АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Carbon: анализ бэкдора второго этапа из арсенала кибергруппы Turla

7 апреля 2017 года

Кибергруппа Turla на протяжении многих лет проводит масштабные операции шпионажа. Недавно мы нашли несколько новых модификаций Carbon – бэкдора второго этапа из арсенала хакеров. В прошлом году швейцарский GovCERT.ch выполнил технический анализ этого компонента в отчете об атаке на швейцарскую оборонную компанию RUAG. Данный пост посвящен техническим новшествам, которые мы обнаружили в последних версиях Carbon.

Различия между модификациями Carbon указывают на то, что авторы активно развивают программу. Известно, что кибергруппа вносит изменения в свои вредоносные инструменты после их обнаружения. Это справедливо и для Carbon – две основные версии различаются мьютексами и названиями файлов.

Векторы заражения

Turla известна тщательной поэтапной работой в скомпрометированных системах. Первоначально хакеры проводят разведку и только после этого развертывают наиболее сложные инструменты, такие как Carbon.

Классическая схема заражения Carbon выглядит следующим образом: пользователь получает фишинговое письмо или заходит на скомпрометированный сайт (как правило, ресурс, который посещает регулярно) – метод watering hole. После успешной атаки на компьютер загружается бэкдор первого этапа, например, Tavdig или Skipper. Когда фаза разведки завершена, в важнейших системах устанавливается бэкдор второго этапа – такой как Carbon.

Технический анализ

Carbon – сложный бэкдор, предназначенный для кражи конфиденциальной информации у объектов, представляющих интерес для хакеров Turla.

Вредоносная программа имеет общие черты с руткитом Uroburos, который использует эта группа. Наиболее значимое сходство – структура коммуникаций. Обе программы обеспечивают каналы коммуникации между различными компонентами, объекты связи, структуры и координирующие таблицы выглядят идентично. Единственное различие – в Carbon предусмотрено меньше коммуникационных каналов. Мы предполагаем, что Carbon – «облегченная» версия Uroburos (без компонентов ядра и эксплойтов).

Установке Carbon предшествует доставка компонента для разведки (например, Tavdig или Skipper). Он собирает информацию о машине жертвы и сети. Если цель показалась интересной, ей будут направлены более сложные вредоносные программы – такие как Carbon или Uroburos.

Глобальная архитектура

Инфраструктура Carbon состоит из следующих компонентов:

- дроппер, который устанавливает компоненты Carbon и его конфигурационный файл
- компонент, который поддерживает связь с C&C сервером
- оркестратор, который обрабатывает задачи, направляет их на другие компьютеры сети и внедряет в легитимный процесс DLL, взаимодействующий с C&C
- загрузчик, который исполняет оркестратор

Хронология разработки

Оркестратор и внедренная библиотека имеют собственную ветвь разработки.

Благодаря датам компиляции и внутренним номерам версий, жестко закодированным в PE-файлах, мы можем получить следующую временную шкалу:

Compilation date	Orchestrator version	Injected library version
2014-02-26	3.71	3.62
2016-02-02	3.77	4.00
2016-03-17	3.79	4.01
2016-03-24	3.79	4.01
2016-04-01	3.79	4.03
2016-08-30	3.81	????
2016-10-05	3.81	????
2016-10-21	3.81	????

Файлы Carbon

Названия файлов инфраструктуры Carbon варьируются в зависимости от версии. При этом все они сохраняют одно и то же внутреннее название (в метаданных):

- дроппер: SERVICE.EXE
- загрузчик: SERVICE.DLL или KmSvc.DLL
- оркестратор: MSIMGHLP.DLL
- внедренная библиотека: MSXIML.DLL

У каждого из файлов есть 32- и 64-битная версии.

Рабочий каталог

Carbon создает несколько файлов для ведения журналов, задач на исполнение и настроек, которые будут модифицировать поведение вредоносной программы. Содержимое большинства этих файлов зашифровано алгоритмом CAST-128.

Основной рабочий каталог будет содержать файлы и папки, относящиеся к Carbon. Каталог выбирается случайным образом из папок в %ProgramFiles%, за исключением WindowsApps.

Названия файлов жестко закодированы в оркестраторе. Эти же названия используются в ветви 3.7x +. Поскольку внедренная DLL обращается к тем же файлам, что и оркестратор, это еще один простой способ связать версию библиотеки и оркестратор.

Файловая структура Carbon 3.7x:

```
\%carbon_working_folder\%   // базовая папка
├── 0208 // результаты выполнения задач и файлы журналов
│   ├── C_56743.NLS // содержит список файлов для отправки на C&C сервер,
этот файл не упаковывается и не шифруется
├── asmcerts.rs
├── getcerts.rs
├── miniport.dat  // конфигурационный файл
├── msximl.dll    // внедренная библиотека (x32)
├── Nls // содержит задачи (команды на исполнение или PE-файл) и их конфигурационные
файлы
│   ├── a67ncodc.ax  // задачи для исполнения оркестратором
│   ├── b9s3coff.ax  // задачи для исполнения внедренной библиотекой
├── System   // папка дополнительных модулей
│   ├── bootmisc.sdi // не используется
├── qavscr.dat    // журнал ошибок
├── vndkrmn.dic   // системный журнал
└── ximarsh.dll   // внедренная библиотека (x64)
```

Начиная с версии 3.80 все названия файлов изменены.

