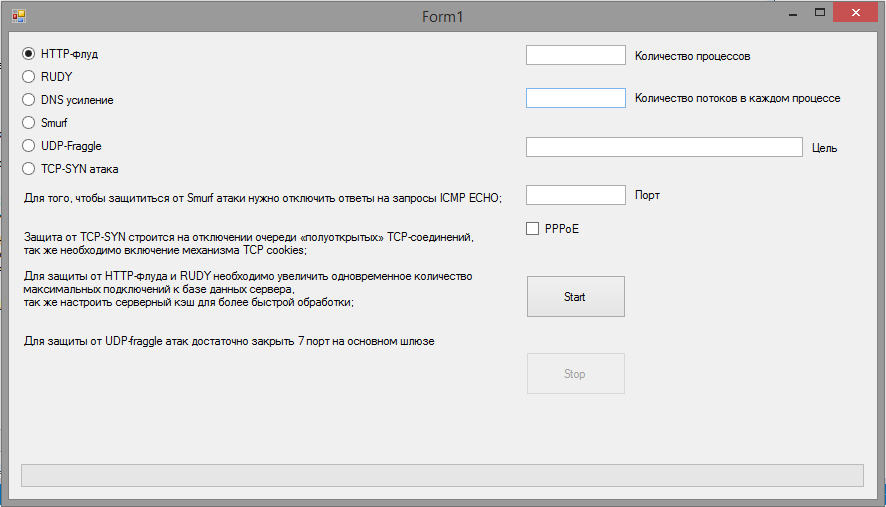


Рисунок 5.1 – Скриншот модуля интерфейса

Таблица 5.1 – Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| Модуль | Содержание теста | Ожидаемый результат | Тест пройден |
| 1 | 2 | 3 | 4 |
| Интерфейса | Отсутствие значений в полях для ввода | Атака не запустится | да |
| Интерфейса | Текстовые данные в полях для ввода чисел | Атака не запустится | да |
| Интерфейса | Некорректные числовые данные в полях для ввода чисел | Атака не запустится | да |
| Интерфейса | Ввод несуществующей цели | Модуль атаки запустится, но атака не начнется | да |

*Продолжение таблицы 5.1*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Интерфейса | Запуск атаки при активной атаке | Кнопка неактивна | да |
| Интерфейса | Остановка атаки до старта атаки | Кнопка неактивна | да |
| Интерфейса | Запуск атаки с корректными значениями | Заполняется полоса загрузки, запускается модуль атаки | да |
| Интерфейса | Остановка запущенной атаки | Полоса загрузки сбрасывается, модуль атаки закрывается | да |
| Атакующего процесса | Запуск с недостаточным количеством данных | Атака не запускается | да |
| Атакующего процесса | Запуск с избыточным количеством данных | Избыточные данные игнорируются | да |
| Атакующего процесса | Запуск с некорректными данными | Модуль запустится, атака не начнется | да |
| Атакующего процесса | Запуск атаки типа Smurf в локальной сети | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса в локальной сети | да |
| Атакующего процесса | Запуск атаки типа UDP Fraggle в локальной сети | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса в локальной сети | да |
| Атакующего процесса | Запуск атаки типа TCP-SYN в локальной сети | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса в локальной сети | да |
| Атакующего процесса | Запуск атаки типа HTTP-Flood | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса | да |

*Продолжение таблицы 5.1*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Атакующего процесса | Запуск атаки типа RUDY | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса | да |
| Атакующего процесса | Запуск атаки типа Smurf | Увеличение времени отклика цели или полная потеря доступа с любого ресурса | нет |
| Атакующего процесса | Запуск атаки типа UDP Fraggle | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса | нет |
| Атакующего процесса | Запуск атаки типа TCP-SYN | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса | нет |
| Атакующего процесса | Запуск атаки типа DNS усиление | Увеличение времени отклика цели или полная потеря доступа к нему с любого ресурса | нет |
| Мониторинга | Запуск модуля | Открытие страницы для проверки доступности | да |
| Определения идентификатора текущей PPPoE сессии | Запуск модуля без PPPoE подключения | Возвращен нуль | да |
| Определения идентификатора текущей PPPoE сессии | Запуск модуля с PPPoE подключением | Возвращен идентификатор текущей PPPoE сессии | да |

