

как бэкдор для Linux проник в государственные инфраструктуры

rt-solar.ru/solar-4rays/blog/4861/



Оглавление

Весной 2023 года сотрудники ИТ-компании, предоставляющей услуги различным секторам российской экономики (в основном государственным органам), зафиксировали дампы хешей пользователей с контроллера домена. Дамп был выполнен при помощи утилиты “impacket-secretsdump”, запущенной на Linux-хосте, после чего системные журналы этого хоста были очищены. Подозрительные события побудили сотрудников пригласить на расследование команду Solar 4RAYS. Это не новые, уникальные техники, DFIR-специалисты часто сталкиваются с подобной активностью, однако вредоносное программное обеспечение, которое мы обнаружили в рамках данного инцидента, заинтересовало своей уникальностью.

GoblinRAT (так мы именуем обнаруженное семейство) использовался в самых скрытных атаках, которые нам довелось расследовать. Вредоносная активность, о которой пойдет речь в этом посте, длилась более двух лет. Заражено по меньшей мере четыре инфраструктуры различных компаний, ориентированных на предоставление услуг государственным организациям.

- Большая часть функциональности GoblinRAT решает единственную задачу – скрывать присутствие ВПО в системе.
- Для коммуникации с C2 GoblinRAT использует взломанные сайты легитимных организаций и DDNS. При этом почти каждый найденный исполняемый файл содержал уникальный адрес.
- Создатели этого вредоносного ПО весьма оригинальны: в процессе расследования мы заметили, что для каждого нового зараженного хоста атакующие применяли уникальные названия задач планировщика, файлов, библиотек и служб для закрепления в системе.

Благодаря обнаруженным артефактам мы смогли проследить историю развития GoblinRAT с 2020 года, однако нам не удалось обнаружить широкого распространения вредоноса. Более того, наши коллеги из других компаний с глобальными сетями сенсоров не обнаружили ничего похожего в своих коллекциях. Мы постоянно отслеживаем появление новых образцов ВПО на публичных площадках для обмена информацией об угрозах, но исследуемое семейство никогда не загружалось на подобные ресурсы.

В рамках данного отчета вы узнаете:

- Что представляет собой GoblinRAT и в чем его особенность.
- Почему обнаружить подобный вид ВПО было сложно, а в некоторых случаях – практически невозможно.
- Как программа для администрирования атор иногда может стать хорошей альтернативой SIEM.
- Что может скрывать Zabbix-агент и другие легитимные сервисы на ваших серверах.

Первая встреча

В ходе расследования мы выяснили, что для удаления системных журналов злоумышленники использовали утилиту “shred”, а дампы хешей пользователей с контроллера домена был выполнен при помощи инструмента “impacket-secretsdump”. Для эксфильтрации данных с хоста использовались “scp” и “curl”. Весь используемый инструментарий размещался в директории “/dev/shm”. Данный каталог используется для работы с разделяемой памятью и полностью расположен в оперативной памяти, что позволяло атакующим не оставлять следов на жестком диске, ведь после перезагрузки ОС каталог полностью очищается. При восстановлении цепочки действий злоумышленников и анализе промежуточных хостов стало понятно, что атакующие хорошо знакомы с инфраструктурой и, до момента пока их не обнаружили, успели заполучить различные административные учетные записи.

В первые несколько часов расследования мы проанализировали десятки хостов, но не нашли никакого дополнительного вредоносного программного обеспечения для закрепления в инфраструктуре. Учитывая накопленные знания об атакующих, мы сформулировали гипотезу о возможных точках закрепления, использованных злоумышленниками: это могли быть серверы мониторинга, оркестрации, терминального доступа и различные сервисы из DMZ-сети. Получив их для анализа, в оперативной памяти одного из серверов, мы обнаружили следующий набор подозрительных строк:

 Информация из дампа памяти
Информация из дампа памяти

Первый подозрительный аспект – присвоение результата работы команд, отображающих информацию об используемой на хосте операционной системе, в переменную “A”. Такое поведение свойственно скрипту по сбору информации о системе с обфускацией переменных.

Второй подозрительный аспект – изменение значения переменной окружения “HISTFILE” на “/dev/null”, что позволяет отключить логирование команд в активной сессии командного интерпретатора. Описываемые строки находились в неразмеченном пространстве и не были связаны с каким-либо процессом. Изначально исследующий дампы памяти аналитик предположил, что это артефакты работы ВПО, похожего на активно обсуждаемый на тот момент BPFDoor.

Однако в ходе анализа следующего дампа памяти уже был найден вредоносный процесс, маскирующийся под агента системы мониторинга Zabbix. На скриншоте ниже представлен вывод плагина “psaux” фреймворка [volatility3](#), в котором можно заметить подозрительный процесс с идентификатором 1079803, у которого имя процесса “zabbix_agent” и путь “/usr/bin/zabbix_agentd” в аргументах процесса отличаются на одну букву – “d”. От легитимного агента Zabbix его дополнительно выделяет отсутствие дочерних процессов.

После анализа образа жесткого диска мы обнаружили следующий systemd-сервис `"/usr/lib/systemd/system/zabbix-agentd.service"`:

```
[Unit]
Description=Zabbix agent service.
After=network.target

[Service]
Type=forking
ExecStart=/usr/sbin/zabbix_agent

[Install]
WantedBy=multi-user.target
```

Содержимое вредоносного systemd-сервиса `zabbix-agentd.service`

Внимательный читатель сразу заметит что название сервиса и запускаемый им файл (параметр `ExecStart`) отличаются на одну букву "d", как и было видно в дампе памяти. Это не случайно, ведь файл по указанному пути – не агент Zabbix, а вредоносное программное обеспечение, которое мы назвали **GoblinRAT**.

Данное семейство ВПО написано на языке программирования Golang и обладает различной функциональностью, ориентированной на удаленный доступ и сокрытие собственного присутствия на системе. Подробно функции ВПО рассмотрены в разделе ["Технический анализ"](#).

Для затруднения обнаружения (помимо подражания названию сервиса легитимного ПО для мониторинга Zabbix) GoblinRAT модифицировал имя процесса путем предварительной подготовки и замены нулевого аргумента `"argv[0]"` на значение `"/usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf"`. Подробно о данной и других техниках сокрытия имени процесса мы рассказывали на конференции [OFFZONE 2024](#).

Из конфигурационного файла, который был зашит в найденный образец ВПО, мы достали серверы управления: `"qfilling[.]jinstanthq.com:443"` и `"37.120.247[.]182:443"`. Добавив полученные индикаторы и Yara-правила в наш сканер безопасности, мы просканировали инфраструктуру, чтобы выявить все возможные точки закрепления. И скоро мы обнаружили еще несколько зараженных серверов.

Примечательно, что атакующие не выбрали какой-то один способ закрепления, а в каждом случае подстраивали его под окружение сервера: учитывался набор установленного ПО и тип операционной системы. Например, на одном из серверов GoblinRAT имитировал под легитимный сервис менеджера подписок Red Hat (`rhsm`) (`rhsm`). Атакующие разместили ВПО по пути `"/var/lib/rhsm/rhsm"` (легитимный файл располагался по пути `"/usr/libexec/rhsm"`) и для закрепления использовали systemd-сервис `"/usr/lib/systemd/system/rhsmcertd.service"`:

```
[Unit]
Description=Enable periodic update of entitlement certificates.
After=network.target

[Service]
Type=forking
ExecStart=/var/lib/rhsm/rhsm

[Install]
WantedBy=multi-user.target
```

Содержимое вредоносного systemd-сервиса `rhsmcertd.service`

Имя процесса при запуске заменялось на `"/usr/bin/rhsmcertd"`.

