АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Gazer: бэкдор второго этапа APT-группы Turla

31 августа 2017 года

В ESET выполнили анализ нового, ранее недокументированного бэкдора, который с большой долей вероятности используется APT-группой Turla. Бэкдор замечен в атаках на правительственные и дипломатические учреждения в странах Европы. Turla – кибергруппа, которая специализируется на операциях кибершпионажа. Типичные методы хакеров Turla – атаки watering hole и целевой фишинг. Новый бэкдор активно используется как минимум с 2016 года. На основе строк, найденных в изученных образцах, мы назвали его Gazer.

Недавно о хакерах Turla вспомнила пресса, чего не случалось довольно давно. Журналисты The Intercept напомнили о презентации Управления защиты связи Канады (CSE – Communication Security Establishment), в которой перечислены ошибки операторов Turla. Авторы презентации называют кибергруппу MAKERSMARK, краткое содержание – на слайде ниже:

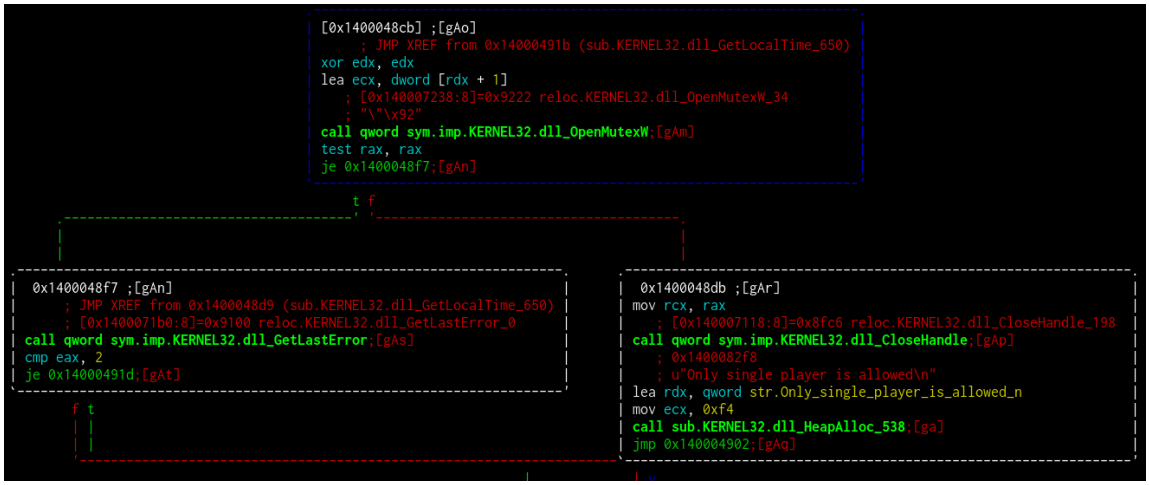
Gazer, подобно другим инструментам Turla, использует передовые методы для шпионажа и обеспечения устойчивости в целевых системах. Рассказываем об атаках с использованием Gazer и представляем технический анализ функциональных возможностей бэкдора.

Резюме

По данным телеметрии ESET, Gazer заражены компьютеры в ряде стран мире, но преимущественно в странах Европы. Техника, тактика и процедуры (TTPs) соответствуют индикаторам, которые мы обычно наблюдаем в работе Turla: бэкдор первого этапа (такой как Skipper), вероятно, доставленный путем целевого фишинга; появление в зараженной системе бэкдора второго этапа, в данном случае Gazer. По данным наших исследований, основные цели Gazer находятся в Юго-Восточной Европе и бывших союзных республиках.

В настоящее время у нас нет исчерпывающих доказательств принадлежности Gazer группе Turla, но на это указывает несколько подсказок. Во-первых, целевые объекты соответствуют сфере интересов Turla – министерства иностранных дел и посольства. Во-вторых, на Turla указывают методы – целевой фишинг, установка бэкдора первого этапа, скрытый бэкдор второго этапа. В большинстве случаев вместе с Gazer обнаруживается бэкдор первого этапа Skipper, также связанный с Turla. В-третьих, есть ряд общих черт Gazer и других бэкдоров второго этапа, используемых Turla, таких как Carbon и Kazuar.