Как видно из таблицы, приложение хорошо справилось с тестами, что говорит о высокой работоспособности, однако некоторый функционал все же не работает. По причине неработающего функционала были проведены исследования с привлечением добровольцев, а так же с использованием программы WireShark, которая позволяет просмотреть содержимое всех отправляемых и принимаемых пакетов.

По результатам исследований были выявлены следующие причины недостаточной функциональности:

* В пакете с PPPoE заголовком нельзя подменить исходящий IP адрес, а так же установить целевым широковещательный адрес, таким образом атаки типа Smurf и UDP Fraggle произвести невозможно;
* По этой же причине нельзя произвести TCP-SYN атаку, так как атакующий компьютер отвечает на ACK пакет если не подменить исходящий адрес на несуществующий;
* В случае отправки DNS-пакета для атаки типа DNS усиление, он автоматически дополняется данными о компьютере-отправителе до размера в 1600 байт, таким образом заведомо нагружая компьютер посылающий пакет сильнее, чем нагружается целевой ресурс, а вкупе с невозможностью сменить IP адрес источника эта атака является абсолютно бесполезной в нынешних реалиях с белорусскими провайдерами. Данный функционал не тестировался в других странах, поэтому там данные могут отличаться.

**6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Программное средство тестирования надежности сетевых ресурсов разрабатывается для начинающих системных администраторов с целью их самоподготовки в сфере защиты от DDoS-атак. Начинающие системные администраторы с помощью этого программного средства тестируют уже выстроенную защиту и собирают статистику для ее создания.

**6.1** Требования к аппаратному обеспечению

Минимальные требования для запуска модуля интерфейса:

* Процессор Inter Pentium 4 или мощнее;
* 512 мегабайт оперативной памяти;
* Операционная система семейства Windows новее Windows XP;
* .NET Framework 2.0 или выше.

Минимальные требования для запуска модуля атакующего процесса:

* Процессор Inter Pentium 4 или мощнее;
* 128 мегабайт оперативной памяти.
* Операционная система семейства Windows новее Windows 2000;

Минимальные требования для запуска модуля мониторинга:

* Процессор Inter Pentium 4 или мощнее;
* 512 мегабайт оперативной памяти;
* Операционная система семейства Windows новее Windows XP;
* .NET Framework 2.0 или выше.

Минимальные требования для запуска модуля определения идентификатора текущей PPPoE сессии:

* Процессор Inter Pentium 4 или мощнее;
* 128 мегабайт оперативной памяти;
* Операционная система семейства Windows новее Windows XP;
* Библиотека WinPCap.

**6.2** Руководство по установке системы

Скопировать все файлы в один каталог. При необходимости установить платформу .NET Framework версии 2.0 и выше и библиотеку WinPCap.

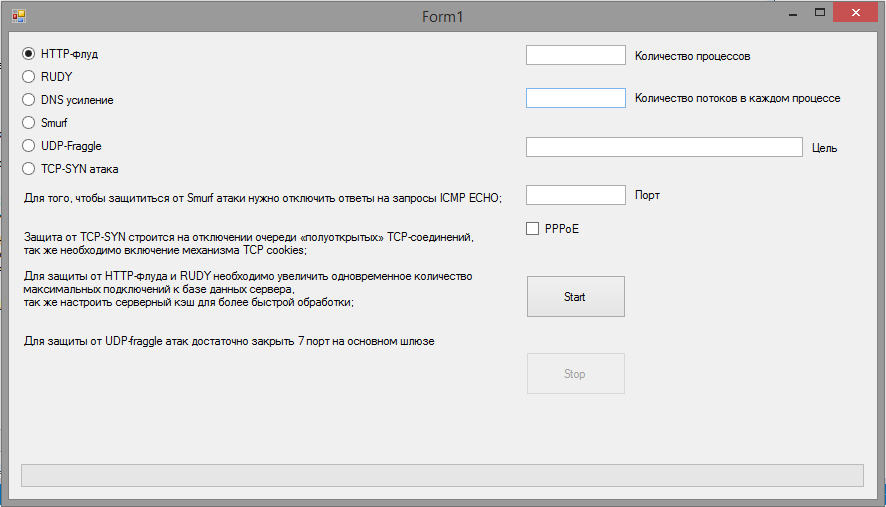
## **6.3** Руководство по использованию программного средства