Файловая структура Carbon 3.8x:

```
\carbon_working_folder\%   // базовая папка
├── 0409  // содержит задачи (команды на исполнение или PE-файлы) и их конфигурационные
файлы
│   ├── cifrado.xml    // задачи для исполнения внедренной библиотекой
│   ├── encodebase.inf // задачи для исполнения оркестратором
├── 1033 // результаты выполнения задач и файлы журналов
│   ├── dsntype.gif // содержит список файлов для отправки на C&C сервер,
этот файл не упаковывается и не шифруется
├── en-US  // папка дополнительных модулей
│   ├── asmlang.jpg // не используется
├── fsbootfail.dat  // журнал ошибок
├── mkfieldsec.dll  // внедренная библиотека (x32)
├── preinsta.jpg    // системный журнал
├── wkstrend.xml    // конфигурационный файл
├── xmlrts.png
└── zcerterror.png
```

Доступ к файлам

Большинство файлов из рабочей папки Carbon при обращении вредоносной программы выполняют следующие шаги:

- для обеспечения эксклюзивного доступа используется специальный мьютекс
- файл дешифруется (CAST-128)
- когда операции с файлом завершены, он шифруется повторно (CAST-128)
- мьютекс освобожден

Мьютексы

Следующие мьютексы создаются оркестратором в версии Carbon 3.7x:

- “Global\\MSCTF.Shared.MUTEX.ZRX” (обеспечивает эксклюзивный доступ к “vndkrmn.dic”)
- “Global\\DBWindowsBase” (обеспечивает эксклюзивный доступ к “C_56743.NLS”)
- “Global\\IEFrame.LockDefaultBrowser” (обеспечивает эксклюзивный доступ к “b9s3coss.ax”)
- “Global\\WinSta0_DesktopSessionMut” (обеспечивает эксклюзивный доступ к “a67ncodc.ax”)
- “Global\{5FA3BC02-920F-D42A-68BC-04F2A75BE158}” (обеспечивает эксклюзивный доступ к новым файлам, созданным в папке “Nls”)
- “Global\\SENS.LockStarterCacheResource” (обеспечивает эксклюзивный доступ к “miniport.dat”)
- “Global\\ShimSharedMemoryLock” (обеспечивает эксклюзивный доступ к “asmcerts.rs”)

В Carbon 3.8x названия файлов и мьютексов изменены:

- “Global\\Stack.Trace.Multi.TOS” (обеспечивает эксклюзивный доступ к “preinsta.jpg”)
- “Global\\TrackFirleSystemIntegrity” (обеспечивает эксклюзивный доступ к “dsntype.gif”)
- “Global\\BitswapNormalOps” (обеспечивает эксклюзивный доступ к “cifrado.xml”)
- “Global\\VB_crypto_library_backend” (обеспечивает эксклюзивный доступ к “encodebase.inf”)
- “Global\{E41B9AF4-B4E1-063B-7352-4AB6E8F355C7}” (обеспечивает эксклюзивный доступ к новым файлам, созданным в папке “0409”)
- “Global\\Exchange.Properties.B” (обеспечивает эксклюзивный доступ к “wkstrend.xml”)
- “Global\\DatabaseTransSecurityLock” (обеспечивает эксклюзивный доступ к “xmlrts.png”)

Эти мьютексы также используются во внедренной DLL, чтобы убедиться, что оркестратор исполнен.

Конфигурационный файл

Конфигурационный файл влияет на поведение вредоносной программы. Его формат напоминает форматы “inf”, используемые в Windows. Помимо всего прочего, он содержит:

- “object_id” – уникальный uuid, использующийся для идентификации жертвы, когда его значение не указано в файле, генерируется случайным образом
- список процессов, в которые вводится код (iproc)
- частота и время исполнения задач / журналов резервного копирования / соединения с C&C ([TIME])
- IP-адреса других компьютеров сети ([CW_LOCAL])
- адрес C&C сервера ([CW_INET])
- именованные каналы, используемые для связи с внедренной библиотекой и другими компьютерами ([TRANSPORT])

Позже этот файл может быть обновлен. Действительно, в коммуникационной библиотеке некоторые криптографические ключи используются для шифрования / дешифрования данных, и они извлекаются из секции [CRYPTO] конфигурационного файла, которого не существует, когда файл сбрасывается из ресурсов загрузчика.

Конфигурационный файл Carbon 3.77:

```
[NAME]
object_id=
iproc = iexplore.exe,outlook.exe,msimn.exe,firefox.exe,opera.exe,chrome.exe
ex = #,netscape.exe,mozilla.exe,adobeupdater.exe,chrome.exe

[TIME]
user_winmin = 1800000
user_winmax = 3600000
sys_winmin = 3600000
sys_winmax = 3700000
task_min = 20000
task_max = 30000
checkmin = 60000
checkmax = 70000
logmin =  60000
logmax = 120000
lastconnect=111
timestop=
active_con = 900000
time2task=3600000

[CW_LOCAL]
quantity = 0

[CW_INET]
quantity = 3
address1 = doctorshand.org:80:/wp-content/about/
address2 = www.lasac.eu:80:/credit_payment/url/
address3 = www.shoppingexpert.it:80:/wp-content/gallery/

[TRANSPORT]
system_pipe = comnap
spstatus = yes
adaptable = no

[DHCP]
server = 135

[LOG]
logperiod = 7200

[WORKDATA]
run_task=
run_task_system=
```

Файл журнала

Инфраструктура Carbon включает файл журнала, который используется для записи действий, выполняемых вредоносной программой, и информации о системе, которая может быть полезна для атакующих (например, если на машине запущен инструмент для анализа, такой как WireShark).

Формат журнала не изменился с версии Carbon 3.71:

- Date|Time|Object-Id|Source|Message

Например:

```
[LOG]
start=1
20/02/17|12:48:24|8hTdJtUBB57ieReZAOSgUYacts|s|OPER|New object ID generated '8hTdJtUBB57
ieReZAOSgUYacts'|
20/02/17|12:48:24|8hTdJtUBB57ieReZAOSgUYacts|s|ST|3/81|0|
20/02/17|12:48:24|8hTdJtUBB57ieReZAOSgUYacts|s|START OK
```

Резервные копии файла периодически отправляются на управляющий сервер.

Дроппер

Дроппер – единственный исполняемый файл, который не является DLL. Это первый PE-файл, который должен быть исполнен, он используется для извлечения других компонентов.

PE-файлы для загрузки основных компонентов извлекаются в системный каталог Windows, в то время как оркестратор, библиотека для связи с C&C и конфигурационный файл – в рабочий каталог Carbon.

