Государственное бюджетное общеобразовательное учреждение города Москвы "Школа № 1525"

«Цифровая безопасность»

Проектная работа

Выполнили:

Куссиг Луиджи

Зобнин Александр

ученики 9 ФМ класса

Москва, 2020

Содержание

[Теория 3](#_Toc40708899)

[Практика 4](#_Toc40708900)

[Испытания и итоги 7](#_Toc40708901)

[Список использованных источников и интернет-ресурсов 10](#_Toc40708902)

[Приложение 1 (Исходный код «вируса») 11](#_Toc40708903)

[Приложение 2 (Исходный код консоли) 12](#_Toc40708904)

[Приложение 3 (Исходный код «змейки») 13](#_Toc40708905)

Введение

В наши дни цифровой безопасности уделяется незаслуженно мало внимания. В России, количество киберпреступлений в первом квартале 2020 года выросло на 83,9% [1]. Также, в официальном ежегодном отчете о киберпреступности (ACR) за 2019 год, опубликованном Cybersecurity Ventures, сообщается, что атаки хакеров во всём мире происходят каждые 14 секунд, а к 2021 году их частота возрастёт до каждой 11 секунды. И данный список статистики можно продолжать бесконечно. Мы решили испытать наши теоретические знания на практике, написав программу, которая будет проверять работоспособность защитных ПО. Стоит уточнить, что целевой платформой является Windows, поэтому, мы использовали язык C++ (версия компилятора C++17) и множество библиотек, установленных по умолчанию на машинах Windows чтобы избежать как можно больше проблем с совместимостью нашего «вируса»

# Теория

Суть программы –получать приказы от нас через консоль. Эта консоль будет связываться с нашим «вирусом» по интернету. Команды, получаемые клиентом, должны исполняться на устройстве жертвы. Мы задумали четыре программы (для начала): Вычитывание на клиенте текста, присылаемого с сервера; Сворачивание открытого пользователем окна; Воспроизведение звуков, заранее на перекопированных на устройство жертвы.

Для нашей программы необходимо три главных компонента:

1. Программа-установщик
2. Клиент (устанавливается на компьютер жертвы)
3. Сервер (устанавливается на наш компьютер)

\*Дальше к этим компонентам мы будем обращаться по их именам

Разберемся поподробнее с каждым из компонентов нашего проекта. Начнем по порядку. Программа-установщик необходима для того, чтобы внедрить клиент и необходимые для работы клиента файлы в папку автозапуска компьютера подопытного (такая директория на компьютере, в которой содержатся программы, которые запускается со включением компьютера). Это необходимо, чтобы наше ПО работало всегда, даже после перезапуска компьютера. Также наш установщик должен выполнять роль «наживки», чтобы скрыть клиент. Клиент, в свою очередь, должен быть компактным и надежным приложением. Надежным в плане того, что оно не должно выключаться без команды сервера, ведь это обозначает потеря контроля над клиентом через интернет. Даже если сервер вдруг перезапустился клиент должен не выключаться и ждать повторного запуска сервера. Клиент также должен быть способен вычитывать текст, скрывать окна и воспроизводить звуки. Наконец, сервер. Он должен иметь что-то на подобие интерфейса, чтобы взаимодействовать с пользователем и отправлять его команды через интернет.

Все коммуникации будут происходить по протоколу TCP. Наш выбор основывался на надежности успешной передачи данных предоставляемый данным протоколом, в отличие от его аналога – быстрый, но не надежный протокол UDP.

# Практика

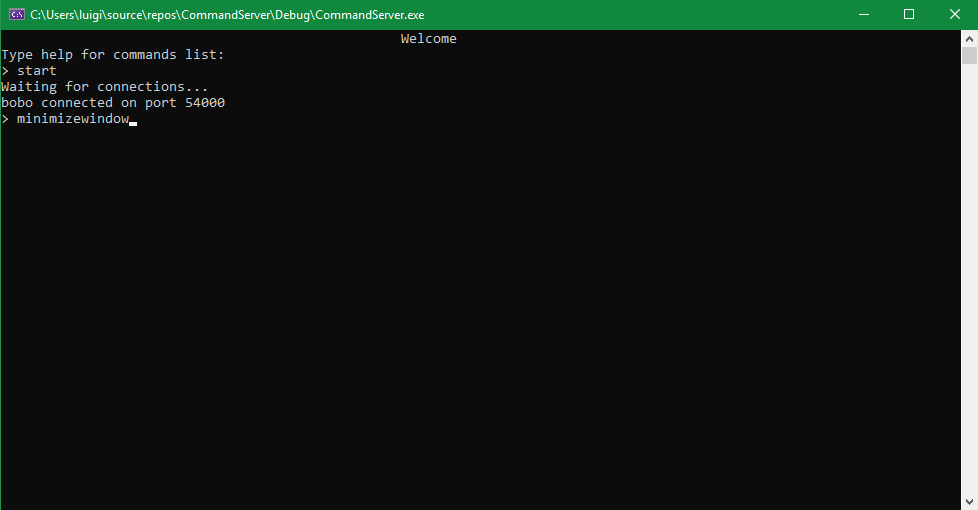
На деле, осуществить нашу задумку оказалось сложнее чем мы думали. Мы разделили задачу на пять подзадач:

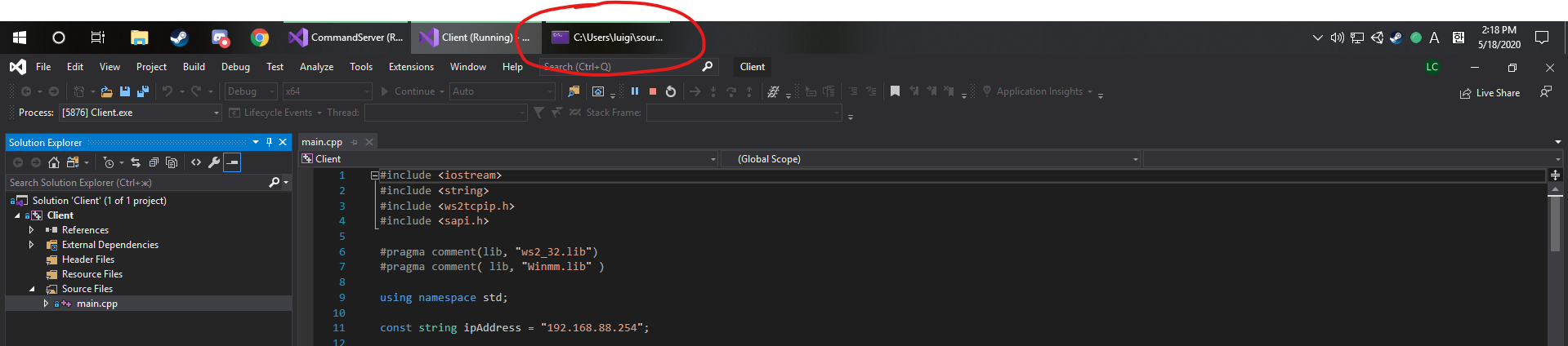
1. Научиться предавать данные по интернету
2. Имплементировать выполнение команд на клиенте
3. Создать интерфейс выбора команды на сервере
4. Разработать систему сервер-клиент чтобы программы могли работать совместно
5. Создать установщик, который способен удовлетворить все необходимые условия для работы клиента на чужом устройстве.

В течение разработки возникло множество технических проблем, которые нам приходилось решать. Нашим главным источником знаний являлся интернет.

Первым шагом мы научились азам коммуникации по интернету. Это было просто, но теперь нам надо было подчинить эту технологию так чтобы она выполняла то, чего мы ходим. Нам надо было придумать каким образом клиент и сервер будут «понимать» друг друга. Наше решение было таким: передаем одно из набора определенных слов каждое из которых соответствует выполнению определенной команды. Так, например, слово «minw» обозначает команду «скрыть окно» (от англ. MINimize Window)