Так как у данного проекта каждый модуль является отдельным приложением, то все модули кроме модуля определения идентификатора текущей PPPoE сессии будут рассмотрены в отдельности. Модуль определения идентификатора текущей PPPoE сессииотдельно не рассматривается так как не имеет параметров командной строки, а результирующее значение возвращает в качестве кода выхода.

В случае работы с интерфейсом необходимо провести следующие операции:

* Запустить файл GUI\_t1.exe;

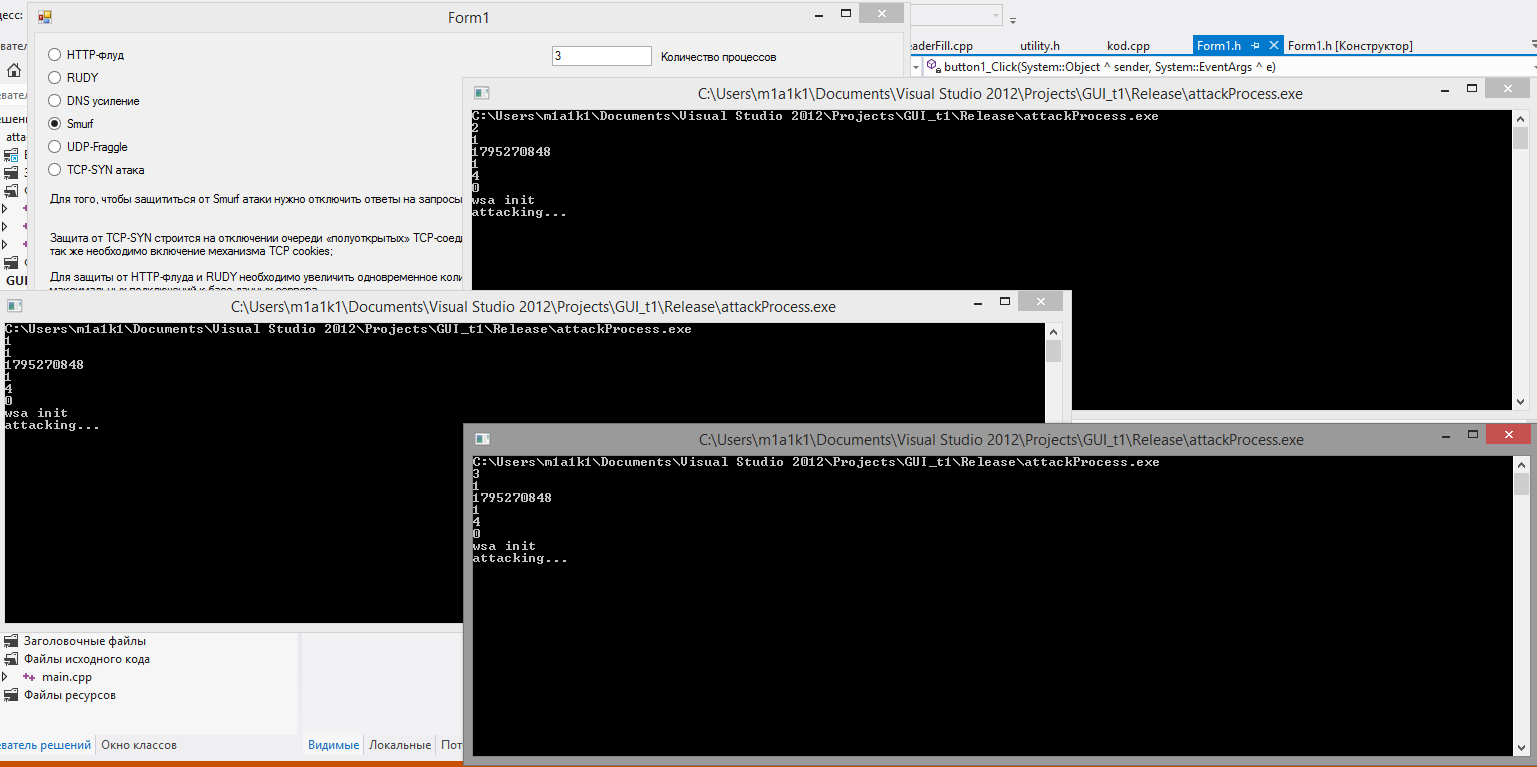
В результате откроется следующее окно (см. рисунок 6.1):