На одном из зараженных серверов, выступающих в роли веб-сервера, атакующие скрывали GoblinRAT под именем популярной утилиты `"memcached"`, предназначенной для кэширования данных в оперативной памяти на основе хеш-таблицы. ВПО разместили по пути `"/usr/bin/memcache"` с закреплением через cron-задачи:

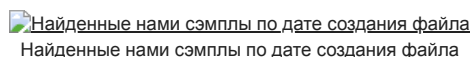
Имя процесса заменялось на `"/usr/bin/memcached -m 64 -p 11211 -u memcache -l 127.0.1.1 -P /var/run/memcached/memcached.pid"`.

Ещё один интересный образец располагался по пути `"/usr/sbin/chronyd_debug"`. При запуске он изменяет имя процесса на `"/usr/bin/vmtoolsd"`. Сэмпл `chronyd_debug` в качестве C2 хранит два адреса: `chronyd[.]tftpd[.]net` и `37.120.247[.]182`. Адрес `chronyd[.]tftpd[.]net` резолвится в `5.63.154[.]23`. В свою очередь, этот IP-адрес в PDNS имеет следующие доменные имена: `test[.]pink-apple[.]ru`, `ftp[.]pink-apple[.]ru`, `www[.]pink-apple[.]ru`, `pink-apple[.]ru`, `www[.]test[.]pink-apple[.]ru`.

Таким образом злоумышленники создали отдельный домен, похожий на название исполняемого файла и сервиса, который работал на зараженной машине. Второй C2 имел доменное имя `chronyd[.]com`, что похоже на название известного сервиса. Однако такой домен отсутствует среди известных нам сэмплов. Здесь можно выделить крайне высокий уровень подготовки к заражению конкретной машины. Помимо мимикрии под конкретный сервис на уровне названия файла, атакующие даже не поленились зарегистрировать отдельное доменное имя. Они потратили немало сил на разведку окружения целевой инфраструктуры.

Новые сэмплы и Hands on-активность атакующих

Благодаря полученным из конфигурационных файлов серверам управления нам удалось выявить три новые организации, зараженные семейством `GoblinRAT`. Все три работают в государственном секторе. Во всех этих случаях установить источник заражения не удалось. Некоторые из систем, на которые вели следы, уже вывели из эксплуатации, а в некоторых случаях глубина хранения журналов не покрывала необходимый период. Однако собранные данные указывают, что предположительная дата первичного заражения - октябрь 2020 года. На тот момент `GoblinRAT` был написан на версии `Golang 1.13.8` (выпущена 12.02.2020) и выступал в качестве сервера, прослушивая все пакеты приходящие на определенный интерфейс, который указан в конфигурации, а для успешного соединения требовалась реализация техники `Port knocking`. Вероятно, данная версия использовалась атакующими в закрытых частях инфраструктур, где доступ серверов к интернету полностью отсутствовал, либо был ограничен белыми списками, и взаимодействие с сервером управления можно было организовать только через проксирование трафика (подобную активность мы наблюдали в рамках первого расследования). Ниже на схеме показали все найденные сэмплы `GoblinRAT`, отсортированные по дате появления файла на зараженной системе. Из интересного можно увидеть, что серверная версия `GoblinRAT` является самой ранней версией — 2020 года появления на системе. Также используется более старая версия `Golang`. Остальные же компилировались одной версией языка программирования `Go`.

Найденные нами сэмплы по дате создания файла
Найденные нами сэмплы по дате создания файла

Нам встречались версии, живущие только в памяти, которые не оставляли никаких следов присутствия ВПО на жестком диске. В одном из таких случаев атакующие размещали ВПО по пути `"/dev/shm/cron"`. После запуска `GoblinRAT` изменял имя своего процесса на `"/usr/bin/cron -f"` и удалял файл из `"/dev/shm"`. Отличить данный процесс от легитимного сервиса `cron` можно по нескольким признакам:

1. Слишком высокое потребление виртуальной памяти, что не свойственно легитимному сервису `cron`:



2. Переменные окружения, свойственные командной оболочке, нежели процессу `cron`:

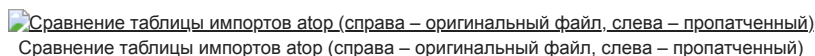


3. Нестандартная рабочая директория процесса – `"/tmp"`:



В случае с данным образцом закрепления на системе не происходило. Возможно, это связано с тем, что сервер был частью системы виртуализации с аптаймом более года и мог выступать промежуточной точкой в цепочке заражения.

Самым сложным с точки зрения обнаружения вредоноса был образец, собранный в библиотеку с именем `"libcuxorg.so.50.1.2"`. Атакующие пересобрали `ator` и добавили в таблицу импортов `GoblinRAT` в качестве библиотеки. На рисунке ниже вы можете увидеть сравнение таблицы импортов оригинального(справа) и пропатченного ПО(слева)

Сравнение таблицы импортов `ator` (справа – оригинальный файл, слева – пропатченный)
Сравнение таблицы импортов `ator` (справа – оригинальный файл, слева – пропатченный)

Этот образец интересен еще и тем, что атакующие, решив внедриться в процесс `ator`, упустили тот факт, что эта утилита предназначена для мониторинга процессорной нагрузки и через заданные промежутки времени сохраняет список всех работающих процессов с их параметрами. При исследовании данного сервера мы столкнулись с проблемой: SIEM-система на момент заражения проходила этап импортозамещения, и некоторые события запусков процессов и исполнения команд по разным причинам были утеряны. Однако слепки процессов, которые `ator` создавал каждые 60 секунд, и его журналы, восстановленные с диска, позволили нам понаблюдать за активностью атакующих в момент их первого входа в систему и закрепления на ней. По результатам анализа журналов `ator` мы сформировали большой таймлайн. Рассмотрим его самые интересные фрагменты.

1. На рисунке ниже представлена активность атакующих по изучению системы и первая попытка закрепления. Предположительно сначала атакующие пытались заменить `ator` на `GoblinRAT`, и подменили легитимный файл `"/usr/bin/ator"`.