Как обычно, группа Turla стремится избежать обнаружения инструментов путем безопасного удаления файлов, изменения строк и рандомизации. В последней изученной версии авторы Gazer модифицировали большую часть строк, добавив в код предложения, связанные с видеоиграми. Пример на рисунке ниже:



Сходство с другими инструментами Turla

Gazer написан на C++ и имеет общие черты с другими инструментами Turla. Бэкдоры Gazer, Carbon и Kazuar получают с удаленного C&C-сервера зашифрованные задачи, которые могут выполняться как в зараженной системе, так и на другой машине в составе сети. Все они используют зашифрованное хранилище для компонентов и конфигурации вредоносной программы, а также записывают операции в файле.

Список C&C-серверов зашифрован и встроен в ресурсы PE Gazer. Это легитимные скомпрометированные сайты (преимущественно на Wordpress), которые выступают в роли прокси первого уровня. Тактика характерна для группы Turla.

Еще одна интересная связь – один из C&C-серверов, встроенный в образец Gazer, использовался бэкдором на JScript KopiLuwak, разобранном Лабораторией Касперского.

Последнее, но не менее важное. Три семейства вредоносных программ (Gazer, Carbon и Kazuar) имеют аналогичный список процессов, которые могут использоваться в качестве цели для внедрения модуля для связи с C&C-сервером. Ресурс, содержащий этот список, может меняться от одного образца к другому. Он, скорее всего, адаптирован к тому, что установлено в системе (например, в некоторых образцах в списке появляется процесс safari.exe).

Кастомное шифрование

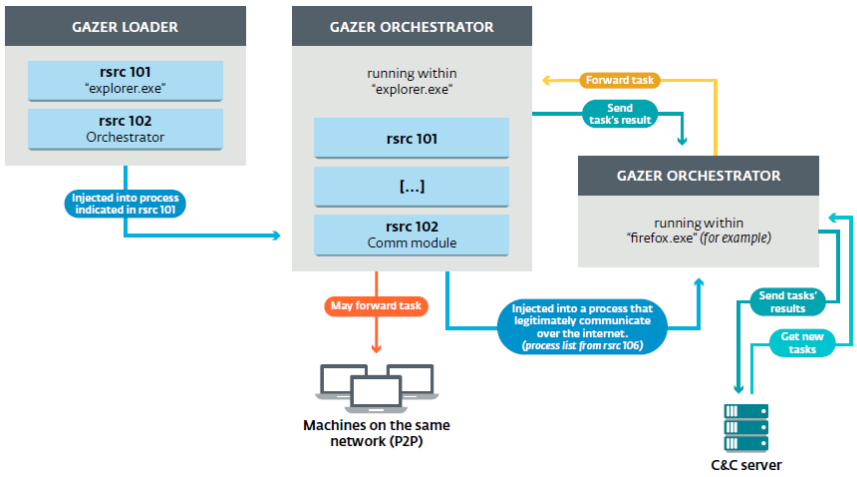
Авторы Gazer активно используют шифрование. Похоже, что вместо Windows Crypto API и других общедоступных библиотек, они используют собственную библиотеку для 3DES и RSA.

Ключи RSA, встроенные в ресурсы, содержат открытый ключ атакующих, который используются для шифрования данных, отправляемых на C&C-сервер, и закрытый ключ для расшифровки ресурсов в двоичном файле. В каждом образце свои уникальные ключи.

Эти ресурсы структурированы тем же образом, что RSA from OpenSSL, но их значения (p, q и др.) вычисляются в пользовательской реализации авторов Gazer.