Рисунок 6.1 – Модуль интерфейса

* В этом окне выбрать тип атаки;
* Ввести количество процессов которое будет запущено для атаки (рекомендуется количество процессов меньше либо равное количеству физических ядер процессора или нескольких процессоров в системе);
* Ввести количество потоков в каждом процессе (не рекомендуется более 50, так как вызовет слишком серьезную нагрузку на систему, что может привести к серьезному снижению производительности);
* Ввести адрес цели (можно в формате доменного имени или IP адреса);
* Ввести порт для атаки (в случае RUDY, HTTP-FLOOD, Smurf, DNS усиления и UDP Fraggle он выбирается по умолчанию необходимым для целевых служб);
* В случае использования PPPoE соединения сделать пункт PPPoE активным;
* Нажать кнопку «Start» и дождаться пока заполнится полоса загрузки.

В левой нижней четверти данного окна так же расположены краткие советы по защите от каждого из перечисленных видов атак. Все возможные способы стоит искать на тематических ресурсах, так же часть рекомендаций находятся в обзоре литературы.

В случае некорректного ввода атака не запустится, сообщения об ошибке так же не будет. В случае корректного ввода и запуска атаки окно будет выглядеть следующим образом (см. рисунок 6.2):

Рисунок 6.2 – Запущенное с помощью модуля интерфейса приложение

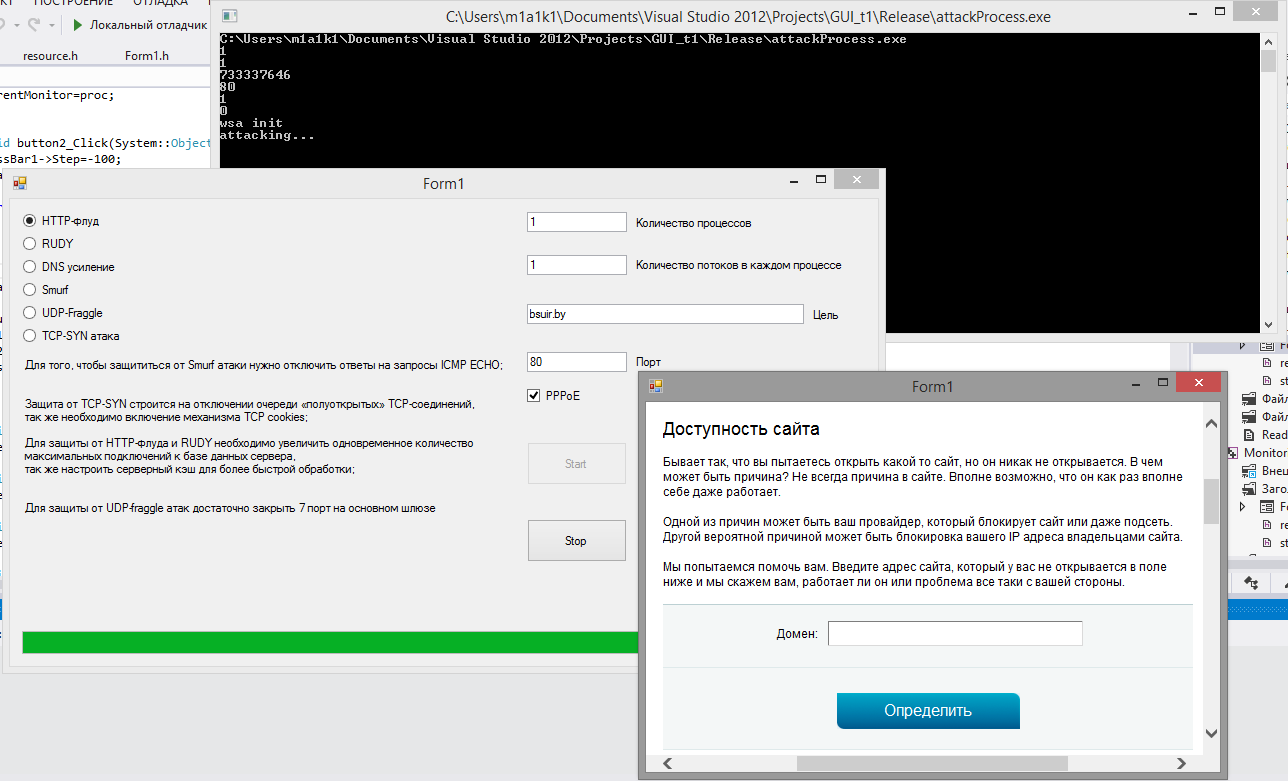


Рисунок 6.3 – Запущенное приложение с активным PPPoE соединением

В случае использование PPPoE соединения и, следовательно, модуля мониторинга приложение выглядит так как это показано на рисунке 6.3.

В случае использования модуля атакующего процесса необходимо выполнить следующие действия:

* Запустить Командую строку от имени Администратора;
* Перейти в каталог с файлом attackProcess.exe;
* Запустить attackProcess.exe со следующими параметрами: идентификатор процесса, количество потоков, IP адрес цели в формате, используемом в программе, порт назначения, тип атаки цифрой и идентификатор текущей PPPoE сессии или 0 если атака проводится в локальной сети.

В частном случае, со значениями идентификатора процесса равного 1, с 1 потоком, IP адресом цели равным 1.1.1.1, 80 портом назначения, атакой типа UDP Fraggle и без использования PPPoE, запуск будет выглядеть как показано на рисунке 6.4, а запущенное приложение идентично рисунку 6.5.

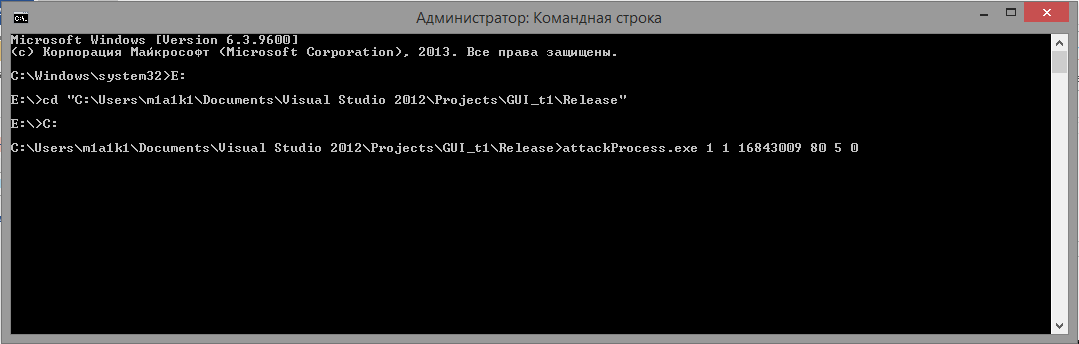
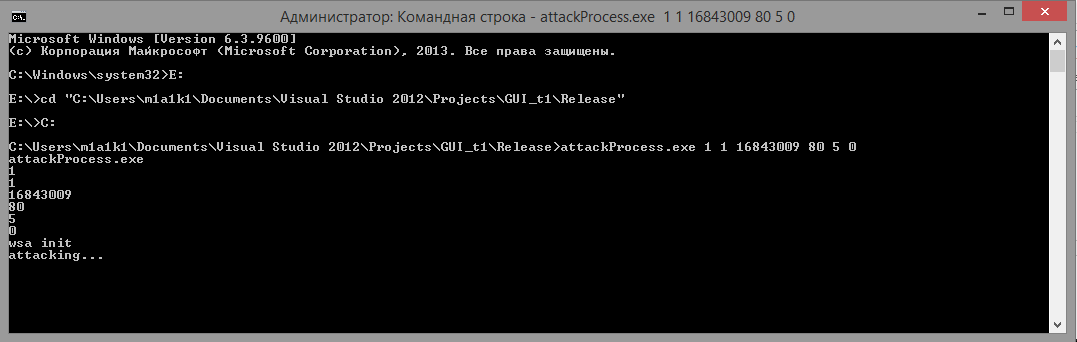