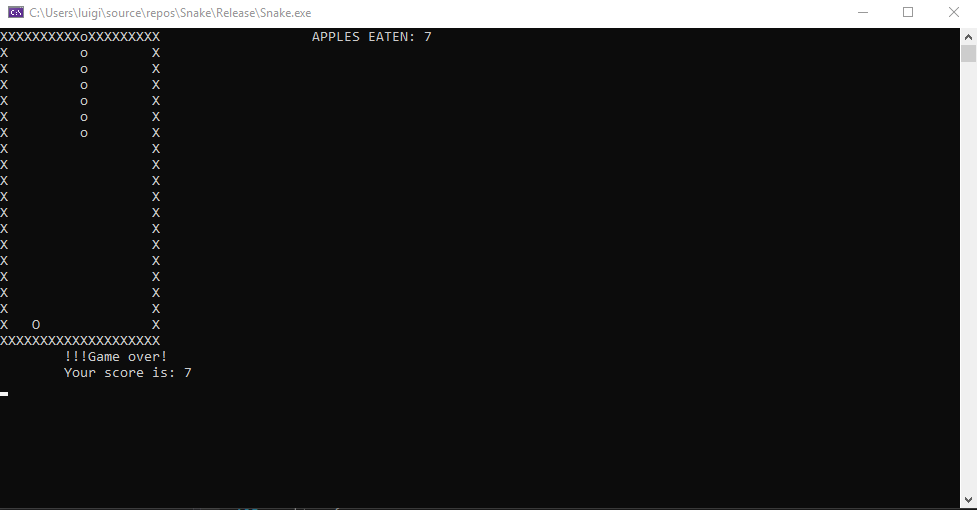
Эта строчка посылается через интернет к клиенту, там она интерпретируется и выполняется соответствующий набор команд, приводящий к тому, что окно, открытое пользователем, сворачивается.

Вот команда «minimizewindow» набрана в консоли:

После нажатия «Enter» окно консоли сервера скрывается (так как для тестов мы запустили клиент и сервер на одном компьютере, и консоль являлась активным окном)

[Ссылка на видео-демонстрацию](https://youtu.be/qnTmumQt_QQ) (Включите субтитры для пояснений)

Мы специально придумывали такие команды, чтобы одновременно демонстрировать работоспособность наших программ, но также не сильно вредить жертве.

Другие команды принимаются и обрабатываются по схожим принципам. Далее, после того как мы полностью завершили разработку всех команд, мы приступили к системе установки. Изначально она представляла из себя простенькую игру «змейка».

[Ссылка на видео-демонстрацию](https://youtu.be/p1KpR0XLJ0A) (Просто видео, показывающее как работает змейка)

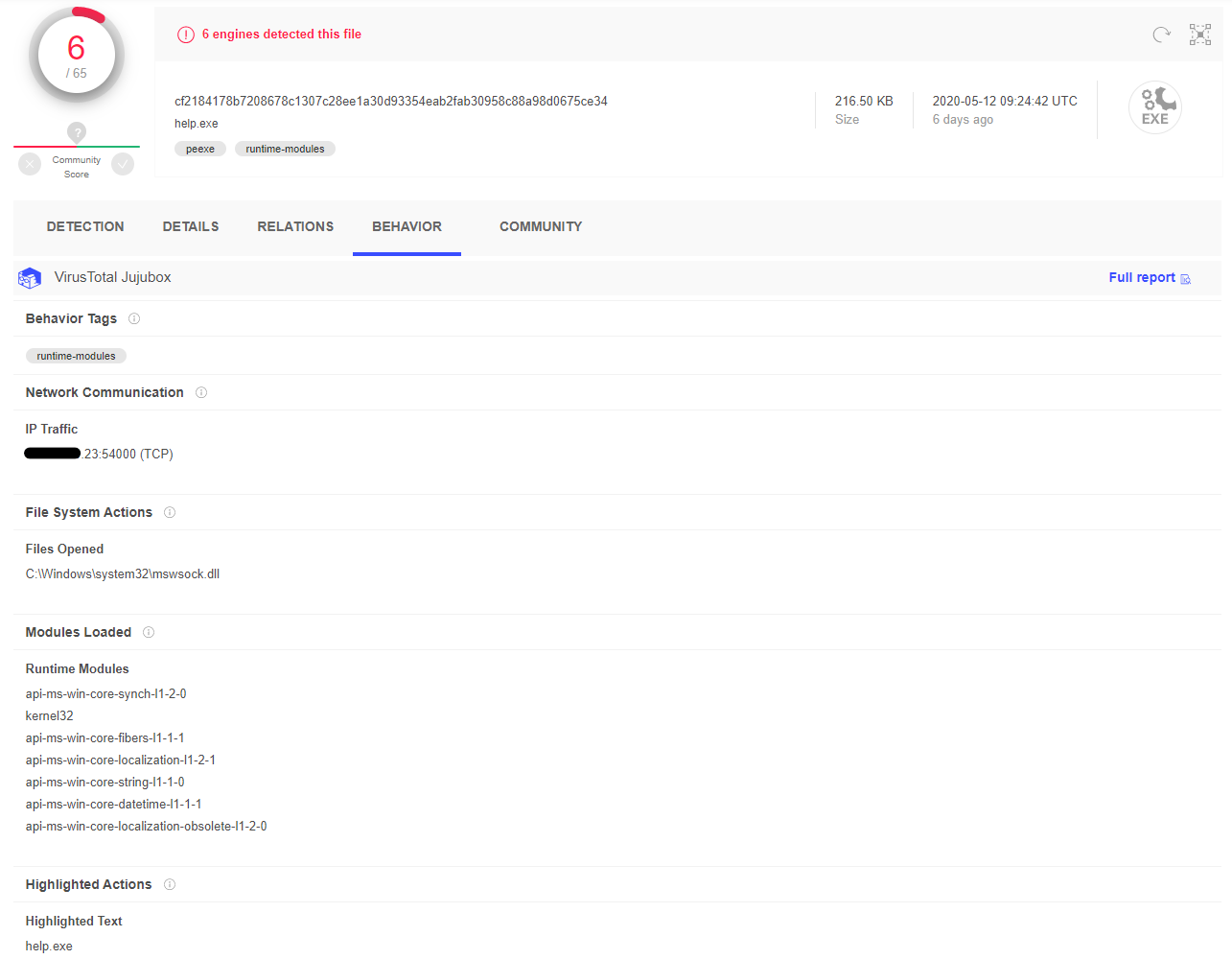
Игра является маскировочной частью установщика, ведь на самом деле она нужна только для того, чтобы отвлечь неопытного пользователя от операций, производимых без его ведома. При запуске «змейки» срабатывает модуль установки клиента и необходимых файлов и модуль первоначального запуска клиента (Ведь если мы не запустим клиент вместе с игрой, то нам придётся ждать до следующей перезагрузки компьютера чтобы клиент включился и совершал попытки подключения к серверу). То есть, даже если пользователь удалит змейку после того, как в нее поиграл, клиент останется на компьютере жертвы. Изначально мы использовали змейку как «наживку», но теперь мы перешли на новую игру. В эту новую игру мы также добавили модули установки, и поэтому она является полноценной заменой предыдущей простенькой змейки.