Глобальная архитектура

В разделе мы опишем в деталях компоненты Gazer — его архитектура представлена на рисунке ниже:



Загрузчик

Загрузчик – первый компонент вредоносной программы, который выполняется в системе. Два ресурса сохраняются незашифрованными в двоичном файле:
101: имя процесса, в который внедряется оркестратор
102: оркестратор

Следующий мьютекс обеспечивает выполнение только одного экземпляра вредоносной программы:
{531511FA-190D-5D85-8A4A-279F2F592CC7}

Создание именованного канала

Чтобы установить канал связи между компонентами Gazer, запущен именованный канал. Он создается из этой строки:
\\\\.\\pipe\\Winsock2\\CatalogChangeListener-FFFF-F

Шаблон FFFF-F заменяется значениями, вычисленными из идентификатора безопасности (SID) текущего пользователя и метки времени.

Рассмотрим, например, текущую дату 2017/04/24 и SID: S-1-5-21-84813077-3085987743-2510664113-1000

Чтобы создать шаблон в конце именованного канала, выполняются следующие вычисления:

time = SystemTime.wDay * Systemtime.wMonth * SystemTime.wYear = 24 * 04 * 2017 = 0x2f460
xsid = (1 * 21 * 84813077 * 3085987743 * 2510664113 * 1000) & 0xFFFFFFFF = 0xefa252d8

((time >> 20) + (time & 0xFFF) + ((time >> 12) & 0xFFF)) % 0xFF = 0x93
((xsid >> 20) + (xsid & 0xFFF) + ((xsid >> 12) & 0xFFF)) % 0xFF = 0x13

((time * xsid >> 24) + (uint8_t)(time * xsid) + ((uint16_t)(time * xsid) >> 8) + (uint8_t)(time * xsid >> 16)) % 0xf) = 0xa

Именованный канал в этом случае:
\\\\.\\pipe\\Winsock2\\CatalogChangeListener-9313-a

Если SID текущего пользователя не может быть восстановлен, по умолчанию используется следующий именованный канал:
\\\\.\\pipe\\\\Winsock2\\CatalogChangeListener-FFFE-D

Инъекция кода через перехват потока

Для внедрения оркестратора в удаленный процесс используется не слишком распространенный метод.

Выполняемый поток из удаленного процесса перехватывается для запуска шелл-кода, который будет выполнять точку входа коммуникационного модуля.

1. Модуль и шелл-код копируются в удаленный процесс;
2. Функция ZwQuerySystemInformation используется для извлечения общего числа выполняемых потоков в целевом процессе;
3. В каждом потоке выполняются следующие операции:

- поток приостановлен функцией OpenThread/SuspendThread;
- контекст потока извлекается с помощью GetThreadContext;
- указатель команд потока сохраняется и модифицируется, чтобы указывать на шелл-код (через SetThreadContext);
- поток возобновляется с помощью ResumeThread.

4. Если одна из предыдущих операций не выполнена, поток возобновляется и те же операции производятся в другом потоке.

```
launcher:
push rax
sub rsp, 38h
movabs rax, 5D20092 ; @ end of payload
mov qword ptr ss:[rsp+28], rax ; lpThreadId
mov qword ptr ss:[rsp+20], 0 ; dwCreationFlags
xor r9d, r9d ; lpParameter
movabs r8, 5D20046 ; lpStartAddress => @payload
xor edx, edx ; dwStackSize = 0
xor ecx, ecx ; lpThreadAttributes = NULL
call qword ptr ds:[CreateThread]
movabs rax, 90A7FACE90A7FACE ; replaced by the saved instruction pointer from thread context ;)
add rsp, 38h
xchg qword ptr ss:[rsp], rax
ret

payload:
sub rsp, 28
movabs r8, 5D20096
```

```
mov edx, 1
movabs rcx, 4000000000000000
call qword ptr ds: [DllEntryPoint]
xor ecx, ecx
call ExitThread
int 3
xxxx; @DllEntryPoint
xxxx ; @CreateThread
xxxx; @ExitThread
xxxx
xxxx
xxxx
xxxx ; TID
```

Шелл-код – загрузчик, который будет выполнять точку входа модуля в новом потоке.