Рисунок 6.4 – Запуск атаки без использования модуля интерфейса

Рисунок 6.5 – Запущенная атаки без использования модуля интерфейса

# 7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВА ТЕСТИРОВАНИЯ НАДЕЖНОСТИ СЕТЕВЫХ РЕСУРСОВ

## **7.1** Характеристика программного продукта

Разрабатываемое в данном дипломном проекте приложение предназначено для тестирования надежности сетевых ресурсов на устройствах под управлением ОС Windows. Приложение позволяет протестировать устойчивость ресурса к нескольким основным типам DDoS-атак.

Данный программный продукт разрабатывается для свободной продажи на рынке IT. Пользователями данного продукта могут являться обычные люди, у которых имеется компьютер под управлением операционной системы Windows. Так как это продукт для операционной системы устанавливаемой на стационарных компьютерах, то на основе исследования аналогов можно заметить, что основным отличием разрабатываемой системы от аналогов является современность и модульность.

Исходя из проведенного маркетингового исследования, лицензии на данный программный продукт будут востребованы в течение четырех лет: в 2017 году планируется продать 1000 лицензий, в 2018 – 2000 лицензий, в 2019 – 1500 лицензий, в 2020 – 1000 лицензий.

Разработка и внедрение данной системы позволят:

* повысить уровень защищенности сетевых ресурсов;
* упростить настройку защиты сетевых ресурсов.

Экономическая целесообразность инвестиций в разработку и использования приложения тестирования надежности сетевых ресурсов осуществляется на основе расчета и оценки следующих показателей:

* чистая дисконтированная стоимость (ЧДД);
* срок окупаемости инвестиций (ТОК);
* рентабельность инвестиций (Рн).

В результате разработки данного программного продукта пользователи смогут приобрести данное приложение и сразу же использовать по назначению, не устанавливая дополнительных модулей.

**7.2** Расчет сметы затрат и отпускной цены программного средства

Основная заработная плата исполнителей проекта определяется по следующей формуле:

(7.1)

где *n* – количество исполнителей, занятых разработкой программных средств (ПС);

– часовая тарифная ставка i-го исполнителя, руб;

– трудоемкость работ, выполняемых i-ым исполнителем, ч.

В настоящий момент тарифная ставка 1-го разряда в компании составляет 65 руб. Расчетная норма рабочего времени на 2017 год для пятидневной рабочей недели составляет 168 часов. Штат исполнителей проекта состоит из руководителя проекта и инженера-программиста первой категории.

Расчет основной заработной платы исполнителей проекта представлен в таблице 7.1.

