Статья Обзор модульного мульти RAT: Taidoor

Taidoor — крайне эффективная вредоносная программа класса RAT (remote access trojan), предназначенная для использования без закрепления в системе. Модульная система, реализованная в Taidoor, отличается от многих других RAT гибкостью: операторы программы могут отправлять на зараженную систему только те модули, которые нужны для достижения целей конкретной атаки.

Чтобы затруднить обнаружение, Taidoor использует несколько разных методов: манипуляции с временными метками, удаление файлов с модулями, обфускацию строк, поиск антивируса на атакуемой машине и др.

Мы изучили функциональные возможности и алгоритмы работы Taidoor, а также ее загрузчиков, и хотим поделиться своими наблюдениями.

Источник информации о зараженной системе, включая имена файлов, — отчет агентства кибербезопасности и безопасности инфраструктуры США (CISA) номер AR20-216A.

Taidoor в арсенале злоумышленника

С точки зрения злоумышленника, Taidoor — достойный базовый инструмент удаленного управления с многочисленными возможностями для динамического расширения. К тому же у малвари довольно удобный интерфейс для запуска процессов и взаимодействия с консолью.

Два встроенных модуля позволяют собирать информацию о зараженной системе. По мере необходимости с сервера управления можно отправлять дополнительные модули.

В ходе общения с управляющим сервером Taidoor использует криптоконверт RSA + AES для обеспечения конфиденциальности. Однако криптография в программе реализована слабо: при перехвате трафика можно подменить команды от сервера. Также при использовании указанного алгоритма на сервере может появиться уязвимость, позволяющая расшифровать отправляемые на сервер данные, к которым удалось получить доступ.

Загрузчики Taidoor

Загрузчики основного тела Taidoor — файлы rasautoex.dll и ml.dll, идентичные по функциональности. Они отличаются только разрядностью: первый является 64-битной версией малвари, второй — 32-битной. Их свойства представлены в табл. 1.

Табл. 1. Свойства загрузчиков Taidoor

Свойства	64-битная версия	32-битная версия
Имя файла	rasautoex.dll	ml.dll
Тип файла	PE32+ Executable (DLL)	PE32 Executable (DLL)
Класс ВПО	Загрузчик (Loader)	Загрузчик (Loader)
MD5	4ec8e16d426a4aaa57c454c58f447c1e	6aa08fed32263c052006d977a124ed7b
SHA-1	5c89629e5873072a9ca3956b67cf7b5080312c80	9a6795333e3352b56a8fd506e463ef634b7636d2
SHA-256	6e6d3a831c03b09d9e4a54859329fbfd428083f8f5bc5f27abbfdd9c47ec0e57	4a0688baf9661d3737ee82f8992a0a665732c91704f28688f6431156
Размер (в байтах)	50 176	43 520

Ниже мы поделимся результатами исследования 64-битной версии.

Функциональные возможности загрузчика

Файл rasautoex.dll предназначен для загрузки основного тела Taidoor в память системы. Он может быть запущен через вызов экспортируемой функции MyStart или зарегистрирован как служба — об этом свидетельствует экспортируемая функция ServiceMain.

Поведение в системе

При запуске загрузчик ищет в той же директории, в которой он находится, файл svchost.dll — зашифрованное основное тело Taidoor. Обнаружив необходимый файл, загрузчик выполняет следующие действия:

1. Загружает содержимое файла в память.
2. Расшифровывает файл алгоритмом RC4 на ключе ar1z7d6556sAyAXtUQc2. В расшифрованном виде svchost.dll представляет собой 64-битную библиотеку. *В случае с 32-битным загрузчиком (ml.dll) библиотека с телом Taidoor, соответственно, 32-битная.*
3. Выполняет маппинг библиотеки в памяти.
4. Находит и заполняет адреса всех импортируемых функций.
5. Находит адрес экспортируемой из библиотеки функции Start. Если такая функция найдена, вызывает ее.