2. Python-скрипт, запущенный процессом PID 16223 (рисунок выше), используемый атакующими для сбора информации о системе, поставлялся в закодированном в base64 формате. В листинге ниже представлен код скрипта после декодирования:

```
import sys, subprocess as s, json as j, random as r
from base64 import b64decode as b
l=["w","last","lastlog","id","uname -a","netstat -apn|ss -apn","ip a","ip n","ip r","env","iptables -S","iptables -t nat -S","ls -
alh /home /home/* /home/*/*.history /root / /dev/shm /tmp /var/tmp /opt","cat /etc/passwd","cat /etc/shadow","cat /etc/group","cat
/etc/sudoers","cat /etc/resolv.conf","cat /etc/fstab","cat /etc/hosts","hostname","mount","df -kh","crontab -l","ps -
efH","date","ls -la /home/*/.ssh/","ls -lah /root/.ssh","lsmod","sestatus","grep -r '@' /etc/rsyslog* | grep -v '#',"tail -n 100
/var/log/secure","cat /proc/cpuinfo","free -m","find /root/.ssh -type f -ls -exec cat {} \;","find /home -maxdepth 3 -iname
'id_rsa' -ls -exec cat {} \;","find /home -maxdepth 3 -iname '.bash_history' -ls -exec cat {} \;","cat
/root/.bash_history","systemd-detect-virt","hostnamectl","lsblk","lspci"]
r.shuffle(l)
d={}
for c in l:
if sys.version_info.major<3:
    d.update({b(c):b(s.Popen(c.strip(),stdout=s.PIPE,shell=True,stderr=s.STDOUT).stdout.read())})
else:
    d.update({b(c.encode('utf-
8')).decode():b(s.Popen(c.strip(),stdout=s.PIPE,shell=True,stderr=s.STDOUT).stdout.read()).decode())})
print(j.dumps(d))
```

3. После подмены легитимного файла atop атакующие, как мы полагаем, опасались, что однажды администраторы системы могут заметить отсутствие журналов atop, и решили перейти к использованию вредоносной библиотеки "libcuxorg.so.50.1.2". На схеме ниже отражены действия, выполненные через интерактивную оболочку "/bin/sh", созданную в результате отправки соответствующей команды с сервера управления. В рамках данной интерактивной сессии атакующие выполнили ряд команд (самые интересные выделены оранжевым цветом). В 13:21:06 была вызвана команда touch, время выполнение которой совпадает с временной меткой ctime файла /usr/bin/atop. В 13:21:27 был выполнен очередной запуск команды touch, по времени совпадающий с временной меткой ctime файла /usr/lib64/libcuxorg.so.50.1.2, а mtime метка этого файла совпадает с некоторыми из той же директории, что указывает на ее подмену.

 Настройка вредоносного сервиса и подмена меток времени на файлах GoblinRAT
Настройка вредоносного сервиса и подмена меток времени на файлах GoblinRAT

 Изменённая mtime метка вредоносной библиотеки, совпадающая с другими файлами
Изменённая mtime метка вредоносной библиотеки, совпадающая с другими файлами

По результатам данных расследований у нас сложилось двоякое впечатление. С одной стороны, в действиях атакующих нет каких-то новых, уникальных техник – для достижения цели в основном использовался предустановленный в ОС набор утилит:

- ssh в Linux и netsh portproxy в Windows – для проксирования трафика;
- curl и scp – для эксфильтрации;
- python – для запуска скрипта по сбору информации об атакуемой системе;
- shred – для удаления логов.

С другой – мы видим скрупулезный подход к закреплению и мимикрии под легитимные сервисы на хосте, использование уникального бэкдора, который нигде больше не встречается и длительное присутствие в инфраструктуре. Все действия кажутся простыми, но в тоже время они выполнены профессионально, что принесло злоумышленникам соответствующий результат.

Другие способы мимикрии и закрепления перечислены в [Приложении 1](#). Далее детальнее разберем механизмы работы GoblinRAT.

Технический анализ GoblinRAT

GoblinRAT реализован на языке программирования Go. Функциональность ВПО включает:

- интерактивную командную оболочку;
- выполнение одиночной команды;
- работу с файловой системой (перечисление файлов в папке, загрузка и выгрузка файлов с зараженной системы, копирование и перемещение файлов, архивирование и разархивирование файлов);
- удаление файлов с предварительным перезаписыванием их данных;
- работу в роли прокси (SOCKS5);
- работу в режиме мультиплексера (Yamux);
- способность мимикрировать под легитимный процесс (изменяется имя процесса).

Мы обнаружили две реализации данного ВПО:

- в виде клиента, который сам подключается к серверу управления,
- в виде сервера, ожидающего подключения для выполнения команд.

Серверная часть реализует технику Port knocking для сокрытия своего присутствия на системе. Для того, чтобы GoblinRAT начал прослушивать порт в ожидании подключения, необходимо отправить любой IP-пакет на строго заданную в конфигурационном файле последовательность портов. Для работы данного функционала в конфигурационном файле задается интерфейс, прослушивая который, GoblinRAT будет проверять входящие IP-пакеты и сверять их с последовательностью из конфигурационного файла.

ВПО обладает рядом функций для скрытого присутствия на системе. Известные нам сэмплы для закрепления использовали пути и названия файлов, которые мимикрируют под легитимный инструментарий. Он вполне мог бы быть установлен легальными администраторами сервера, а в некоторых случаях работал параллельно с мимикрируемым ПО. У ВПО нет функций автоматического закрепления: всякий раз атакующие закрепляли его вручную, что недвусмысленно указывает на таргетированный характер атаки.

Используемые внешние библиотеки:

- [go-socks5](#)
- [yamux](#)

Для противодействия анализу используется средство Gobfuscate: зашифрованы хранимые строки и заменены на случайные значения имена функций. На сегодня наиболее популярным обфускатором для языка Go является garble, а Gobfuscate встречается крайне редко, к тому же проект прекратил свое развитие три года назад. Конфигурация зашифрована при помощи AES GCM.

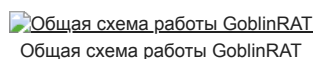
Описание последовательности работы вредоносного кода

Описание производится на основе "клиентской" реализации вредоносного файла:

MD5	3f9b1b506dfab7a5cc32004a45ed780d
SHA1	2167f0a1b7f8a3358c416d59124333ce70c2a137
SHA256	b074749f160453053989277e2eee3d1f31d618c0813f6379415a4727ed856806
File name	zabbix_agent
File size	6.42 MB (6729728 bytes)

Работа вредоносного кода состоит из следующих этапов:

1. работа с конфигурацией;
2. сокрытие активности;
3. подключение к серверу управления;
4. обработка команд в случае успешного подключения к серверу управления.

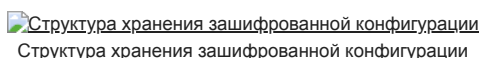


Этап расшифровки конфигурации

Конфигурация хранится в исполняемом файле в зашифрованном виде. Алгоритм шифрования – AES, режим шифрования – GCM. Для расшифровки в этом режиме требуется три параметра:

- попсе, который хранится в составе зашифрованной конфигурации;
- tag, который в данном случае представляет из себя 16 нулевых байт;
- ключ шифрования, который хранится в исполняемом файле.

Зашифрованная конфигурация состоит из двух значений: первые 12 байт занимают значение попсе, необходимое для расшифровки в режиме шифрования GCM. Остальные данные – зашифрованная конфигурация. Схема структуры хранения зашифрованной конфигурации представлена на рисунке ниже.



Расшифрованная конфигурация представлена в виде json-данных. Вот список ключей, которые содержатся в структуре конфигурации:

- Base
 1. Shell – путь до команды, которая выполняется для вызова интерактивной оболочки и выполнении команд в шелле;
 2. SystemInfoCmd – команда, которая выполняется для сбора информации о системе;
 3. KillSwitchDelay – Время в днях, по истечении которого ВПО удалит себя из системы в случае отсутствия подключения к авторизованному серверу управления;
 4. AuthSecret – ключ для отключения самоуничтожения;
 5. SocksTimeout – время ожидания подключения к стороннему серверу;
 6. FirstCmds – команды, которые выполняются при сборе данных о системе;
 7. HostFingerprint – значение для генерации UID;
 8. AltProcessName – название процесса для маскировки;
 9. TempFilePath – путь до временной директории;
- ClientKey – ключ для аутентификации управляющего сервера;
- ClientPEM и RootCAPEM – сертификаты для аутентификации управляющего сервера;
- RemoteServers – список адресов управляющих серверов;
- RetryInterval – интервал в минутах для попытки подключения к управляющему серверу;
- Proxy – адрес прокси-сервера;
- HTTPProxyUserAgent – значение User-Agent при подключении к прокси серверу.