Таблица 7.1 – Расчет основной заработной платы исполнителей

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Исполнитель | Раз-ряд | Тарифный коэффици-ент (Tк) | Часовая тарифная ставка (Тч), руб. | Трудо-емкость работ (*t*i), ч. | Премия | | | Заработ-ная плата (З), руб. | |
| (%) | руб. | | |  |
| Руководитель проекта | 17 | 3,98 | 2,29 | 320 | 50 | 366,40 | | | 1099,60 |
| Инженер-программист | 10 | 2,48 | 1,42 | 480 | 340,80 | | | 1022,40 |
| Основная заработная плата, Зо | | | | | | | 2122 | | |

Дополнительная заработная плата исполнителей проекта определяется по формуле:

(7.2)

где Нд – норматив дополнительной заработной платы, 10%

После подстановки значений в формулу (7.2) дополнительная заработная плата составит:

Отчисления в фонд социальной защиты населения и на обязательное страхование (Зсз) определяются в соответствии с действующими законодательными актами по формуле:

(7.3)

где Нсз – норматив отчислений в фонд социальной защиты населения и на обязательное страхование, 34+0,6%.

Размер отчислений в фонд социальной защиты населения и на обязательное страхование по формуле (7.3) составит:

Расходы по статье «Машинное время» (Рм) определяются по формуле:

(7.4)

где Цм – цена одного машино-часа;

Тч – количество часов работы в день;

Ср – длительность проекта.

Стоимость машино-часа на предприятии составляет 50 коп. Разработка проекта займет 60 дней. Определим по формуле (7.4) затраты по статье «Машинное время»:

Затраты по статье «Накладные расходы» (Рн), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (Рн), определяются по формуле:

(7.5)

где Нрн – норматив накладных расходов, 50%.

После подстановки значений в формулу (7.5) накладные расходы составят:

Общая сумма расходов по всем статьям сметы (Сп) на ПО рассчитывается по формуле:

(7.6)

По формуле (7.6) получаем сумму расходов по всем статьям сметы:

Кроме того, организация-разработчик осуществляет затраты на сопровождение а адаптацию ПС (Рса), которые определяются по формуле:

(7.7)

где Нрса – норматив расходов на сопровождение и адаптацию, 20%.

Затраты на сопровождение и адаптацию программного продукта по формуле (7.7) составят:

Общая сумма расходов на разработку (с затратами на сопровождение и адаптацию) как полная себестоимость ПС (Сп) определяется по формуле:

. (7.8)

Полная себестоимость программного средства, рассчитанная по формуле (7.8) составит:

Затраты на реализацию определяются по формуле:

(7.9)

Затраты на реализацию, рассчитанные по формуле (7.9) составят:

Затраты с учетом затрат на реализацию определяются по формуле:

(7.10)

Цена программного продукта, с учетом затрат на его реализацию составит:

## **7.3** Расчет экономического эффекта продажи программного продута

Экономический эффект организации-разработчика программного продукта в данном случае заключается в получении прибыли от его продажи множеству потребителей. Прибыль от реализации в данном случае напрямую зависит от объемов продаж, цены и затрат на разработку ПО.

На основании маркетингового исследования цена одной копии лицензии составила 4 руб. При этом лицензии на программный продукт будут востребованы на рынке в течение четырех лет: в 2017 году планируется продать 1000 лицензий, в 2018 – 2000 лицензий, в 2019 – 1500 лицензий, в 2020 – 1000 лицензий.

Прибыль от продажи одной лицензии программного продукта определяется по формуле:

, (7.11)

где Ц – цена реализации одной лицензии ПО, руб.;

*N* – количество лицензий, которое будет куплено клиентами за период с 2017 по 2020 год;

Пед – прибыль, получаемая организацией-разработчиком от реализации одной копии программного продукта, руб;

НДС – сумма налога на добавленную стоимость, руб.

Сумма налога на добавленную стоимость рассчитывается по формуле:

(7.12)

где НДС = 20%.

Прибыль от продажи одной копии программного продукта, рассчитанная по формулам (7.11) и (7.12) составит:

Суммарная годовая прибыль рассчитывается по формуле:

(7.13)

Суммарная годовая прибыль за каждый год, рассчитанная по формуле (7.13), составит:

Чистая прибыль рассчитывается по формуле:

(7.14)

где Нприб – ставка налога на прибыль, 18%.