Индикатор компрометации (IoC)
Строка: ar1z7d6556sAyAXtUQc2

Основное тело Taidoor

Основное зашифрованное тело Taidoor представлено в виде одного из файлов svchost.dll, которые отличаются разрядностью. Свойства файлов основного тела Taidoor:

Свойства	64-битная версия	32-битная версия
Имя файла	svchost.dll	svchost.dll
Тип файла	Зашифрованный PE32+ Executable (DLL)	Зашифрованный PE32 Executable (DLL)
Класс ВПО	Троян удаленного доступа (Remote Access Trojan)	Троян удаленного доступа (Remote Access Trojan)
MD5	6627918d989bd7d15ef0724362b67edd	8cf683b7d181591b91e145985f32664c
SHA-1	21e29034538bb4e3bc922149ef4312b90b6b4ea3	f0a20aaf4d2598be043469b69075c00236b7a89a
SHA-256	0d0ccfe7cd476e2e2498b854cef2e6f959df817e52924b3a8bcdae7a8faaa686	363ea096a3f6d06d56dc97ff1618607d462f366139df70c88310bbf77
Размер (в байтах)	183 808	158 208
C2	www[.]infonew[.]dubya[.]net:443	• 210.68.69.82:443 • www[.]cnaweb[.]mrslove[.]com:443

В файле с MD5-хешем, оканчивающимся на 7edd, зашифрована 64-битная версия программы. В файле с MD5-хешем, оканчивающимся на 2664c, — 32-битная. Файлы почти идентичны, но в них прописаны разные адреса серверов управления.

Далее речь пойдет об исследовании 64-битной версии.

Общая характеристика исследуемого образца Taidoor

Изученный образец — зашифрованная вредоносная библиотека, которую запускает отдельная программа, описанная выше.

При запуске с помощью rundll32 Taidoor проверяет наличие сохраненных вне тела программы зашифрованных настроек в параметре RValue ключа реестра SOFTWARE\Microsoft\Windows NT\CurrentVersion. При их отсутствии или другом типе запуска программа использует стандартные настройки, сохраненные в ее теле.

Далее Taidoor расшифровывает настройки, которые включают в себя следующие данные:
- адреса сервера управления (домены или IP-адреса и порты);
- адрес прокси-сервера (необязательно);
- настройки ожидания (время переподключения к серверу в различных ситуациях);
- публичный RSA-ключ сервера.

Если указан прокси, программа пытается подключиться к серверу управления через него. При подключении Taidoor отправляет один зашифрованный по алгоритму RSA пакет с идентификатором и ждет от сервера ответ. Если попытка успешна, программа создает файл `%ALLUSERSPROFILE%\\Application Data\\Microsoft\\~svc_.TMp` — вероятнее всего, это индикатор успешного заражения, который предотвращает повторную загрузку Taidoor.

Программа в ходе основного цикла инициализирует два плагина: один для сбора информации о системе, другой — для создания процессов и работы с командной строкой.

После этого программа запускает в отдельном потоке цикл отстука, в котором она обращается к управляющему серверу каждые 10 секунд. В основном же потоке программа начинает цикл получения команд от сервера. При отправке сообщений на сервер используется криптоконверт RSA + AES. На каждое сообщение генерируется отдельный ключ AES.

Вот какие действия может совершать злоумышленник в зараженной системе при помощи Taidoor в базовой комплектации (основной модуль и два встроенных плагина):

- сохранять файлы на диск,
- изменять настройки подключения к серверу управления,
- загружать новые модули,
- подменять временные метки у файлов,
- запускать процессы и получать их результаты,
- взаимодействовать с командной строкой.