После расшифровки конфигурации происходит обработка некоторых исходных значений, а именно: сертификаты загружаются в локальное хранилище программы для последующей аутентификации сервера и прохождения аутентификации клиента на сервере. Далее устанавливаются настройки прокси.

Соккрытие активности

Соккрытие активности GoblinRAT состоит из следующих этапов:

1. Переименование процесса;
2. Удаление бэкапов. Проверяет наличие файла /path/to/file/filename.bak;
3. Высчитывает свой хэш при помощи алгоритма sha256 и сохраняет его в глобальную переменную (используется в команде для сбора информации с компьютера);
4. Запускает в отдельном потоке функцию с таймером на самоуничтожение;
5. Инициализирует логгер с выводом в /dev/null вместо stdout.

Для переименования процесса программа сверяет текущие аргументы и наименование процесса со значением из конфигурации. Если имя процесса не совпадает с полем AltProcessName, то ВПО перезапускает себя же, подменяя имя процесса. Это возможно благодаря особенности функции создания процесса `"execve(char *pathname, char * argv[], char *envp[])"`, в которой в аргументе pathname указываются имя исполняемого файла для запуска, а аргументы, включая отображаемое имя исполняемого файла, передаются в argv. Поэтому в списке процессов показывается измененное имя процесса, мимикрирующее под легитимное ПО. После запуска нового процесса, вредонос завершает текущий процесс.

Далее GoblinRAT удаляет файлы с расширением .bak, которые могут являться артефактами установки вредоносного файла. Впрочем, файлов с таким расширением в процессе расследования мы не обнаружили.

Отдельно остановимся на процессе удаления, который применяется во всех операциях удаления файла. Для удаления файлов ВПО перезаписывает четыре раза содержимое файла случайными данными. Для этого читает 0x100000 байт из файла /dev/urandom и записывает их в удаляемый файл – тоже блоками по 0x100000. После каждой итерации записи синхронизирует содержимое файла с данными на диске при помощи Sync. После уже перед непосредственным удалением с файловой системы, файл еще раз перезаписывается нулевыми байтами, после чего “спит” одну секунду и удаляет файл.

Время самоуничтожения берется из конфигурации, в которой хранится время в днях. Таймер может быть либо отложен, либо отключен отдельными командами. Для отключения требуется знать ключ из конфигурации AuthSecret.

Подключение к С2

Для подключения к серверу ВПО последовательно проходит по всем адресам из конфигурации. Если ни один из серверов не ответил, то ожидает определенное время, указанное в конфигурации в ключе RetryInterval.

При успешном подключении сначала производится TLS-рукопожатие с двусторонней аутентификацией. Для этого в исполняемом файле зашиты два сертификата: публичный ключ сервера (ClientPEM, RootCAPEM) и приватный ключ клиента (ClientKey). После чего ВПО ожидает подключения Yatux в рамках установленного TLS-соединения. В случае успеха начинается обработка команд, пришедших с сервера.

Обработка команд

Команда – это один байт, который интерпретируется как однобайтовое целое число. Сначала GoblinRAT считывает именно один байт. Если передано свыше одного байта, то подключение прерывается. Ниже в таблице представлен список возможных команд. Дальше уже идет передача последующего взаимодействия выбранному обработчику команды, который считывает необходимое количество байт. Обработчики могут иметь подкоманды, которые тоже представляют из себя однобайтовое число. Ниже можно увидеть псевдокод на примере команды 1.11.0 — приблизительное представление схемы взаимодействия с C2 сервером.

```
def handle_commands(s: socket_handler):
    cmd = s.recv(1)
    match cmd:
        case b'\x01': handle_command_1(s)
        ...

def handle_command1(s: socket_handler)
    sub_cmd = s.recv(1)
    match sub_cmd:
        case b'\x0b': handle_command_1_11(s)
        ...

def handle_command_1_11(s: socket_handler)
    sub_sub_cmd = s.recv(1)
    match sub_sub_cmd:
        case b'\x00': enum_dir(s)
        case b'\x01': enum_proc(s)
        ...
```

Команда	Подкоманда	Описание
0		Отключение самоуничтожения после прохождения аутентификации
2		Перевод текущего соединения в SOCKS5 проху. Для реализации используется go-socks5
7		Завершение соединения. Есть возможность также завершить работу исполняемого файла
12		Открытие слушающего порта. Общение на этом порте также обернуто в мультиплексер Yamux
14		Команда по работе с конфигурацией
	0	Возвращение значения определенного поля из конфигурации
	1	Полная замена конфигурации
	2	Замена конфигурации в самом бинарном файле. Копирует себя с расширением lib, после чего запускает новый процесс и самоуничтожается
	3	Возвращение всей конфигурации в ответ на запрос
	4	Переподключение для применения новой конфигурации без перезапуска самого процесса
	5	Замена по определенному ключу значения в конфигурации
1 или другое однобайтовое число		Команда для управления зараженным компьютером.
	0	Получить информацию о зараженной системе
	1	Выполнить команду. Для этого берется значение из конфигурации Base.Shell
	2	Интерактивная командная оболочка

3	Перечисление файлов в директории
4	Смена текущей директории
5	Отправка zip-архива на зараженную систему и его распаковка
6	Архивирование файла при помощи zip и отправка его на сервер
7	Сделать снимок экрана
8	Перенаправление данных на другой сетевой адрес
9	Удаление файла. При удалении, файл сначала заполняется случайными байтами
10	Архивирование и разархивирование файлов
11	Работа с файловой системой. Имеет встроенные команды
11.0	Перечисление файлов. Возможно применить glob (/*)
11.1	Перечисление запущенных процессов. Сбор информации происходит из директории /proc с чтением данных из файлов stat, cmdline и exe
11.2	Копирование файла
11.3	Переименование файла
11.4	Чтение файла

Самоуничтожение

Функция самоуничтожения работает только в том случае, если в значении ключа KillSwitchDelay из конфигурации указано положительное число. Для самоуничтожения ВПО получает информацию о файле: размер в байтах и имя. После получения данных ВПО удаляет свой файл, выполняя следующую команду:

```
sleep 5; for n in $(seq 8); do dd if=/dev/urandom of='%[1]s' bs=%[2]d count=1 conv=notrunc; done; dd if=/dev/zero of='%[1]s' bs=%[2]d count=1 conv=notrunc; rm '%[1]s'
```

Данная команда сначала ожидает пять секунд, после чего перезаписывает файл восемь раз случайными байтами. В конце еще раз перезаписывает файл нулевыми байтами и удаляет его .

Если при получении информации возникла какая-либо ошибка, то выполняется следующая команда:

```
sleep 5 && shred -ufz '%[1]s' || rm '%[1]s'
```

bash команда "shred -ufz" делает примерно то же самое, что и команда выше. То есть сначала перезаписывает случайными байтами файл, а затем "затирает" нулями. После этой процедуры файл удаляется.