Сохранение данных

Загрузчик отправляет двоичные данные через именованный канал в оркестратор. Blob содержит:

- идентификатор команды: CMC_TAKE_LOADER_BODY
- путь к файлу загрузчика
- PE загрузчика

Когда сообщение получено оркестратором, загрузчик безопасно удаляется путем перезаписи содержимого файла и с помощью функции DeleteFile.

Информация о персистентности извлекается из ресурса 105 и сохраняется в Gazer. Среди этих данных есть значение dword, которое используется для выбора режима сохранения данных.

Ресурс 105 структурирован следующим образом:

- значение dword показывает режим сохранения данных
- значение dword показывает объем данных
- информация о сохранении данных

Существует шесть различных режимов сохранения данных.

0: ShellAutorun
Персистентность достигается через реестр Windows путем добавления параметра Shell со значением explorer.exe, %malware_pathfile% для следующего ключа:
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon

1: HiddenTaskAutorun
Очень похоже на метод TaskScheduler Autorun (4), описанный ниже. Главное отличие – задача скрыта от пользователя с помощью признака TASK_FLAG_HIDDEN (устанавливается через метод SetFlags из интерфейса Itask).

2: ScreenSaverAutorun
В этом режиме Gazer обеспечивает персистентность, установив в системном реестре Windows исполняемый файл, используемый в качестве скринсейвера.

Большинство параметров создаётся в ветке реестра HKCU\Control Panel\Desktop:

- SCRNSAVE.exe путь к вредоносному исполняемому файлу
- значение ScreenSaveActive – 1: включить скринсейвер
- значение ScreenSaverIsSecure – 0: скринсейвер не защищен паролем
- ScreenSaveTimeout устанавливается значение, заданное в ресурсе. Указывает, как долго система остается в режиме ожидания перед запуском скринсейвера (в данном случае вредоносного ПО).

3: StartupAutorun
Если ресурс 105 начинается со значения dword 3, в меню «Пуск» будет создан LNK-файл. Ресурс также предоставляет описание файла ярлыков, путь и имя файла для LNK.

Интерфейс IshellLink используется для создания ссылки на оболочку.

4: TaskSchedulerAutorun
Метод используется для достижения персистентности путем создания запланированной задачи. Задача создается и настраивается через интерфейсы COM, связанные с задачами (ITaskService, ITaskSettings, …).

Некоторая информация, в частности, имя задачи и ее описание, извлекается из ресурса. Например, в одном из ресурсов образца режим сохранения данных установлен на 04 (TaskSchedulerAutorun) с данными:

%APPDATA%\Adobe\adobeup.exe Adobe Acrobat Reader Updater. This task was generated by Adobe Systems, Inc to keep your Adobe Software up-to-data. \Adobe\AcrobatReader.Adobe

В этом примере запланированная задача создается и настраивается следующим образом:

1. Название задачи: "Adobe Acrobat Reader Updater"
2. Исполняемый файл: "%APPDATA%\Adobe\adobeup.exe"
Оркестратор скопирует загрузчик, полученный по именованному каналу в этой локации
3. Описание задачи: "Задача создана Adobe Systems, Inc для обновлений вашего программного обеспечения Adobe"
4. Папка задачи: \Adobe\AcrobatReader.Adobe

Наконец, задача сконфигурирована для запуска планировщиком задач в любое время по прошествии запланированного. Задача будет запущена, когда текущий пользователь войдет в систему.

5: LinkAutorun
Метод изменяет существующие LNK-файлы для выполнения вредоносного ПО через cmd.exe. Для каждого файла LNK в папке, заданной в ресурсе, значок и аргументы удаляются. Путь в cmd.exe задается с аргументом:
/q /c start "%s" && start "%s"

В большинстве изученных нами образцов файл конфигурации указывает, что должен использоваться метод TaskSchedulerAutorun.

Журналы

Все три компонента Gazer записывают действия в журнальные файлы. Они зашифрованы с помощью одного и того же алгоритма – 3DES.