Новый раздел добавляется в случайный файл “.inf” из %SystemRoot%\INF. Название раздела – номер серийного диска скомпрометированного компьютера, значение «root» создается с выбранным рабочим каталогом Carbon.

Например:

```
[5049654F]
root="C:\Program Files\Windows Portable Devices"
```

Оркестратор

Оркестратор – основной компонент инфраструктуры Carbon. Он используется преимущественно для внедрения кода в процесс, который осуществляет легитимные коммуникации через интернет, и отправки задач, полученных из внедренной библиотеки, на другие компьютеры в той же сети через именованные каналы или TCP.

Вредоносная программа создает семь потоков. Несложно определить характеристики Carbon, поскольку каждый поток играет определенную роль.

Загрузка настроек

Конфигурационный файл может быть обновлен вредоносной программой, поэтому некоторые атрибуты, в частности, адреса управляющих серверов, отслеживаются каждые 10 минут.

Регулярная проверка папки хранения Carbon

В рабочем каталоге Carbon есть папка для хранения. Она содержит некоторые файлы, загруженные с C&C сервера (задачи, которые являются командами к исполнению или PE-файлами, и их конфигурационные файлы).

Этот поток будет работать постоянно и каждые два часа проверять, достаточно ли места в папке;

если нет, в файл журнала записывается уведомление.

Выполнение задач

Выполнение задач в контексте процесса оркестратора похоже на способ, которым они выполняются в коммуникационной библиотеке (см. Коммуникационная библиотека / Выполнение задач).

В отличие от коммуникационной библиотеки, список задач к исполнению содержит файл encodebase.inf (Carbon v3.8x) или a67ncode.ax.

Каждая строка файла состоит из следующих элементов:

- task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath | [execution_mode | username | password]

Необходимы первые пять полей, последние три – опциональны. Если поле execution_mode существует, его значение влияет на способ исполнения задачи:

- 0 или 1: нормальное исполнение
- 2: задача исполняется в контексте безопасности данного пользователя (учетные данные предоставляются через поля для ввода логина и пароля)
- 3 или 4: задача исполняется в контексте безопасности пользователя, представленного токеном explorer.exe

P2P

Как и Uroburos/Snake, Carbon может отправлять задачи на другие компьютеры сети через именованный канал или TCP. Это можно использовать для передачи и исполнения задач на компьютерах, не подключенных к интернету.

Коммуникационные каналы

Uroburos использует несколько типов коммуникационных протоколов, которые можно классифицировать следующим образом:

- тип 1: TCP
- тип 2: enc, np, reliable, frag, m2b, m2d
- тип 3: t2m
- тип 4: UDP, doms, domc

```
.data:00079018 off_79018       dd offset aTcp          ; DATA XREF: .data:off_790BC↓o
.data:00079018                                         ; "tcp"
.data:0007901C                 dd 1
.data:00079020                 dd offset handler_tcp
.data:00079024                 dd offset aEnc          ; "enc"
.data:00079028                 dd 2
.data:0007902C                 dd offset handler_enc
.data:00079030                 dd offset aNp           ; "np"
.data:00079034                 dd 2
.data:00079038                 dd offset handler_np
.data:0007903C                 dd offset aReliable     ; "reliable"
.data:00079040                 dd 2
.data:00079044                 dd offset handler_reliable
.data:00079048                 dd offset aFrag         ; "frag"
.data:0007904C                 dd 2
.data:00079050                 dd offset handler_frag
.data:00079054                 dd offset aUdp          ; "udp"
.data:00079058                 dd 4
.data:0007905C                 dd offset handler_udp
.data:00079060                 dd offset aM2b          ; "m2b"
.data:00079064                 dd 2
.data:00079068                 dd offset handler_m2b
.data:0007906C                 dd offset aT2m          ; "t2m"
.data:00079070                 dd 3
.data:00079074                 dd offset handler_t2m
.data:00079078                 dd offset aM2d          ; "m2d"
.data:0007907C                 dd 2
.data:00079080                 dd offset handler_m2d
.data:00079084                 dd offset aDoms         ; "doms"
.data:00079088                 dd 4
.data:0007908C                 dd offset handler_doms
.data:00079090                 dd offset aDomc         ; "domc"
.data:00079094                 dd 4
.data:00079098                 dd offset handler_domc
```

У Carbon каналов связи меньше:

- тип 1: TCP, b2m
- тип 2: np, frag, m2b

```
off_20021100    dd offset aTcp          ; DATA XREF: .data:off_20021154↓o
                                        ; "tcp"
                dd 1
                dd offset handler_tcp
                dd offset aNp           ; "np"
                dd 2
                dd offset handler_np
                dd offset aFrag         ; "frag"
                dd 2
                dd offset handler_frag
                dd offset aM2b          ; "m2b"
                dd 2
                dd offset handler_m2b
                dd offset aB2m          ; "b2m"
                dd 1
                dd offset handler_b2m
```

Данные, отправляемые на одноранговые узлы, обычно фрагментируются и переносятся по TCP или через именованный канал. Например, если фрагментированные данные отправляются с одного компьютера на другой по именованному каналу, создается объект frag.np. В этом случае

будет вызван конструктор класса frag, за которым следует вызов конструктора подкласса np.

Существует структура, состоящая из нескольких обработчиков для каждого объекта: инициализация связи, подключение (к каналу / IP-адресу), чтение данных, отправка данных и др.

Отправка задач на другой компьютер

Чтобы отправить данные с одного компьютера на другой, необходимо выполнить несколько шагов:

- создается коммуникационный канал (объект frag.np или frag.tcp) с определенным именованным каналом или IP-адресом
- параметры передаются объекту связи (например, размер фрагмента, информация об узле связи и др.)
- выполняется подключение к одноранговому узлу связи
- выполняется аутентификация хостинга и узла связи:
- *подтверждение подключения происходит, когда хостинг отправляет «магическое» значение A110EAD1EAF5FA11 и ожидает получить C001DA42DEAD2DA4 от узла*
- *команда WHO отправляется на узел связи, где хост посылает uuid жертвы и ожидает получить тот же uuid*
- если аутентификация успешна, данные отправляются на узел связи

Все коммуникации между хостингом и узлом зашифрованы алгоритмом CAST-128.
Обратите внимание, что функция P2P также реализована в коммуникации DLL.