У каждого плагина Taidoor есть свой идентификатор, который указывается в начале каждого сообщения с обеих сторон. Это позволяет оператору программы адресовать команды конкретному плагину и различать источники сообщений от программы (основной модуль и конкретные плагины). По умолчанию используются следующие идентификаторы:

- 1 — основной модуль;
- 2 — плагин для старта процессов и работы с командной строкой;
- 3 — плагин для получения дополнительной информации о системе.

Внешние плагины Taidoor загружает в виде динамических библиотек по команде с сервера управления. В каждом плагине есть три экспортируемые функции:

- Install — инициализация плагина и возвращение идентификатора плагина;
- Proxy — передача плагину адресованного ему сообщения от сервера;
- Uninstall — деинициализация плагина.

Taidoor применяет всевозможные методы, чтобы затруднить обнаружение и расследование атаки:

- Вместо оригинального файла может использоваться копия `cmd.exe` .
- Временные метки индикатора заражения `%ALLUSERSPROFILE%\\Application Data\\Microsoft\\~svc_.TMp` заменяются временными метками системного файла `C:\Windows\System32\services.exe` , что осложняет определение даты заражения.
- Файлы загружаемых плагинов удаляются после запуска.
- До старта процессов плагин проверяет, нет ли на атакуемой машине антивируса Kaspersky.

Инициализация Taidoor

Перед использованием модуль должен быть расшифрован — это делает специализированный загрузчик, описанный выше. Исполнение самого образца начинается с вызова экспортируемой функции Start.

В начале процесса используется механизм для препятствования ручной отладке, в ходе которого программа:

1. считает, сколько секунд по времени от 0 до 60 должно быть через 10 секунд;
2. запускает цикл, ждет по 10 секунд на каждой итерации до тех пор, пока не получит ожидаемое значение.

При ручной отладке с момента вычисления ожидаемого значения до вызова функции wait может пройти больше секунды, в результате чего цикл может сбиться и исполняться вечно.

После запуска Taidoor препятствует ручной отладке с помощью примитивной функции.

Далее программа сама импортирует все необходимые функции за исключением LoadLibrary и GetProcAddress.

Taidoor вызывает еще несколько неопасных и бессмысленных функций, скорее всего, чтобы защититься от обнаружения.

```
__int64 dummy()
{
  const char *v0; // rcx
  char v2[16]; // [rsp+20h] [rbp-19h] BYREF
  char Destination[16]; // [rsp+30h] [rbp-9h] BYREF
  char Str[16]; // [rsp+40h] [rbp+7h] BYREF
  char v5[56]; // [rsp+50h] [rbp+17h] BYREF

  printf("Enter the terminals\n");
  strcpy(Destination, "term");
  printf("Enter the nonterminals\n");
  strcpy(Destination, "nnterm");
  strlen(Str);
  strlen(Destination);
  printf("Enter the input string\n");
  strcpy(v2, "selina");
  v5[1] = Str[0];
  v5[0] = 36;
  v5[2] = 0;
  sub_180001EA8(v5, (__int64)v2, "***", 0);
  v0 = "Accepted\n";
  if ( v2[0] != 36 )
    v0 = "Not Accepted\n";
  printf(v0);
  return 0i64;
}
```

"

Пример бесполезной функции, используемой Taidoor

После этого малварь запускает основную функциональность в новом потоке. Большинство строк вредоносной программы (например, названия импортируемых функций) обфусцированы с использованием нестандартного поточного шифра для усложнения обнаружения. Строки деобфусцируются прямо перед использованием.

Соединение с сервером управления

В начале основного цикла Taidoor проверяет имя приложения, при помощи которого малварь была запущена, чтобы определить источник параметров соединения с управляющим сервером:

- Если использовалась стандартная утилита rundll32.exe, то программа пытается загрузить параметры соединения с сервером из параметра RValue ключа реестра `SOFTWARE\Microsoft\Windows NT\CurrentVersion` .
- Если библиотека была запущена иным способом или параметр RValue не существует, настройки берутся из тела Taidoor.