В некоторых версиях Gazer легко получить эти файлы, поскольку их имена жестко закодированы в двоичных файлах:

- %TEMP%\CVRG72B5.tmp.cvr: журналы загрузчика
- %TEMP%\CVRG1A6B.tmp.cvr: журналы оркестратора
- %TEMP%\CVRG38D9.tmp.cvr: журналы коммуникационного модуля

Каждый журнальный файл структурирован следующим образом:
[LOGSIZE][DECRYPTION_KEY][ENCRYPTED_LOG]

- logsize: когда это значение (2 байта) вычитается из магического числа 0xf18b, получаем размер зашифрованного журнала.
- decryption_key: когда этот 12-байтный blob зашифрован при помощи XOR с другим жестко закодированным ключом из 12 байтов, получаем ключ 3DES, который может использоваться для расшифровки журнала
- encrypted_log: журнал зашифрован с помощью алгоритма 3DES в режиме CBC

После расшифровки каждая запись журнала форматируется следующим образом:
|Hour:Min:Sec:Ms| [log ID] [log]

Ниже пример с расшифрованным журнальным файлом оркестратора:

```
|10:29:56:197| [1556]
|10:29:56:197| [1557]
**********************************************************************************
*****
|10:29:56:197| [1558] DATE: 25.05.2017
|10:29:56:197| [1559] PID=900 TID=2324 Heaps=32 C:\Windows\Explorer.EXE
|10:29:56:197| [1565] DLL_PROCESS_ATTACH
|10:29:56:197| [1574] 4164
|10:29:58:197| [0137]
==========================================================================
|10:29:58:197| [0138] Current thread = 2080
|10:29:58:197| [0183] Heap aff0000 [34]
|10:29:58:197| [0189] ### PE STORAGE ###
|10:29:58:197| [0215] ### PE CRYPTO ###
|10:29:58:197| [0246] ### EXTERNAL STORAGE ###
|10:29:58:197| [1688] Ok
|10:29:58:197| [0279] Path =
\HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ScreenSaver
|10:29:58:197| [0190] \HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ScreenSaver
|10:29:58:197| [0338] ---FAILED
|10:29:58:197| [0346] Initializing standart reg storage...
|10:29:58:197| [0190] Software\Microsoft\Windows\CurrentVersion\Explorer\ScreenSaver
|10:29:58:197| [2605] Storage is empty!
|10:29:58:197| [0392] ### EXTERNAL CRYPTO ###
```

```
|10:29:59:666| [1688] Ok
|10:29:59:713| [1473] Ok
|10:29:59:760| [1688] Ok
|10:29:59:775| [1473] Ok
|10:29:59:775| [1688] Ok
|10:29:59:775| [1473] Ok
|10:29:59:791| [1688] Ok
|10:29:59:791| [1473] Ok
|10:29:59:806| [1688] Ok
|10:29:59:806| [1473] Ok
|10:29:59:806| [0270] 08-00-27-90-05-2A
|10:29:59:806| [0286] _GETSID_METHOD_1_
|10:29:59:806| [0425] 28 7 8 122
|10:29:59:806| [0463] S-1-5-21-84813077-3085987743-2510664113-1000
|10:29:59:806| [0471]
|10:29:59:806| [0787] Ok
|10:29:59:806| [1473] Ok
|10:29:59:822| [0514] ### QUEUES ###
|10:29:59:822| [0370] T Empty
|10:29:59:822| [0482] R Empty
|10:29:59:822| [1754] Ok
|10:29:59:822| [1688] Ok
|10:29:59:822| [1473] Ok
|10:29:59:838| [0505] R #4294967295 PR_100 TR_00000000 SZ_172 SC_0(50) --+- EX_0
|10:29:59:838| [0625] ### TRANSPORT ###
|10:29:59:838| [0286] _GETSID_METHOD_1_
|10:29:59:838| [0425] 28 7 25 122
|10:29:59:838| [0463] S-1-5-21-84813077-3085987743-2510664113-1000
|10:29:59:838| [0471]
|10:29:59:838| [0165] \\.\pipe\Winsock2\CatalogChangeListener-2313-4
|10:29:59:838| [0131] PipeName = \\.\pipe\Winsock2\CatalogChangeListener-2313-4
|10:29:59:838| [0041] true
[...]
```

Обратите внимание, что в старых версиях Gazer «log ID» заменен именем текущей функции.