Дополнительные модули

Вредоносная программа поддерживает дополнительные модули, чтобы расширить функциональность.

В конфигурационном файле есть раздел под названием PLUGINS. Он может отсутствовать, если конфигурационный файл удален из ресурсов загрузчика, но этот файл может быть обновлен вредоносной программой. Раздел PLUGINS содержит строку, образованную следующим образом:

- %plugin_name%=%enabled%|%mode%[:%username%:%password%]|%file_path%

%file_path% — путь к PE-файлу или файл, содержащий командную строку для исполнения.
%enabled% — строка, которая используется, чтобы знать, должен ли быть исполнен плагин. Если это так, значение строки — enabled.

Атрибут %mode% используется для управления контекстом, в котором исполняется PE-файл или командная строка. Это может быть:

- 1 = исполнение с текущими привилегиями пользователя в текущем контексте процесса через CreateProcess().
- 2 = исполнение от пользователя, указанного в конфигурации (атрибуты :%username%:%password%), токен конкретного пользователя извлекается при помощи функции LogonUserAs().
- 3 = исполнение в контексте безопасности пользователя, представленного токеном explorer.exe (токен процесса explorer.exe дублируется и передается через функцию CreateProcessAsUser().

- 4 = как в п.3, но переменные окружения для пользователя, представленного токеном explorer.exe, извлекаются и передаются функции CreateProcessAsUser()

Если это PE-файл:

- файл загружается в память процесса вредоносной программы
- модуль анализируется, чтобы проверить, является ли он библиотекой DLL
- если модуль является DLL и экспортирует функцию ModStart (начиная с Carbon 3.77) или ModuleStart (для более ранних версий Carbon), для выполнения этой функции создается новый поток
- если модуль – не DLL, а валидный PE, он выполняется из точки входа

Внедрение коммуникационной библиотеки в удаленные процессы

Библиотека, которая используется для связи с C&C сервером, внедряется в удаленные процессы. Чтобы знать, где внедрить эту DLL, анализируется конфигурационный файл. Раздел [NAME] содержит поле iproc со списком процессов, которые могут легитимно обмениваться данными с интернетом.

Например:

```
[NAME]
iproc = iexplore.exe,outlook.exe,msimn.exe,firefox.exe,opera.exe,chrome.exe
```

Библиотека DLL будет добавлена в каждый процесс из списка, который исполняется в системе, если название родительского процесса explorer.exe или ieuser.exe.

Классическая схема инъекции:

- функции CreateToolHelp32Snapshot / Module32FirstW / Module32NextW используются для извлечения базового адреса модуля kernel32.dll
- модуль EAT анализируется, чтобы получить адрес функции LoadLibraryW
- привилегия SeDebugPrivilege включена для текущего процесса
- память выделяется в удаленный процесс, в нее записывается путь к библиотеке
- NtCreateThreadEx или CreateRemoteThread (если адрес первой функции не может быть получен) вызывается для исполнения LoadLibraryW для загрузки библиотеки DLL в память удаленного процесса

Коммуникационная библиотека

Следующий анализ основан на msximl версии 4.x. Этот компонент может быть изменен в последних версиях.

Загрузка настроек

Помимо кода, в потоке «Выбор конфигурации» от оркестратора (подобного), поле sethttp1 извлекается из секции [TRANSPORT].

Если это значение установлено, HTTP 1.1 будет использоваться для будущих подключений.

Выполнение задач

Задачи извлекаются с управляющего C&C сервера.

Задачи, которые необходимо выполнить коммуникационной библиотеке, перечислены в файле b9s3coff.ax (Carbon v3.7x) или cifrado.xml (Carbon v3.8x).

Каждая строка файла состоит из следующих элементов:

- task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath

Файл задачи и его конфигурация шифруются (CAST-128), задача выполняется. Есть несколько параметров, которые извлекаются из конфигурационного файла Carbon: time2task используется для установки времени ожидания выполнения задачи (по умолчанию один час), task_min и task_max – для ожидания случайного времени между выполнением задач из списка (случайное время будет установлено между task_min и task_max). Эти параметры взяты из раздела [TIME].

Если задача является корректным DLL-файлом, она загружается в текущую память процесса, и создается новый поток для выполнения «стартового» экспорта. В противном случае это, вероятно, команда на исполнение. В этом случае анализируется конфигурационный файл. Как конфигурационный файл Carbon, файл конфигурации задачи формируется как inf и должен содержать раздел [CONFIG] со следующими параметрами:

[CONFIG]

- NAME (по умолчанию cmd.exe)
- ARG
- RESULT (по умолчанию stdout)
- COMPRESSION (по умолчанию yes)
- DELETE (по умолчанию no)

Команда с ее аргументами исполняется через функцию CreateProcess(), результаты сохраняются в файле %task_result_filepath%, если для параметра RESULT не задано значение stdout – в этом случае ошибка или дополнительная информация добавляются в файл журнала задачи.

Если для параметра RESULT установлено значение stdout:

- удаляется файл результата задачи, результат выполнения задачи записывается в файл задач журнала
- строка task_id | “1” | task_log_filepath | object_id добавляется в конец файла C_56743.NLS (Carbon 3.7x) или dsntype.gif (Carbon 3.8x)

В противном случае:

- выполнение задачи выводится в файл результата задачи и ошибки / дополнительная информация – в файл журнала задач
- строка task_id | “2” | task_log_filepath | task_result_filepath | object_id добавляется в конец файла C_56743.NLS (Carbon 3.7x) или dsntype.gif (Carbon 3.8x)

В обоих случаях в поле run_task из конфигурационного файла Carbon записывается та же строка, что и в C_56743.NLS. Она может быть предназначена для того, чтобы авторы вредоносной программы знали, какая задача выполнена последней, когда конфигурационный файл будет

отправлен на C&C сервер (см. «Резервное копирование конфигурационного файла»).