Особенность серверной версии

Особенность серверной версии ВПО заключается в отсутствии постоянного соединения с сервером управления. Серверный GoblinRAT ожидает подключения от оператора, прослушивая заданный в конфигурации интерфейс, и реализует технику Port Knocking: ожидает получения пакетов на определенную последовательность портов, после чего открывает порт для установки соединения. Таймаут между пакетами — десять секунд, после чего оператору необходимо заново отправлять верную

последовательность портов. Далее срабатывает уже знакомый нам алгоритм двусторонней аутентификации TLS и преобразование установленного соединения в Yaptix-сессию. Пример конфигурации серверной версии приведен в [Приложении 3 Примеры конфигурации](#).

C2

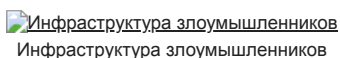
В качестве управляющего сервера злоумышленники пользовались легитимными взломанными сервисами, которые располагались на хостингах и обычно являлись лендингами (одностраничный сайт без взаимодействия с пользователем).

Для установления взаимодействия с C2 не открывался новый порт, а использовался порт 443, работающий параллельно с сайтом. Учитывая, что самый ранний “серверный” сэмпл умел “прослушивать” интерфейс, а для установки соединения используются “защитные” сертификаты, злоумышленникам не составляло труда разграничивать оригинальный трафик сервера и вредоносный трафик с зараженных машин. То есть у них имелся опыт работы с “сырыми” сокетами для обеспечения маршрутизации.

Пару примеров.

1. В сэмпле libiculxorg.so.50.1.2 в качестве C2 установлен IP 185.146.157[.]230. На момент публикации отчета этот адрес уже не отвечает на запросы ВПО, однако на нем замечен сертификат на доменное имя server.bishenov.com.
2. Исполняемый файл cron хранил в качестве C2 IP 194.58.100[.]120. Согласно данным pdns, домен mdconsultant.ru и его поддомены резолвились в этот IP.

Полное исследование на сентябрь 2024 года по открытым источникам найденных нами C2 можно изучить на рисунке ниже.



Функциональность GoblinRAT на первый взгляд не выглядит особенной. Напротив, она стандартна: интерактивный шелл и возможность проксирования трафика. Также отсутствуют какие-либо техники обхода защитных решений, а кроме того, применяется крайне распространенный метод обфускации кода. Однако есть ряд особенностей, которые не дают поместить GoblinRAT в разряд обычных вредоносов. Вот почему.

GoblinRAT обладает специфическими функциональными особенностями: таймер на самоуничтожение, мимикрия процессов с возможностью устанавливать вид процесса в конфигурации, а также применение техники Port knocking в серверной версии GoblinRAT. Всё это выделяет вредонос на фоне прочих. Он не автоматизирует атаку, но даёт операторам все возможности действовать скрытно.

Функциональность скрытности GoblinRAT эволюционировала с течением времени. Это видно по последним двум сэмплам из [таймлайна](#): libiculxorg.so.50.1.2 и cron. Первый использовался в качестве библиотеки и не создавал лишних процессов, второй – уже не оставлял следов на жестком диске и работал без закрепления на системе. На этом примере отлично прослеживается развитие ВПО в сторону сокращения количества артефактов присутствия.

Мы считаем, что GoblinRAT разрабатывался для проведения максимально скрытных и человекоуправляемых операций по кибершпионажу в тщательно выбранных инфраструктурах.

Атрибуция и выводы

В абсолютном большинстве наших расследований мы имеем представление о том, с кем имеем дело. Характерные тактики, применение известных по другим атакам вредоносов и серверной инфраструктуры, специфический выбор цели и некоторые другие параметры позволяют предположить регион происхождения атакующих и их принадлежность к той или иной группировке или кластеру вредоносной активности.

Инциденты, в которых мы обнаружили GoblinRAT – иного толка. На сегодняшний день ни мы, ни коллеги по индустрии, с которыми мы поделились информацией об этом вредоносном ПО, не обнаружили каких-либо артефактов, указывающих на его происхождение.

По совокупности признаков можно сказать, что пока это самая сложная целевая атака из тех, что нам доводилось расследовать. Для ее реализации необходимы высокий уровень знаний устройства Linux-систем и сетевых коммуникаций, а кроме того, – большое количество времени для проведения самой атаки. Сам по себе GoblinRAT не является особенно сложным ВПО. В отличие от многих аналогов, он не приспособлен для автоматического закрепления и других действий по продвижению в атакованных инфраструктурах. Атаки, в которых был задействован GoblinRAT, велись вручную, а большая часть его функциональности направлена на то, чтобы помочь атакующим скрывать их присутствие в системах как можно дольше.

Основываясь на имеющихся данных, мы можем предположить, что операторов GoblinRAT интересует конфиденциальная информация, хранящаяся на серверах государственных органов и их подрядчиков (в том числе в сегментах сети с ограниченным доступом в Интернет).

GoblinRAT может быть либо операцией одной из известных прогосударственных группировок кардинально обновивших свои инструментарий и тактики, либо – свидетельством существования новой группировки (прогосударственной или профессиональных наемников), либо – делом рук мотивированного и крайне профессионального взломщика-одиночки.

Данные, которые добавили бы очков какой-либо из этих версий, нам еще предстоит обнаружить. В том числе эту цель мы преследуем, публикуя данный отчет: мы призываем ИБ-сообщество совместно разгадать происхождение и мотивы операторов GoblinRAT. Пока же, следуя нашей собственной системе таксономии, мы присвоили активности, связанной с GoblinRAT, наименование NGC2140.

Приложение 1. IOCs

File hashes

MD5

3f9b1b506dfab7a5cc32004a45ed780d
12ce9cd39153064ed7bcdd775cc2c469
04ea8d70b5a449d7db05f9e7be37bfff6
f6e689a238240b483d2652b53bc8b808
fb7e2bf770f83d39df81789ab881abfa
ca36b1a673277c2db1995d736bada389
c1028c62e8538b29656872d48863f62b
653a34f744917c3acd4969358f232685
0a762c0a6802545dc74c9852705b1cf3
5a517596ef9cfff95ea755cdb93a51bf9
cc5fdeaab47f63e54340adbf1dc8f61d
cadaffae0d7e9a1a94735648b713a47e
3270a8c8e88518cdf8d0de94beb3a73f

SHA1

2167f0a1b7f8a3358c416d59124333ce70c2a137
9d5efc9a018e29c4f6960bfcd5e53d9daa91cd76
d964b1c6a076858570b4fe893b9d7e0295eca0b1
478a4fc65c89682f9847dcbf1721e00de7e2c6f2
202bdc8c8be32c7ca857cb01a854bc1c5f1528d1
74fcaef8aedfa626ec3abc95bae348cfc87c02d4
b2a4c93d2f3c101a0e6b1b57d6fb95088a54f18f
7f3f85c062ca26cb4c20513babd98cc15eb2198a
124870723f6c408e18ad32a88212bc45b3767489
56b8e7f399578097a5b8e181e316ab350f52cc52
438e49080d011d99e91ab0d644232b8e814c2bc8
af4b404a05dbe9e336179e1423d0817d36a7c043
627e6e30169c2bd7e7ca3262b81dae57ba7734c7