Настройки, полученные тем или иным способом, расшифровываются алгоритмом AES в режиме ECB с ключом `2B7E151628AED2A6ABF7158809CF4F3C` (представлен в шестнадцатеричном виде).

Далее Taidoor пытается подключиться к серверу управления. В полученных ранее параметрах подключения может быть указано до 4 адресов сервера и до 3 портов на каждый сервер. Также может быть указан адрес и порт HTTP-прокси-сервера. В изученном образце был указан один управляющий сервер и один порт, а в 32-битной версии указаны два управляющих сервера и по одному порту на каждый.

Подключение проходит по следующему алгоритму:

1. Программа берет адрес и порт из конфигурации. Если в конфигурации указаны параметры прокси-сервера, то программа пытается подключиться к нему.
2. Программа подключается к серверу управления или дает прокси-серверу соответствующую команду (при наличии параметров прокси и успешном подключении к нему).
3. Программа отправляет на управляющий сервер массив из 263 символов, где:
    - первые 3 символа — фиксированная строка F::;
    - следующие 4 — количество миллисекунд, прошедших между стартом системы и запуском программы;
    - оставшиеся 256 — строка `0x040x230x190x340xfe0xc1` , зашифрованная при помощи алгоритма RSA_PKCS1v1.5 с использованием криптографически стойкого генератора псевдослучайных чисел (ГПСЧ).
4. В ответ программа ожидает строку 200 OK\r\n\r\n:
    - Если программа получает такой ответ, фаза подключения к серверу завершается.
    - Если программа не получает ожидаемого ответа, то она пробует соединиться со следующим сервером, указанным в параметрах. Если ни к одному из них не удается подключиться, программа засыпает на указанный в параметрах период времени (в данном образце — 30 минут), затем повторяет описанный выше цикл.

Работа с временными метками

Перед соединением с сервером управления Taidoor открывает журнал событий системы и начинает по очереди читать записи. Особое внимание при этом программа обращает на записи типов 6005(«Начало работы службы журнала событий») и 6006 («Конец работы службы журнала событий») — они могут использоваться для определения времени работы машины. В рассмотренном экземпляре реализован проход по этим данным, но программа их не использует.

После подключения к серверу программа создает файл `%ALLUSERSPROFILE%\\Application Data\\Microsoft\\~svc_.TMp` , а также обновляет его временные метки. Затем программа проверяет файл `C:\Windows\win.ini` на наличие секции Micros с ключом source. Если ключ присутствует, утилита cmd.exe копируется по указанному в ключе адресу (в отчете CISA указан адрес c:\temp\cmd.exe).

Функциональные возможности основного модуля

Если в первом байте расшифрованного буфера, пришедшего от сервера, стоит 1, остаток буфера обрабатывается самой Taidoor. Отдельные команды могут устанавливать или использовать глобальные переменные. Вот самые важные из них:

- <имя_файла>;
- <файловый_дескриптор>;
- <идентификатор>;
- <глобальный_массив>;
- <дополнительное_имя_файла>;
- <дополнительный_дескриптор_файла>.

Первый байт остатка буфера, который передается на обработку, идентифицирует одну из команд. Команды могут манипулировать подключением малвари к серверу, обновлять ее настройки, загружать и выгружать плагины, а также выполнять другие вредоносные действия.

С полным списком команд можно ознакомиться тут. Обратите внимание, что команды есть не у всех идентификаторов.