Ротация журнала

Резервные копии файла журнала периодически отправляются на управляющий сервер (по умолчанию это происходит каждые два часа).

Подобно резервному копированию конфигурационного файла, это действие выполняется только в определенное время суток. Действительно, ротация журнала будет выполняться только тогда, когда местное время находится в диапазоне, заданном конфигурационным файлом.

Поля wh_min и wh_max извлекаются из секции [TIME] конфигурационного файла (по умолчанию 8 и 20). Этот поток будет исполнять код только в том случае, если текущий час находится между %wh_min% и %wh_max%.

Если значение атрибута из секции [TIME], который выглядит как wDay: wMonth: wYear: wHour: wMinute, timestop, поток выполнит код только после этой конкретной даты и времени.

Атрибуты lastsend и logperiod из секции [LOG] используются для указания времени задержки при резервном копировании и отправке журнала на C&C сервер (по умолчанию каждые два часа).

Временный файл со случайным именем с префиксом ~D создается в папке 208 (Carbon v3.7x) или 1033 (Carbon v3.8x). Содержимое файла журнала копируется в этот новый файл, затем он сжимается при помощи Bzip2 и шифруется (CAST-128).

Новая строка добавляется в конец файла C_56743.NLS (Carbon v3.7x) или dsntype.gif (Carbon v3.8x):

“10|1|%s|%s”

- 1-е поле: ID для идентификации файла в виде файла журнала
- 2-е поле: 1 (файл для отправки на C&C сервер)
- 3-е поле: путь к временному файлу
- 4-е поле: uuid жертвы

Наконец, атрибут lastsend обновляется в текущем времени, и исходный файл журнала удаляется.

Коммуникации с C&C сервером

Код этого потока используется для извлечения новых задач с управляющего сервера, отправки новых файлов на сервер (перечисленных в файле C_56743.NLS / dsntype.gif) и отправки новых задач в оркестратор.

Первый запрос

Адрес случайного C&C сервера выбирается из указанных в разделе CW_INET. Если порт и путь к ресурсу HTTP не указаны, по умолчанию используется порт 80 и /javascript/view.php.

Пользовательский агент настраивается следующим образом:

Версия Internet Explorer извлекается через раздел реестра: HKLM\Software\Microsoft\Internet Explorer\Version и присоединяется к строке Mozilla/4.0 (compatible; MSIE %d.0;

- *например, Mozilla/4.0 (compatible; MSIE 8.0.6001.18702.0;*

Объединить предыдущую строку со значением основной / младшей версии ОС (через GetVersionExA())

- *Mozilla/4.0 (compatible; MSIE 8.0.6001.18702.0; Windows NT 5.1; Trident/4.0*

Перечислить значения ключа в HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\User Agent\Post Platform и объединить каждое значение с предыдущей строкой, а затем добавить заключительное выражение.

- *например, Mozilla/4.0 (compatible; MSIE 8.0.6001.18702.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; .NET4.0C; .NET4.0E; Media Center PC 6.0; SLCC2)*

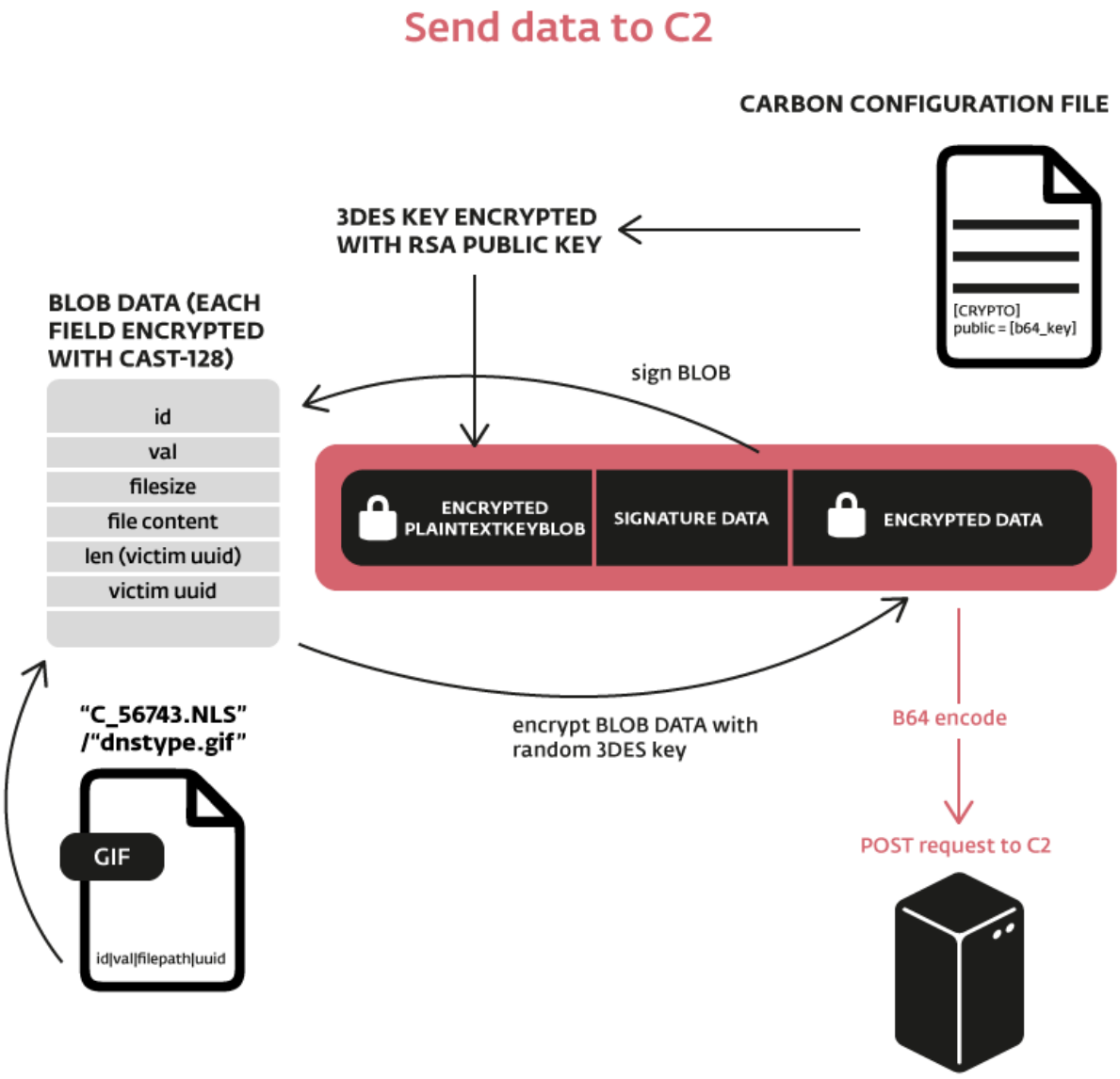
Поле trans_timemax из секции [TIME] выбрано. Оно используется для установки таймаута для интернет-запросов (через InternetSetOption()). По умолчанию, его значение – 10 минут.