Наконец мы смогли собрать все три компонента нашего проекта, и мы были готовы к первому тесту. Сначала мы проверили код между собой, исправность всех модулей, запускается ли программа при включении компьютера и самое главное, дает ли она желаемого результата. Естественно, не все пошло гладко с первого теста, но в конце концов, мы смогли починить все ошибки.

Мы были готовы ставить нашей программе настоящее испытание.

# Испытания и итоги

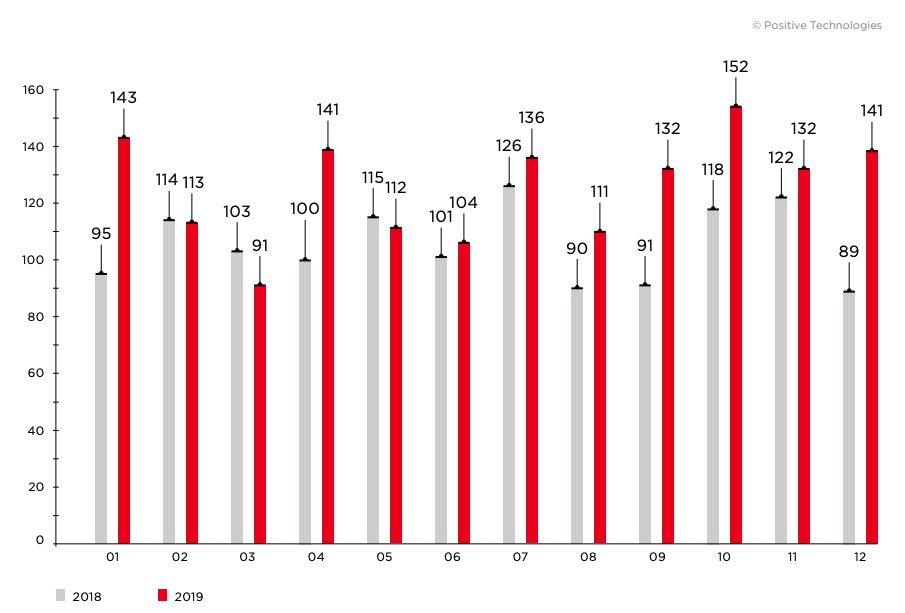
В качестве первого испытания, мы просканировали наш софт на сайте virustotal.com – сайт, который проверяет загруженный в него файл или приложение на вирусы. Для этого он проводит анализ многими антивирусами.