SHA256

b074749f160453053989277e2eee3d1f31d618c0813f6379415a4727ed856806
e3e68c6f9635a4745b6a97fe97fb271dec7e27415d8f42c0c87c6f8ca302e3d0
890c4dea46b45d6f86dc87d953a7370c816d614b5d3e494c285a7260839f9ed6
fcb989f128ee9fe7daa68640b57fd9dca2bf431a9324a08882803805989c2013
9fe8eb5c81366d20952e0682f9d2de55775fb9c04355e96e6d45415978d797b6
b9ba5f3d56330647d19abbe94bd9f62f944503e1fc7ea8771bddaac534a9abe3
8babfd28ee7d0a08515fff5353000df05bdf84a262aeb9837094593d4a992e79
aa475cd6c77b044ea08cec3fa96eafbfff2d651b9b671dfa55903238f25c32544
e274d528dc35b3e693d8dceb7ff657513a93c7d8ab81e855f905d5d06c9ae25f
048e722e883eada9577c2f66a11db9eae7c5a39ce18602c21ea71676725baed8
08febd2a7cee9d17709a953c064da507909420eb0e2eea9294b470348091cba2
690f5031deede7d3357d0ca24c89866ae8c60e6c63b3a2c8bba813a6ac10ae5b
f80aa6aae40f45765ec14f56fc046e669c93f51faf91adfeafa4713a99f4f390

Используемые названия и сервисы для закрепления

Имя сервиса/задачи	Отображаемое имя процесса	Запускаемый файл
rhsmcertd.service	/usr/bin/rhsmcertd	/var/lib/rhsm/rhsmcd
/etc/cron.hourly/0anacron	gpg-agent --homedir /var/cache/dnf/akopytov_sysbench-source-ce4388a64cd951b5/pubring --use-standard-socket --daemon	/usr/bin/psupply
zabbix-agentd.service	/usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf	/usr/sbin/zabbix_agent

pacemaker_pengine.service	/usr/sbin/pacemakerd -f	/usr/libexec/pacemaker/pengine_ctl
/var/spool/cron/crontabs/root	/usr/bin/memcached -m 64 -p 11211 -u memcache -l 127.0.1.1 -	/usr/bin/memcache
/var/spool/cron/crontabs/memcache	P /var/run/memcached/memcached.pid	

Сетевые индикаторы

CnC GoblinRAT – актуальность Сентябрь 2023.

qfilling.instanthq[.]com
 37.120.247[.]182
 188.120.249[.]79
 doh.box-host.cloudns[.]cl
 188.127.232[.]78
 chronyd.tftpd[.]net
 194.58.100[.]120
 62.109.0[.]95

Приложение 2. Расширенная информация по файловым IOCs

Путь	Хэш-сумма MD5	Хэш-сумма SHA1	Хэш-сумма
/usr/sbin/zabbix_agent	3f9b1b506dfab7a5cc32004a45ed780d	2167f0a1b7f8a3358c416d59124333ce70c2a137	b074749f16c
/var/lib/rhsm/rhsmc	cc5fdeaab47f63e54340adb1dc8f61d	438e49080d011d99e91ab0d644232b8e814c2bc8	08febd2a7ce
/var/lib/rhsm/rhsmc	5a517596ef9cff95ea755cdb93a51bf9	56b8e7f399578097a5b8e181e316ab350f52cc52	048e722e88
/usr/lib64/libiculxorg.so.50.1.2	0a762c0a6802545dc74c9852705b1cf3	124870723f6c408e18ad32a88212bc45b3767489	e274d528dc
/dev/shm/cron	fb7e2bf770f83d39df81789ab881abfa	202bdc8c8be32c7ca857cb01a854bc1c5f1528d1	9fe8eb5c813
/usr/bin/getty	ca36b1a673277c2db1995d736bada389	74fcaef8aedfa626ec3abc95bae348cfc87c02d4	b9ba5f3d563
/usr/bin/http	c1028c62e8538b29656872d48863f62b	b2a4c93d2f3c101a0e6b1b57d6fb95088a54f18f	8babfd28ee7
/usr/bin/psupply	04ea8d70b5a449d7db05f9e7be37bff6	d964b1c6a076858570b4fe893b9d7e0295eca0b1	890c4dea46
/usr/libexec/pacemaker/pengine_ctl	12ce9cd39153064ed7bcdd775cc2c469	9d5efc9a018e29c4f6960bfdc5e53d9daa91cd76	e3e68c6f963
/usr/bin/memcache	653a34f744917c3acd4969358f232685	7f3f85c062ca26cb4c20513babd98cc15eb2198a	aa475cd6c7

/usr/sbin/chronyd_debug	f6e689a238240b483d2652b53bc8b808	478a4fc65c89682f9847dcbf1721e00de7e2c6f2	fc989f128e
/dev/shm/s	cadaffae0d7e9a1a94735648b713a47e	af4b404a05dbe9e336179e1423d0817d36a7c043	690f5031de
test.py	3270a8c8e88518cdf8d0de94beb3a73f	627e6e30169c2bd7e7ca3262b81dae57ba7734c7	f80aa6aae4c