Первый запрос GET выполняется на корневой странице C&C веб-сервера, чтобы проверить подключение. Если в системе не выполняется захват пакетов, на управляющий сервер направляется новый запрос, чтобы проверить, доступны ли новые задачи. К запросу добавляется файл cookie PHPSESSID с uuid жертвы в качестве значения. Также добавляется заголовок Referer и устанавливается URL-адрес C&C сервера.

Вредоносная программа ожидает ответ на запрос GET, аналогичный следующему:
input name=”%name%” value=”%data_in_b64%”

Если в поле value есть какое-либо значение, доступна новая задача.

Отправка данных на сервер



Если файл C_56743.NLS / dsntype.gif не пустой, это значит, что он содержит данные для отправки на C&C сервер. Файл анализируется, извлекается последняя строка, которая содержит детали о данных для отправки. Двоичный объект данных построен, каждое из следующих полей зашифровано алгоритмом CAST-128:

id | val | tmp_filesize | tmp_content | [OPTIONAL (if val == 2) tmp2_filesize | tmp2_content] | len_object_id | object_id

id = тип данных для отправки на C&C сервер, это может быть:

- 10: резервная копия журналов
- 11: конфигурационный файл
- 20: криптографический ключ
- В противном случае, связанный с задачей ID может быть результатом задачи или журналом ошибок в случае сбоя выполнения задачи

val = 1 если только один файл для отправки, 2 если два файла
object_id = uuid жертвы

Если в поле dtc из секции [CRYPTO] конфигурационного файла установлено 0, этот BLOB кодируется base64 и отправляется на C&C сервер посредством запроса POST.

В противном случае используется другой уровень кодирования. В этом случае блок данных подписан, и для шифрования используется случайный ключ 3DES. Поскольку ключ 3DES генерируется случайным образом, и он нужен серверу для дешифрования данных, ключ шифруется открытым ключом сервера. Ключ сервера извлекается из поля public секции [CRYPTO] конфигурационного файла.

Новый BLOB (encrypted_key | signature_data | encrypted data) кодируется в base64 и отправляется на управляющий сервер посредством запроса POST.

Чтобы избежать обнаружения на основе объема данных, отправленных в запросе, BLOB можно разбить на несколько пакетов. Опция в конфигурационном файле (post_frag в секции [TRANSPORT]) определяет, будет ли объект фрагментирован или отправлен в одном запросе POST.

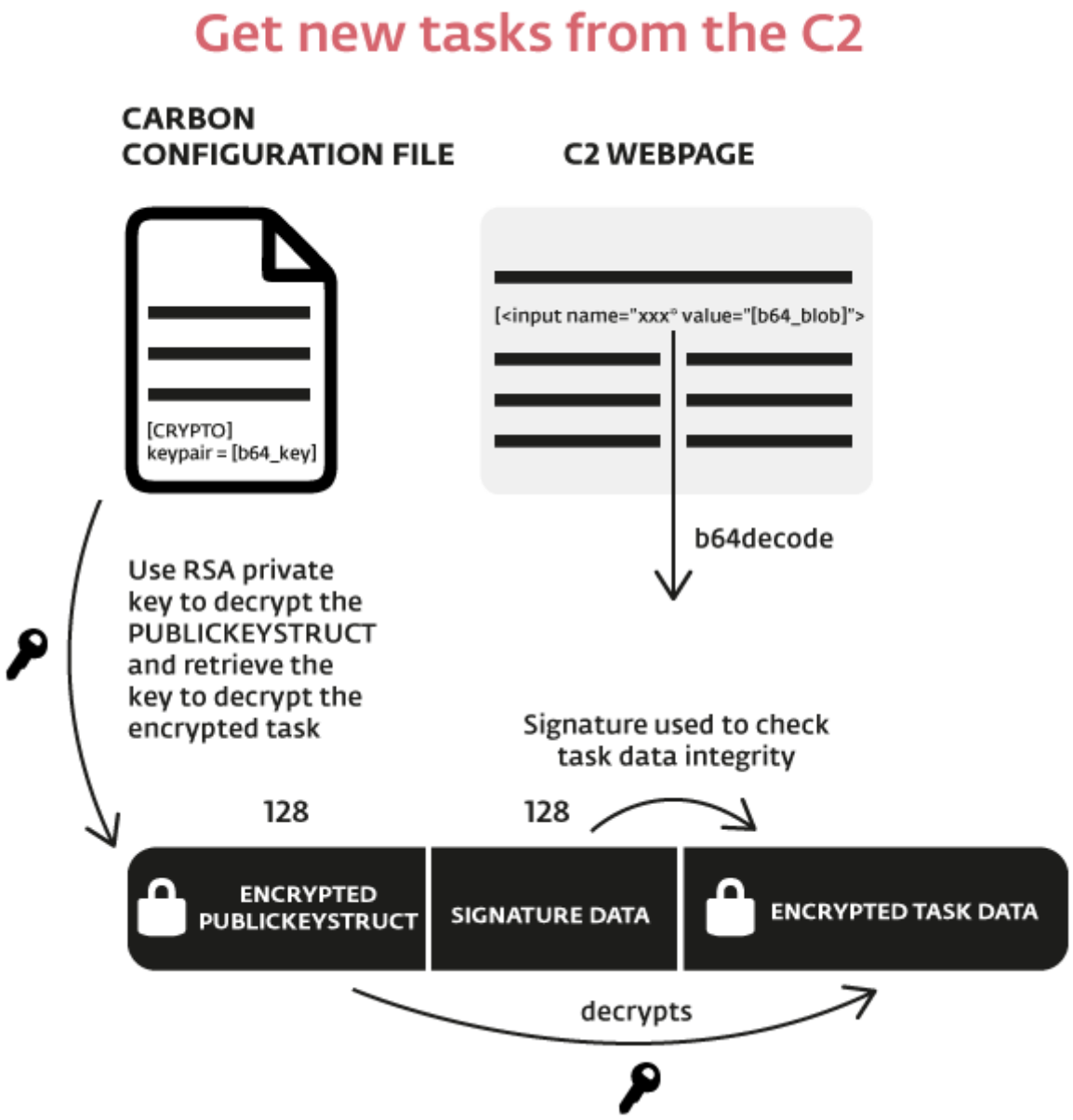
Если для опции задано значение yes, объект делится на несколько фрагментов определенного размера. Размер задается другим полем конфигурационного файла – post_frag_size.