Результат был неожиданным. Несмотря на то, что наше приложение классифицируется как вредоносное ПО всего лишь 6 из 65 антивирусов смогли выявить угрозу.

Таким образом, мы смогли создать программу, которую не смог обнаружить встроенный в Windows антивирус. Более того, протестировав нашу программу на других антивирусах, мы удостоверились в том, что большинство из них не способны обнаружить клиент, который и позволяет манипулировать действиями на компьютере жертвы. Также проведя эксперимент по внедрению данной программы нашему другу, без его ведома, мы убедились, что обычные пользователи зачастую действуют неосторожно, пользуясь интернетом и скачивая файлы из непроверенных источников. Итак, “вирус” оставался у нашего друга, пока мы не предложили ему установить сторонний антивирус Avast, который и заблокировал клиент в папке автозапуска. Можно сделать вывод, что, если такие школьники, как мы - не особо разбирающиеся в сфере информационной безопасности, сумели спрятать от обнаружения и впоследствии воспользоваться программой, посредством которой можно управлять чужим компьютером, тогда задумайтесь: что может сделать настоящий киберпреступник, обладающий ещё большими знаниями, чем те, которые получили мы.

По данным газеты “Ведомости”: 22% промышленных предприятий в России столкнулись с целевыми атаками хакеров в 2017 г. В мире с этой проблемой столкнулось 28% предприятий.

А международная компания Pozitive Technologies сообщает, что в 2019 году было зафиксировано более полутора тысяч атак; это на 19% больше, чем в прошлом году.

И цифры, к сожалению, продолжают расти, а потому стоит побеспокоиться и выполнять следующие пункты для сохранения конфиденциальности и защиты от атак злоумышленников:

* Используйте только лицензионное ПО;
* Используйте эффективные средства антивирусной защиты на всех устройствах;
* Проверяйте все вложения, полученные по электронной почте, с помощью антивирусного ПО;
* Не загружайте файлы с подозрительных веб-ресурсов или из других неизвестных источников.
* Будьте предельно внимательны при вводе учетных данных на сайтах и во время работы с онлайн-платежами;
* С осторожностью относитесь к сайтам с некорректными сертификатами и учитывайте, что введенные на них данные могут быть перехвачены злоумышленниками;

# Список использованных источников и интернет-ресурсов

1. Отчёт о киберпреступности за 2019г. от Cybersecurity Ventures

<https://vc.ru/services/103616-utechki-dannyh-2019-statistika-tendencii-kiberbezopasnosti-i-mery-po-snizheniyu-riskov-vzloma>

1. Рост киберпреступности в 2017г. по данным газеты “Ведомости”

<https://www.vedomosti.ru/technology/articles/2018/09/10/780404-fsb-tsentr-po-borbe-s-kiberatakami>

1. Итоги 2019г. от “Pozitive Technologies”

[https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2019/#id21](https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2019/%23id21)

1. std::string API  
    <http://www.cplusplus.com/reference/string/string/>
2. time.h header API  
   <https://pubs.opengroup.org/onlinepubs/007908799/xsh/time.h.html>
3. stdlib.h header API  
   <https://www.tutorialspoint.com/c_standard_library/stdlib_h.htm>
4. Creating a Basic Winsock Application  
   <https://docs.microsoft.com/en-us/windows/win32/winsock/creating-a-basic-winsock-application>
5. Winbase.h header API  
   <https://docs.microsoft.com/en-us/windows/win32/api/winbase/>
6. Microsoft Speech API (SAPI) 5.3   
    <https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723627(v%3Dvs.85)>
7. Microsoft winsock API  
    <https://docs.microsoft.com/en-us/windows/win32/winsock/windows-sockets-start-page-2>
8. Sloan Kelly Creating a TCP Server in C++  
   <https://www.youtube.com/watch?v=WDn-htpBlnU>
9. Sloan Kelly Creating a TCP Client in C++  
    <https://www.youtube.com/watch?v=0Zr_0Jy8mWE>

Ссылка на исходный ход трех компонентов нашего проекта:

<https://github.com/spicymuffin/bob>

# Приложение 1 (Исходный код «вируса»)



\*Двойное нажатие чтобы показать полностью

# Приложение 2 (Исходный код консоли)

 \*Двойное нажатие чтобы показать полностью

# Приложение 3 (Исходный код «змейки»)

\*Двойное нажатие чтобы показать полностью