Идентификатор	Команда
2	Отключиться от сервера
3	Создать в текущей директории файл с именем }{. Деинициализировать все плагины. Завершить процесс
4	Загрузить плагин из директории по пути, переданному сервера управления. При успешной инициализации удалить все файлы с именем uaq*.dll в текущей директории. Если инструкции выполнены, отправить серверу сообщение \x01\x05 и тип плагина. Если во время работы произошла ошибка, отправить на сервер сообщение \x01\x06, конкатенированное с описанием ошибки: • Can't find plug file — не получилось найти файл, • Can't load more plug — уже загружено максимальное количество плагинов, • Load Dll Plug Failed — возникли проблемы при загрузке
7	Деинициализировать плагин указанного типа. Если получилось, отправить на сервер сообщение \x01\x08, если нет — сообщение \x01\x06Can't find plug file
9	Отправить на сервер сообщение \x03\x06 и массив из конфигурации. Попытаться открыть файл %temp\~lpz.zp.. Если получилось, отправить файл в нескольких сообщениях — частями по 6000 символов. В сообщении каждая тысяча символов оригинального файла разделена переносом строки, а каждому отсылаемому сообщению предшествует код \x03\x07. После отправки содержимого файл удаляется
10	Сохранить полученный массив (новую конфигурацию) в параметр RValue ключа SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion. Если получилось, отправить конфигурацию на сервер с кодом \x03\x06
11	Попытаться открыть файл с указанным в сообщении именем. Если не получилось, отправить серверу сообщение \x03\x05Can't open update file и закончить обработку команды. Если получилось, проверить его размер. Если файл пуст, отправить серверу сообщение \x03\x05File too small и закончить обработку. Если обработка продолжается, создать <глобальный_массив> и сохранить туда содержимое файла. Если <дополнительный_дескриптор_файла> открыт, закрыть его, файл <дополнительное_имя_файла> переместить в текущую директорию с именем <5 случайных символов латинского алфавита в нижнем регистре>. XMP-файл из сообщения переместить на место <дополнительное_имя_файла>. После этого файл, указанный в сообщении, открывается как <дополнительное_имя_файла>. Вне зависимости от того, был ли открыт до этого <дополнительный_дескриптор_файла>, удалить файл с именем из сообщения (если он не был перемещен), удалить параметр RValue из ключа реестра SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion. В конце отправить на сервер сообщение \x03\x04aaa
12	Проверить, инициализирован ли плагин с указанным типом. Если да, отправить серверу сообщение \x01\x0d, если нет — \x01\x0e
15	Скопировать остаток буфера в переменную <имя_файла>. Попытаться открыть файл на запись и сохранить соответствующий дескриптор в <файловый_дескриптор>. Если не получилось открыть файл или переданный путь к файлу длиннее 260 байт, отправить на сервер сообщение с ошибкой \x01\x06Create File Failed\x00. Если все прошло успешно, отправить \x01\x10
17	Записать остаток буфера в ранее открытый <файловый_поток>

18	Если <файловый_дескриптор> открыт, закрыть его. Заменить значения временных меток файла <имя_файла> на значения временных меток файла C:\Windows\System32\services.exe. Отправить на сервер сообщение \x01\x13 вместе с <имя_файла>
20	Закрыть <файловый_дескриптор>, удалить файл <имя_файла>
32	Отправить \x01\x21 и <идентификатор> на сервер
34	Отключиться от сервера

Тут можно увидеть реализацию команды с идентификатором 15, которая запускает процесс открытия файла.

```
fileStreamInitialized = 0;                  // if command == 15
if ( strlen(buf + 1) >= 0x104 )             // create someFile and open
{
  *(_WORD *)response = 0x601;
  createFileFailed_1 = (const char *)sub_180004204(v100, off_1800277F0);
  if ( *((_QWORD *)createFileFailed_1 + 3) >= 0x10ui64 )
    createFileFailed_1 = *(const char **)createFileFailed_1;
  strcpy(&response[2], createFileFailed_1);
  if ( v101 >= 0x10 )
    j_free(v100[0]);
  v101 = 15i64;
  v100[2] = 0i64;
  LOBYTE(v100[0]) = 0;
  goto LABEL_164;
}
strcpy(::someFileName, buf + 1);
openedFileStream = fopen(::someFileName, "wb");
if ( !openedFileStream )
{
  *(_WORD *)response = 0x601;
  createFileFailed = (const char *)sub_180004204(v104, off_1800277F0);
  if ( *((_QWORD *)createFileFailed + 3) >= 0x10ui64 )
    createFileFailed = *(const char **)createFileFailed;
  strcpy(&response[2], createFileFailed);
  if ( v105 >= 0x10 )
    j_free(v104[0]);
  v105 = 15i64;
  v104[2] = 0i64;
  LOBYTE(v104[0]) = 0;
  v85 = socket->__vftable_0;
  v86 = strlen(response);
  v85->sub_18001DB7C(socket, response, v86 + 1);
  return 0i64;
}
*(_WORD *)response = 0x1001;
fileStreamInitialized = 1;
v53 = 2i64;
goto LABEL_168;
```