В запрос будет добавлен дополнительный заголовок:

- Content-Range: bytes %u-%u/%u; id=%u\r\n”, i, i+(fragment_size-1), data_size, task_id
- Если установлена опция http11, будет также добавлен определенный заголовок:
- Expect: 100-continue\r\n

Для каждого отправляемого фрагмента поля post_frag_size и pfslastset из конфигурационного файла (секция [CW_INET_RESULTS]) обновляются вместе с размером фрагмента и меткой времени.

Получение новых задач



Новые задачи извлекаются с управляющего сервера путем анализа html-страницы. Вредоносная программа ожидает найти html-тег на странице с BLOB в кодировке base64 в атрибуте value. После декодирования объект содержит:

- зашифрованный блок из 128 байт, который содержит структуру PUBLICKEYSTRUC, за которой следует криптографический ключ (возможно, 3DES)
- данные подписи (128 байт) для проверки целостности следующего объекта
- блок зашифрованных данных, содержащий задачу

Вредоносная программа использует закрытый ключ RSA (извлекается из поля keypair секции [CRYPTO] конфигурационного файла), чтобы дешифровать первый блок, а затем использует только что расшифрованный ключ для дешифрования третьего блока. В этом блоке данных может быть:

1. Задача к исполнению:

- данные дешифрованы и хранятся в нескольких временных файлах, задача (команда или PE-файл) и ее конфигурационный файл – в папке Nls, файлы результатов (содержат результаты выполнения задач и файл журнала) – в папке 0208 (0409 и 1033 для Carbon v3.8x)
- task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath
- эта строка добавляется к началу файла b9s3coff.ax (cifrado.xml в версии v3.8x)

2. Задача для исполнения оркестратором:

- данные дешифрованы и хранятся в нескольких временных файлах (задача, ее конфигурационный файл и др.) в папках Nls и 0208 (0409 и 1033 для Carbon v3.8x)
- в зависимости от содержимого данных, одна из следующих строк добавляется в начало файла a67ncode.ax (encodebase.info в версии v3.8x)

```
task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath
task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath |
execution_mode | username | password
task_id | task_filepath | task_config_filepath | task_result_filepath | task_log_filepath |
execution_mode
```

3. Новый открытый RSA ключ сервера

- в этом случае конфигурационный файл обновляется с новым ключом в кодировке base64 (поле publicc)

4. Данные для отправки другому экземпляру Carbon, запущенному на другом компьютере в данной сети

- данные должны содержать определенный IP-адрес и порт, именованный канал, возможно, с логином и паролем.

Проверка интернет-доступа

Соединение с интернетом проверяется каждый час. Первая проверка выполняется вызовом функции InternetAttemptConnect(). Если это работает, другой тест осуществляется путем отправки запросов HTTP GET на следующие сайты:

- google.com
- yahoo.com
- bing.com
- update.microsoft.com
- windowsupdate.microsoft.com
- microsoft.com

Событие используется для уведомления других потоков в случае потери доступа к интернету.

Резервное копирование конфигурационного файла

Как в файле журнала, конфигурационный файл также периодически копируется и отправляется на управляющий C&C сервер. Поток исполняет код в определенном временном диапазоне (по умолчанию между 8 и 20).

Значение configlastsend извлекается из секции [TIME] конфигурационного файла. Если конфигурационный файл был отправлен больше месяца назад, он копируется во временный файл со случайным названием с префиксом ~D в папке 208 (Carbon v3.7x) или 1033 (Carbon v3.8x). Затем файл шифруется с применением алгоритма CAST-128.

Чтобы уведомить поток, который обращается к командному серверу, что новый файл готов к отправке на C&C, следующая строка добавляется к файлу C_56743.NLS (для Carbon v3.7x) или dsntype.gif (Для Carbon v3.8x):

“11|1|%s|%s”

- 1-е поле: ID для идентификации файла как конфигурационного
- 2-е поле: 1 (файл для отправки на C&C сервер)
- 3-е поле: временный путь к файлу
- 4-е поле: uuid жертвы

Наконец, атрибут configlastsend обновляется в текущем времени.