Таким образом, чистая прибыль за каждый год реализации программного продукта составит:

## **7.4** Расчет показателей экономической эффективности разработки и реализации программного продукта

Для расчета показателей экономической эффективности использования программного продукта, необходимо полученные суммы результата (прироста чистой прибыли) и затрат (капитальных вложений) по годам приводят к единому времени – расчетному году (за расчетный год принят 2017 год) путем умножения результатов и затрат за каждый год на коэффициент дисконтирования (*αt*) который рассчитывается по формуле:

(7.15)

где Ен – норматив приведения разновременных затрат результатов на 1.03.2017, 15%;

*t* – номер года, результаты и затраты которого приводятся к расчетному (2017 – 1, 2018 – 2, 2019 – 3, 2020 – 4).

Расчеты коэффициента приведения за каждый год по формуле (7.15) примут вид:

Результаты расчетов показателей эффективности разработки и реализации программного продукта сведены в таблицу 7.2.

Таблица 7.2 – Расчет экономической эффективности от реализации нового программного средства

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование показателей | Ед.  изм. | Усл.  обозн. | Расчетный период | | | |
| 2017 | 2018 | 2019 | 2020 |
| Экономический эффект | руб. | ЧП | 1804 | 3608 | 2706 | 1804 |
| Коэффициент дисконтирования | руб. | *αt* | 1 | 0,87 | 0,75 | 0,66 |
| Дисконтированный результат | руб. |  | 1804 | 3139 | 2029,50 | 1190,60 |
| Затраты на разработку программного средства | руб. | Ср |  | - | - | - |
| Дисконтированные инвестиции | руб. | Иt |  | - | - | - |
| ЧДД по годам | руб. | ЧДД*t* | -3793,90 | 3139 | 2029,50 | 1190,60 |
| ЧДД с нарастающим итогом | руб. | ЧДД | -3793,90 | -654,90 | 1374,60 | 2565,20 |

Рассчитаем рентабельность инвестиций в разработку и реализации программного продукта (Ри) по формуле:

(7.17)

где Пчср – среднегодовая величина чистой прибыли за расчетный период, руб., которая определяется по формуле:

(7.18)

где Пч*t*– чистая прибыль, полученная в году *t*, руб.

По формулам (7.17) и (7.18) рассчитаем рентабельность инвестиций:

Тогда, имеем:

В результате технико-экономического обоснования разработки программного продукта были получены следующие значения показателей эффективности:

* чистый дисконтированный доход за четыре года продаж продукта составит 2565,20 руб.;
* затраты на разработку программного продукта окупятся на третий год продаж;
* рентабельность инвестиций составит 44%.

Таким образом, разработка и продажа данного программного продукта является эффективным вложением инвестиций и поэтому такую разработку целесообразно осуществлять.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Хакер : справочник / К. Касперски [и др.] – М. : Издательство Gameland, 2003. – 657 с.

[2] Шнайер, Б. Секреты и ложь. Безопасность данных в цифровом мире. / Б. Шнайер. – СПб. : Питер, 2000. – 432 c.

[3] TCP SYN Flooding Attacks and Common Mitigations : Databook / Verizon. – Wesley M. Eddy, London, 2007.

[4] Deep Inside a DNS Amplification DDoS Attack : Databook / Intel Corporation. – Matthew Prince, New York, 2012.

[5] Applied Cryptography : Databook / John Wiley & Sons. – Bruce Schneier, Ca, 1996.

[6] Intrusion Signatures and Analysis : Databook / New Riders Publishing. –Stephen Northcutt, San Francisco, 2001.

[7] Ixbt [Электронный ресурс]. – Режим доступа: https://www.ixbt.com/.

[8] Distributed DoS Attacks: classification and Defense Mechanisms : Databook / INDIACom – Dileep Kumar, Pune, 2015.

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Вводный плакат

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Структурная схема