Часть обработчика открытия файла

Работа плагинов

Инициализация плагинов

Taidoor инициализирует два встроенных плагина:

- MyPlugCmd — для исполнения команд на машине. В качестве одного из аргументов при инициализации передается путь, по которому располагается копия cmd.exe. Если необходимый ключ в win.ini не найден или копирование сорвалось, передается пустая строка.
- MyPlugInfo — для получения базовой информации о зараженной машине.

Вот как выглядит процесс загрузки нового плагина (функция представлена на рис. 3):

1. Библиотека загружается в память.
2. Программа проверяет плагин на наличие следующих экспортируемых функций:
    - Install,
    - Uninstall,
    - Proxy.
3. Программа вызывает функцию Install, передает в нее информацию о подключении к серверу управления. В качестве идентификатора выставляется полученное из Install значение.
4. После инициализации модуль добавляется в массив плагинов и, если имя файла соответствует схеме uaq*.dll, файл удаляется.

```
char __fastcall LoadPlugin(MyPlugDll *a1, SOCKET *a2, const char *libraryPath)
{
  HMODULE v6; // rax
  FARPROC Install; // rax
  HMODULE v8; // rcx
  FARPROC Uninstall; // rax
  HMODULE v10; // rcx
  FARPROC Proxy; // rax
  __int64 (__fastcall *Install_1)(SOCKET *); // rdx
  char v13; // al

  memset(a1->typeName, 0, sizeof(a1->typeName));
  strcpy(a1->typeName, libraryPath);
  v6 = LoadLibraryA(a1->typeName);
  a1->loadedLibrary = v6;
  if ( v6 )
  {
    Install = GetProcAddress(v6, "Install");
    v8 = a1->loadedLibrary;
    a1->Install = Install;
    Uninstall = GetProcAddress(v8, "Uninstall");
    v10 = a1->loadedLibrary;
    a1->Uninstall = Uninstall;
    Proxy = GetProcAddress(v10, "Proxy");
    Install_1 = (__int64 (__fastcall *)(SOCKET *))a1->Install;
    a1->Proxy = Proxy;
    if ( Install_1 )
    {
      if ( a1->Uninstall )
      {
        if ( Proxy )
        {
          v13 = Install_1(a2);
          a1->plugType = v13;
          if ( v13 )
            return 1;
        }
      }
    }
    FreeLibrary(a1->loadedLibrary);
  }
  else
  {
    GetLastError();
  }
  return 0;
}
```

Функция загрузки плагина

Каждый модуль может быть деинициализирован, чтобы предоставить место другому.

При получении команды от сервера управления Taidoor проверяет первый байт, который указывает на адресата: команда может быть предназначена либо для основного модуля, либо для одного из плагинов. Если команда должна быть передана плагину, остаток сообщения передается в функцию плагина Proxy.

Активировав плагины, Taidoor в отдельном потоке запускает функцию периодического обращения к серверу. В этой функции программа каждые 10 секунд отправляет управляющему серверу сообщение \x01\xff.