Дополнения

Вызов функций API

Базовый адрес интересующих модулей извлекается путем анализа PEB или, если модули не загружаются в память процесса, путем загрузки необходимых файлов с диска в память и анализа их заголовков.

Как только базовые адреса получены, PEB будет пройден снова, а поле LoadCount из структуры LDR_DATA_TABLE_ENTRY будет проверено. Это значение используется для подсчета ссылок, чтобы отслеживать загрузку и выгружать модуль.

Если LoadCount положительный, модуль EAT анализируется для получения необходимого адреса функции.

Шифрование

Названия модулей и функций зашифрованы простым образом (начиная, как минимум, с версии v3.77, в отличие от v3.71), для каждого символа применяется логический сдвиг в 1 бит.

Имена процессов также зашифрованы, здесь для каждого символа применяется XOR с ключом 0x55 (для Carbon v3.7x, по крайней мере, с версии 3.77) и ключом 0x77 для Carbon v3.8x.

Всего несколько исключений – каждый файл из рабочего каталога Carbon зашифрован алгоритмом CAST-128 в режиме OFB. Тот же ключ и IV используются с версии 3.71 до версии 3.81:

- key = “\x12\x34\x56\x78\x9A\xBC\xDE\xF0\xFE\xFC\xBA\x98\x76\x54\x32\x10”

- IV = “\x12\x34\x56\x78\x9A\xBC\xDE\xF0”

Проверка запуска захвата пакетов

До взаимодействия с C&C сервером или другими компьютерами Carbon проверяет, что в системе не запущена ни одна из программ для захвата пакетов:

- TCPdump.exe
- windump.exe
- ethereal.exe
- wireshark.exe
- ettercap.exe
- snoop.exe
- dsniff.exe

Если хотя бы один из процессов запущен, обращения к серверу не будет.

Приложения

Индикаторы заражения Carbon доступны на GitHub.

Правила для YARA

```
import “pe”
rule generic_carbon
{
strings:
$s1 = “ModStart”
$s2 = “ModuleStart”
$t1 = “STOP|OK”
$t2 = “STOP|KILL”
condition:
(uint16(0) == 0x5a4d) and (1 of ($s*)) and (1 of ($t*))
}
rule carbon_metadata
{
condition:
(pe.version_info[“InternalName”] contains “SERVICE.EXE” or
pe.version_info[“InternalName”] contains “MSIMGHLP.DLL” or
pe.version_info[“InternalName”] contains “MSXIML.DLL”)
and pe.version_info[“CompanyName”] contains “Microsoft Corporation”
}
```

Шифратор / дешифратор файлов Carbon

```
#!/usr/bin/env python2
from Crypto.Cipher import CAST
import sys
import argparse
def main():
parser =
argparse.ArgumentParser(formatter_class=argparse.RawTextHelpFormatter)
parser.add_argument("-e", "-encrypt", help="encrypt carbon file",
required=False)
parser.add_argument("-d", "-decrypt", help="decrypt carbon file",
required=False)
try:
args = parser.parse_args()
except IOError as e:
parser.error(e)
return 0
if len(sys.argv) != 3:
parser.print_help()
return 0
key = "\x12\x34\x56\x78\x9A\xBC\xDE\xF0\xFE\xFC\xBA\x98\x76\x54\x32\x10"
iv = "\x12\x34\x56\x78\x9A\xBC\xDE\xF0"
cipher = CAST.new(key, CAST.MODE_OFB, iv)
if args.encrypt:
plaintext = open(args.encrypt, "rb").read()
while len(plaintext) % 8 != 0:
plaintext += "\x00"
data = cipher.encrypt(plaintext)
open(args.encrypt + "_encrypted", "wb").write(data)
else:
ciphertext = open(args.decrypt, "rb").read()
while len(ciphertext) % 8 != 0:
ciphertext += "\x00"
data = cipher.decrypt(ciphertext)
open(args.decrypt + "_decrypted", "wb").write(data)
if __name__ == "__main__":
main()
```

Образцы Carbon

SHA1
7f3a60613a3bdb5f1f8616e6ca469d3b78b1b45b
a08b8371ead1919500a4759c2f46553620d5a9d9
4636dccac5acf1d95a474747bb7bcd9b1a506cc3
cbde204e7641830017bb84b89223131b2126bc46
1ad46547e3dc264f940bf62df455b26e65b0101f
a28164de29e51f154be12d163ce5818fceb69233
7c43f5df784bf50423620d8f1c96e43d8d9a9b28
7ce746bb988cb3b7e64f08174bdb02938555ea53
20393222d4eb1ba72a6536f7e67e139aadfa47fe
1dbfcb9005abb2c83ffa6a3127257a009612798c
2f7e335e092e04f3f4734b60c5345003d10aa15d
311f399c299741e80db8bec65bbf4b56109eedaf
fbc43636e3c9378162f3b9712cb6d87bd48ddbd3
554f59c1578f4ee77dbba6a23507401359a59f23
2227fd6fc9d669a9b66c59593533750477669557
87d718f2d6e46c53490c6a22de399c13f05336f0
1b233af41106d7915f6fa6fd1448b7f070b47eb3
851e538357598ed96f0123b47694e25c2d52552b
744b43d8c0fe8b217acf0494ad992df6d5191ed9
bcf52240cc7940185ce424224d39564257610340
777e2695ae408e1578a16991373144333732c3f6
56b5627debb93790fdbcc9ecbffc3260adeafbab

SHA1
678d486e21b001deb58353ca0255e3e5678f9614

C&C серверы
soheylistore.ir:80:/modules/mod_feed/feed.php
tazohor.com:80:/wp-includes/feed-rss-comments.php
jucheafrica.com:80:/wp-includes/class-wp-edit.php
61paris.fr:80:/wp-includes/ms-set.php
doctorshand.org:80:/wp-content/about/
www.lasac.eu:80:/credit_payment/url/