Приложение 3. Примеры конфигураций GoblinRAT

Конфиг для клиента:


```
"Proxy": "",  
"HTTPProxyUserAgent": ""  
}
```

Конфиг для сервера:

```

{
  "Base": {
    "Shell": "/bin/sh",
    "SystemInfoCmd": "A=$(lsb_release -d -s 2>&1)||A=$(cat /etc/*-release|grep PRETTY_NAME|cut -d \"'\" -f 2);echo $A",
    "KillSwitchDelay": 20,
    "AuthSecret": "d604e5e0834932566d346151c74c27aebc7a539fa88da439df5c428da60480c1fb57ed13ea1f62053dad0f4346f52fb9239c3540d0693429afafc31ead433ff5",
    "SocksTimeout": 30,
    "FirstCmds": [
      "w"
    ],
    "HostFingerprint": "",
    "AltProcessName": "/usr/bin/rhsmcertd"
  },
  "ServerKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEJwIBAAKCAgEA2G0JXYAmiowCnV2MzA2IpcUS7dKwq0aXiInqPeVcIXf8dCj6\ngnc56BgYfPzTteUdFbEcfcqi+J51CU/u4UE0x+YmXwZG2o/2839tv0JLBhnA16S
1\nR6ANq8SuU7rdqugCVV09xFGf3BT2vU8K/kYqf3PR5Wx/BNQYiqJba1du25wyKE\nnGhxoCjthPPHK0cwRr8EqavwvYV1I1fdak1M9kq6batsIIFx0zKKZt3brL8ir
HN\n8Lq68nu0767r0CIC33y/84yGZq8bgly5i/hRYqVPZ4dKD0Gxh9THGLv5vcoR98s40\nnVwYzEtz5mJZKAu3Pzmhn230L5FChw01XZ9HoV6E2nRbby57arI/C6u5kS5zB
QEAK6KXs61lj\nnphb9yL58r2m8VRZ1eeXwg7VOU0K0r/+3gE0H4pp1J5of/FBqZVIWjptq\nnQByhzIVPtSc9xW3hnQUwTE5Qx9ULbM9l5u9Ks15slgD2N1JeB+YDRXt550
4R/9\nnwJmNERZ+10JPG9UfPnpDYi6jQDDypQtzEuBi+Wei5o7JZZNF0mt8yz1x4tAsu0Mv\nnQCKSxMrG1q0BvBMM2qHAzZ0C/5G43oa0B6cvcEnoJjwbhDT7PNZtTrrKhuFu
ZJiakVn+VQAUrSHv9NayOv7ENhbkBBeuYiINPdkC5+9w4I10k7x9721fnH7E7shcJEC
AwEA\nnAQKCAQAJYBf5xBVnVRfsvEEYKVPQ1D3ienE87GyoRAMBguYypf/Mfb
y1fBdM\nnh/onO9QRYX3Bfsw2hu4LjDn16iKQyEdq6m0xXiEThSyzG/H8N+18gyz4KUSdrwy\n\nnjnRDZ4+KULlcwshxCttzEfSA9Vs9r1ahQCo00Rwvt79CnF5VtZHyjmhExA
h1eB+eJqC/nKSPLAukX0u/4AIUYh9Ywn66g1M02GKLHjapGUH2ldcWjTz83puIr9nrYu/
1f\nnida+c40DnjxmeYTs99lyP8AN7zDjy64tZu/vR0ui3bX0T40vFAZ2NiV89
AB4wcQAi\nn+wezvseMwnn08yLKUwVvWh24s4ivXqrr/MZi9fHRqyPqHsmaoTJIs0Aag418dJ\n\nnxQoeZCa0t5jXCgg5TG1lQIRN2lUJp/vdjfD4ntK040ERNnsgIkg41p
st+8BmIPN\nngwur0dFKAdeD2n0tCv58mBC5UyMRF95cmL8yjDMrMkhu5RpEP3JcXSBWYkzZt/Wg\n\nn4n+9cf7njHJ2PQqV5yBY8LaA3+66+rWox1AUDvVSioz8DigrPxanx
x3r8vxsZSg\nnSSAv+1tHfT20DEL3tzPwfA0M3EZjmgYVno9CYyWb3IXncYAtbK2TXHD+ryy/avYD\n\nn1zKr6RD+rnVrREBB6573ok1SEKfCUpap9PApJtpShoGjDmDWKQA
QEAK6KXs61lj\nnphb9yL58r2m8VRZ1eeXwg7VOU0K0r/+3gE0H4pp1J5of/FBqZVIWjptq\nnQByhzIVPtSc9xW3hnQUwTE5Qx9ULbM9l5u9Ks15slgD2N1JeB+YDRXt550
4R/9\nnwJmNERZ+10JPG9UfPnpDYi6jQDDypQtzEuBi+Wei5o7JZZNF0mt8yz1x4tAsu0Mv\nnQCKSxMrG1q0BvBMM2qHAzZ0C/5G43oa0B6cvcEnoJjwbhDT7PNZtTrrKhuFu
ZJiakVn+VQAUrSHv9NayOv7ENhbkBBeuYiINPdkC5+9w4I10k7x9721fnH7E7shcJEC
AwEA\nnAQKCAQAJYBf5xBVnVRfsvEEYKVPQ1D3ienE87GyoRAMBguYypf/Mfb
y1fBdM\nnh/onO9QRYX3Bfsw2hu4LjDn16iKQyEdq6m0xXiEThSyzG/H8N+18gyz4KUSdrwy\n\nnjnRDZ4+KULlcwshxCttzEfSA9Vs9r1ahQCo00Rwvt79CnF5VtZHyjmhExA
h1eB+eJqC/nKSPLAukX0u/4AIUYh9Ywn66g1M02GKLHjapGUH2ldcWjTz83puIr9nrYu/
1f\nnida+c40DnjxmeYTs99lyP8AN7zDjy64tZu/vR0ui3bX0T40vFAZ2NiV89
AB4wcQAi\nn+wezvseMwnn08yLKUwVvWh24s4ivXqrr/MZi9fHRqyPqHsmaoTJIs0Aag418dJ\n\nnxQoeZCa0t5jXCgg5TG1lQIRN2lUJp/vdjfD4ntK040ERNnsgIkg41p
st+8BmIPN\nngwur0dFKAdeD2n0tCv58mBC5UyMRF95cmL8yjDMrMkhu5RpEP3JcXSBWYkzZt/Wg\n\nn4n+9cf7njHJ2PQqV5yBY8LaA3+66+rWox1AUDvVSioz8DigrPxanx
x3r8vxsZSg\nnSSAv+1tHfT20DEL3tzPwfA0M3EZjmgYVno9CYyWb3IXncYAtbK2TXHD+ryy/avYD\n\nn1zKr6RD+rnVrREBB6573ok1SEKfCUpap9PApJtpShoGjDmDWKQA
QEAK6KXs61lj\nnphb9yL58r2m8VRZ1eeXwg7VOU0K0r/+3gE0H4pp1J5of/FBqZVIWjptq\nnQByhzIVPtSc9xW3hnQUwTE5Qx9ULbM9l5u9Ks15slgD2N1JeB+YDRXt550
4R/9\nnwJmNERZ+10JPG9UfPnpDYi6jQDDypQtzEuBi+Wei5o7JZZNF0mt8yz1x4tAsu0Mv\nnQCKSxMrG1q0BvBMM2qHAzZ0C/5G43oa0B6cvcEnoJjwbhDT7PNZtTrrKhuFu
ZJiakVn+VQAUrSHv9NayOv7ENhbkBBeuYiINPdkC5+9w4I10k7x9721fnH7E7shcJEC
AwEA\nnAQKCAQAJYBf5xBVnVRfsvEEYKVPQ1D3ienE87GyoRAMBguYypf/Mfb
y1fBdM\nnh/onO9QRYX3Bfsw2hu4LjDn16iKQyEdq6m0xXiEThSyzG/H8N+18gyz4KUSdrwy\n\nnjnRDZ4+KULlcwshxCttzEfSA9Vs9r1ahQCo00Rwvt79CnF5VtZHyjmhExA
h1eB+eJqC/nKSPLAukX0u/4AIUYh9Ywn66g1M02GKLHjapGUH2ldcWjTz83puIr9nrYu/
1f\nnida+c40DnjxmeYTs99lyP8AN7zDjy64tZu/vR0ui3bX0T40vFAZ2NiV89
AB4wcQAi\nn+wezvseMwnn08yLKUwVvWh24s4ivXqrr/MZi9fHRqyPqHsmaoTJIs0Aag418dJ\n\nnxQoeZCa0t5jXCgg5TG1lQIRN2lUJp/vdjfD4ntK040ERNnsgIkg41p
st+8BmIPN\nngwur0dFKAdeD2n0tCv58mBC5UyMRF95cmL8yjDMrMkhu5RpEP3JcXSBWYkzZt/Wg\n\nn4n+9cf7njHJ2PQqV5yBY8LaA3+66+rWox1AUDvVSioz8DigrPxanx
x3r8vxsZSg\nnSSAv+1tHfT20DEL3tzPwfA0M3EZjmgYVno9CYyWb3IXncYAtbK2TXHD+ryy/avYD\n\nn1zKr6RD+rnVrREBB6573ok1SEKfCUpap9PApJtpShoGjDmDWKQA
QEAK6KXs61lj\nnphb9yL58r2m8VRZ1eeXwg7VOU0K0r/+3gE0H4pp1J5of/FBqZVIWjptq\nnQByhzIVPtSc9xW3hnQUwTE5Qx9ULbM9l5u9Ks15slgD2N1JeB+YDRXt550
4R/9\nnwJmNERZ+10JPG9UfPnpDYi6jQDDypQtzEuBi+Wei5o7JZZNF0mt8yz1x4tAsu0Mv\nnQCKSxMrG1q0BvBMM2qHAzZ0C/5G43oa0B6cvcEnoJjwbhDT7PNZtTrrKhuFu
ZJiakVn+VQAUrSHv9NayOv7ENhbkBBeuYiINPdkC5+9w4I10k7x9721fnH7E7shcJEC
AwEA\nnAQKCAQAJYBf5xBVnVRfsvEEYKVPQ1D3ienE87GyoRAMBguYypf/Mfb
y1fBdM\nnh/onO9QRYX3Bfsw2hu4LjDn16iKQyEdq6m0xXiEThSyzG/H8N+18gyz4KUSdrwy\n\nnjnRDZ4+KULlcwshxCttzEfSA9Vs9r1ahQCo00Rwvt79CnF5VtZHyjmhExA
h1eB+eJqC/nKSPLAukX0u/4AIUYh9Ywn66g1M02GKLHjapGUH2ldcWjTz83puIr9nrYu/
1f\nnida+c40DnjxmeYTs99lyP8AN7zDjy64tZu/vR0ui3bX0T40vFAZ2NiV89
AB4wcQAi\nn+wezvseMwnn08yLKUwVvWh24s4ivXqrr/MZi9fHRqyPqHsmaoTJIs0Aag418dJ\n\nnxQoeZCa0t5jXCgg5TG1lQIRN2lUJp/vdjfD4ntK040ERNnsgIkg41p
st+8BmIPN\nngwur0dFKAdeD2n0tCv58mBC5UyMRF95cmL8yjDMrMkhu5RpEP3JcXSBWYkzZt/Wg\n\nn4n+9cf7njHJ2PQqV5yBY8LaA3+66+rWox1AUDvVSioz8DigrPxanx
x3r8vxsZSg\nnSSAv+1tHfT20DEL3tzPwfA0M3EZjmgYVno9CYyWb3IXncYAtbK2TXHD+ryy/avYD\n\nn1zKr6RD+rnVrREBB6573ok1SEKfCUpap9PApJtpShoGjDmDWKQA
QEAK6KXs61lj\nnphb9yL58r2m8VRZ1eeXwg7VOU0K0r/+3gE0H4pp1J5of/FBqZVIWjptq\nnQByhzIVPtSc9xW3hnQUwTE5Qx9ULbM9l5u9Ks15slgD2N1JeB+YDRXt550
4R/9\nnwJmNERZ+10JPG9UfPnpDYi6jQDDypQtzEuBi+Wei5o7JZZNF0mt8yz1x4tAsu0Mv\nnQCKSxMrG1q0BvBMM2qHAzZ0C/5G43oa0B6cvcEnoJjwbhDT7PNZtTrrKhuFu
ZJiakVn+VQAUrSHv9NayOv7ENhbkBBeuYiINPdkC5+9w4I10k7x9721fnH7E7shcJEC
AwEA\nn
```

```
        62517
    ],
    "AuthDelay": 20,
    "InactivityTimeout": 1
}
```