Все обращения к серверу здесь и далее создаются по следующему алгоритму:

1. Высчитывается размер данных с паддингом. Он будет равен длине сообщения + 4 байта — это значение округляется до 16, обязательно в большую сторону. Например, 16 → 20 → 32, а 12 → 16 → 32.
2. Изначальный размер записывается в первые 4 байта нового массива с вычисленным размером. Сразу после него копируются передаваемые данные (т. е. по отступу 4). Лишние байты в конце массива заполняются нулями. Таким образом данные запаковываются для симметричного шифрования.
3. Программа генерирует временный ключ из 16 символов нижнего регистра в латинском алфавите. Механизм генерации небезопасен, поскольку позволяет перебрать все возможные варианты за приемлемое время. К тому же если известно примерное время заражения, можно сократить количество вариантов.
4. Программа шифрует созданный ключ при помощи алгоритма RSA_PKCS1v1.5 с использованием криптографически стойкого ГПСЧ на ключе, взятом из параметров.
5. Программа шифрует запакованный массив при помощи алгоритма AES-128 в режиме ECB на временном ключе.
6. Программа отправляет на сервер 4 байта полного размера пакета: 256 (размер зашифрованного ключа) + размер зашифрованных данных. Если удалось отправить размер, то программа последовательно посылает 256 байт зашифрованного ключа и зашифрованные данные.

Общение плагинов с сервером управления

Включив плагины, малварь начинает получать команды от сервера управления. Дальнейшие действия вредоноса зависят от команд.

Если при получении данных от сервера возникла ошибка, программа завершает соединение и ждет столько времени, сколько установлено в параметрах (в изученном образце это 1 секунда), после чего вновь пытается подключиться к серверу.

Если сервер ответил пустым сообщением, программа это запоминает. Если сервер ответил так 300 раз подряд, программа завершает соединение и столько времени, сколько установлено в параметрах (в изученном образце это 30 минут), после чего пытается подключиться снова.

Если размер сообщения от сервера больше или равен 256 байтам, то программа действуют по следующему алгоритму:

1. Taidoor берет первые 256 байт и расшифровывает их на публичном ключе RSA с применением RSA_PKCS1v1.5. Полученные данные используются в качестве симметричного ключа для следующих данных. Использование подписи в данном случае бессмысленно, так как не защищает от перехвата управления.
2. Программа использует полученный ключ, чтобы расшифровать остаток сообщения алгоритмом AES-128 в режиме ECB.
3. В первый байт полученного массива записывается идентификатор адресата команды:
    - Если он равен 1, то остальные данные предназначаются для основного модуля и обрабатываются в отдельной функции.
    - Все остальные значения обозначают тот или иной плагин. Программа ищет его и передает ему данные. Если такой плагин не найден, программа не делает ничего.

Функциональные возможности плагинов

Встроенный модуль MyPlugCmd предназначен для запуска процессов и передачи команд консоли. При инициализации ему передается параметр, который может быть сохранен в файле `C:\Windows\win.ini`. Этот параметр указывает на копию cmd.exe, которая используется, чтобы затруднить обнаружение программы.

Модуль хранит несколько переменных в своем теле. Вот наиболее важные из них:

- <файловый_дескриптор>;
- <имя_файла>;
- <альтернативный_путь_к_cmd>;
- <шелл>.

Стоит отметить, что переменная <шелл> хранит информацию о запущенном процессе консоли, которая нужна для записи в поток ввода данного процесса и получения данных из потока вывода.

Полный список команд, выполняемых модулем MyPlugCmd можно посмотреть тут.