Приложение 4. Yara-правило GoblinRAT

```

rule nix_mem_GoblinRAT_go {

    meta:
        description = "Detects GoblinRAT in file and memory"
        author = "4RAYS"

    strings:
        //Memory strings from commands
        $m_comm1 = "SOCKS CONN"
        $m_comm2 = "DISCONNECT CONN"
        $m_comm3 = "FORWARD CONN"
        $m_comm4 = "CONFIG CONN"
        $m_comm5 = "SHELL CONN"
        $m_comm6 = "HANDSHAKE CONN"

        //Memory strings from logger
        $m_log1 = "Conn from %s allowed: %t"
        $m_log2 = "Client %v connecting..."
        $m_log3 = "Handshake failed"
        $m_log4 = "Handshake succeeded"
        $m_log5 = "Softkill..."
        $m_log6 = "Rearming kill switch"
        $m_log7 = "Machine ID: %s"
        $m_log8 = "machineid: %v"
        $m_log9 = "HostFingerprint"
        $m_log10 = "Applying config: %#v"
        $m_log11 = "Server: ver: %2x"
        $m_log12 = "Client: ent: %2x"
        $m_log13 = "Listener: %s"
        $m_log14 = "Knocking from %s"
        $m_log15 = "Shreding file: %s"
        $m_log16 = "Renaming process from %s to %s"

        //Other memory strings
        $m_tool1 = "/var/lib/dbus/machine-id"
        $m_tool2 = "/etc/machine-id"
        $m_tool3 = ".bak"
        $m_tool4 = ".lib"
        $m_tool5 = "HISTFILE=/dev/null"
        $m_tool6 = "LANG=en_US.UTF-8"
        $m_tool7 = "XINERAMA"
        $m_tool8 = "PATH=%s"
        $m_tool9 = "[ERR] soocks: %v"
        $m_tool10 = "ip4:tcp"
        $m_tool11 = "/dev/ptmx"
        $m_tool12 = "/dev/pts/"

        //Packages
        $s_pack1 = "/lib64/ld-linux-x86-64.so.2" ascii wide nocase
        $s_pack2 = "golang" ascii
        $s_pack3 = "crypto/aes/cipher.go"
        $s_pack4 = "encoding/json"
        $s_pack5 = "crypto/tls.X509KeyPair"

        //Main session struct functions
        $s_session_funcs1 = ".HandleAuthSuccess"
        $s_session_funcs2 = ".HandleAuthFailure"
        $s_session_funcs3 = ".HandleClose"
        $s_session_funcs4 = ".HandleKill"
        $s_session_funcs5 = ".IsAllowed"
        $s_session_funcs6 = ".RefreshConfig"

        //Packages functions
        $s_packages_funcs1 = ".XORKeyStream"
        $s_packages_funcs2 = ".NewConnDisplay"
        $s_packages_funcs3 = ".getInputFocusRequest"
        $s_packages_funcs4 = ".AtomErrorNew"
        $s_packages_funcs5 = ".IsTerminal"
        $s_packages_funcs6 = ".Ping"
        $s_packages_funcs7 = ".Authenticate"

        //Gobfuscate algs
        $gob1 = {C6 44 [3] C6 44 [3] C6 44 [3] 31 C0 EB ?? 0F B6 [3] 0F B6 [3] 31 D1 [4] 48 FF C0 48 83 F8 01 7C} // gobfuscator
string decode t1
        $gob2 = {0F 57 C0 0F 11 44 [2] 0F 11 44 [2] 31 C0 EB ?? 0F B6 [3] 0F B6 [3] 31 D1 [4] 48 FF C0 [0-3] 48 83 F8 ?? 7C} //
gobfuscator string decode t2
        $gob3 = {31 C0 EB ?? 0F B6 [3] 0F B6 [3] 31 D1 [4] 48 FF C0 [0-8] 48 83 F8 ?? 7C} // gobfuscator xor loop

```

```

condition:
//File
(
    (uint32(0) == 0x464c457f) and filesize > 3MB
    and
    (
        4 of ($s_pack*)
    )
    and
    (
        3 of ($s_session_funcs*)
    )
    and
    (
        ($gob1 and #gob2 > 10) or #gob3 > 20
    )
)
or
// Memory
(
    (any of ($m_comm*) or 5 of ($m_log*))
    and
    (
        6 of ($m_tool*) or
        (($gob1 and #gob2 > 10) or #gob3 > 20)
    )
)
}

```