Идентификатор	Команда
1	Проверить, что среди запущенных процессов нет avp.exe (антивируса Kaspersky): • Если он найден, отправить \x02\x07kb и закончить обработку. • Если не найден, проверить, инициализирована ли переменная <альтернативный_путь_к_cmd>. Если нет, то записать туда cmd.exe. Затем создать процесс <альтернативный_путь_к_cmd> с перенаправленным вводом-выводом. При успешном запуске поместить информацию о процессе вместе с дескрипторами перенаправления ввода-вывода в <шелл>, а в файл C:\Windows\win.iniдобавить секцию Micros с ключом source со значением <альтернативный_путь_к_cmd>. Также запускается дополнительный поток, в котором каждые 32 миллисекунды до 4096 байт вывода запущенного процесса отправляются на сервер с идентификатором \x02\x09. При ошибке создания процесса отправить на сервер сообщение \x02\x03, при успехе — \x02\x02
4	Если <шелл> инициализирован, завершить его процесс и отправить на сервер \x02\x05. Если нет — отправить на сервер \x02\x06no shell
8	Записать остаток буфера в поток ввода <шелл>
10	Сохранить путь к файлу, переданный в сообщении, в <имя_файла>, открыть его для чтения и сохранить дескриптор в <файловый_дескриптор>. При ошибке отправить \x02\x0eCreate File Failed, при успехе — \x02\x0b
12	Сохранить остаток буфера в переменную <имя_файла>, открыть этот файл для чтения, сохранить дескриптор в <файловый_дескриптор>. Если возникла ошибка при открытии файла, отправить на сервер \x02\x0eOpen File Failed. Если он пуст — \x02\x03File Size is 0. При обеих ошибках закончить обработку команды. Если файл открыт и он не пустой, отправить на сервер сообщение \x02\x0d. Также запустить дополнительный поток, в котором содержимое файла будет отправляться на сервер в сообщениях с идентификатором \x02\x0f частями по 4096 байт с промежутком в одну миллисекунду. Остаток файла не отправлять. Завершить процесс сообщением \x02\x12 и закрытием <файловый_дескриптор>
15	Подать остаток буфера на вход процессу <шелл>, если он инициализирован

16	Обновить временные метки у файла <имя_файла>, скопировав временные метки файла C:\Windows\System32\services.exe. Отправить серверу сообщение \x02\x11
19	Закрыть <файловый_дескриптор>, удалить файл <имя_файла>
31	Создать временный файл и запустить файл, расположенный по пути из остатка буфера, перенаправляя вывод во временный файл. Если произошла ошибка при создании временного файла, отправить сообщение об ошибке \x02\x20Create result file failed. Если при запуске файла по пути из остатка буфера произошла ошибка, отправить сообщение \x02\x32CreateProcess Error:, конкатенированное с кодом ошибки. Если запуск прошел успешно, читать из временного файла по 4096 символов и отправлять их с кодом сообщения \x02\x21 каждую миллисекунду
34	Запустить файл, расположенный по пути из остатка буфера. При удачном запуске отправить серверу сообщение \x02\x32CreateProcess succ, при ошибке — сообщение \x02\x32CreateProcess Error:, конкатенированное с кодом ошибки.
37	Заменить <альтернативный_путь_к_cmd> остатком полученного буфера. Отправить серверу сообщение \x02\x26

Встроенный модуль MyPlugInfo может собирать и отправлять на сервер информацию об IP-адресах и MAC-адресах сетевых интерфейсов, идентификатор текущего процесса, а также идентификаторы заражения. Еще он умеет выполнять команду 11 основного обработчика.

Команды с идентификаторами можно найти тут

Значение байта	Команда
1	Собрать информацию об IP-адресах и MAC-адресах сетевых интерфейсов, идентификатор текущего процесса, а также идентификаторы заражения (устанавливаются при инициализации плагина). Отправить всю эту информацию на сервер с кодом \x03\x02
3	Выполнить команду 11 основного обработчика

Индикаторы компрометации (IoC) обоих вариантов svchost.dll

Файл	Параметр реестра
%ALLUSERSPROFILE%\\Application Data\\Microsoft\\~svc_.TMp	RValue в ключе SOFTWARE\Microsoft\Windows NT\CurrentVersion

Автор: @rumata888 Иннокентий Сенновский
https://twitter.com/rumata888