Рабочий каталог

Использование системного реестра Windows

Все файлы, связанные с Gazer (кроме журналов), хранятся в зашифрованном виде в реестре. Ресурс оркестратора 109 содержит корневой путь хранилища (в отчете называется %RootStoragePath%). В каждом изученном нами образце этот ресурс указывал на тот же путь хранения:
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ScreenSaver

Если этот ресурс пуст, по умолчанию используется раздел реестра. За исключением RSA ключей, все данные в хранилище зашифрованы.

Создается несколько подкаталогов (их имена жестко закодированы в двоичном формате):

%RootStoragePath%{119D263D-68FC-1942-3CA3-46B23FA652A0}
Object ID: уникальный ID для идентификации жертвы

%RootStoragePath%{1DC12691-2B24-2265-435D-735D3B118A70}
Task Queue: связанный список выполняемых задач

%RootStoragePath%{28E74BDA-4327-31B0-17B9-56A66A818C1D}
Плагины

%RootStoragePath%{31AC34A1-2DE2-36AC-1F6E-86F43772841F}
Communication Module: DLL для связи с C&C-сервером

%RootStoragePath%{3CDC155D-398A-646E-1021-23047D9B4366}
Autorun: метод обеспечения персистентности

%RootStoragePath%{4A3130BD-2608-730F-31A7-86D16CE66100}
Local Transport Settings: IP-адреса компьютеров в одной сети

%RootStoragePath%{56594FEA-5774-746D-4496-6361266C40D0}
Last Connection: время последнего соединения с C&C-сервером (SYSTEMTIME)

%RootStoragePath%{629336E3-58D6-633B-5182-576588CF702A}
RSA Private Key: генерируется в процессе, используется для расшифровки данных Gazer

%RootStoragePath%{6CEE6FE1-10A2-4C33-7E7F-855A51733C77}
Result Queue: связанный список результатов выполнения задач

%RootStoragePath%{81A03BF8-60AA-4A56-253C-449121D61CAF}
Inject Settings: список процессов, которые используются для внедрения коммуникационного модуля

%RootStoragePath%{8E9810C5-3014-4678-27EE-3B7A7AC346AF}
C&C-серверы

Использование альтернативных потоков данных

Если доступ к реестру недоступен, эти элементы конфигурации сохраняются с использованием альтернативных потоков данных.

Вызывается функция GetVolumeInformation(), чтобы гарантировать, что диск C:\\ поддерживает именованные потоки для использования ADS.

Те же идентификаторы GUID, что и выше, используются для скрытия данных в ADS для файла (жестко закодированы в двоичном формате):
"%TEMP%\\KB943729.log"

Например, здесь полный путь для доступа к ID объекта:
%TEMP%\KB943729.log:{1DC12691-2B24-2265-435D-735D3B118A70}

Оркестратор

Ресурсы Gazer

Файлы, связанные с Gazer, хранятся в ресурсах оркестратора.

Формат файла

В общей сложности 11 ресурсов (от 101 до 111) структурировано следующим образом:
DATATYPE | SIZE | DATA | PADDING

DATATYPE: dword определяет тип данных ресурса:

- 0: первичные данные
- 0xFFFFFFFF: пустой
- 0x4: не определено
- 0x1030001: массив строк
- 0x1: двоичный формат

SIZE: объем данных (без заполнения)

Шифрование

За исключением ресурсов 101 и 102, являющимися ключами RSA, каждый ресурс упакован с помощью Bzip и зашифрован с 3DES.

[RSAEncryptedBlob][SignatureBlob][3DESBlob]

- RSAEncryptedBlob: первые 1024 бит данных представляют собой blob, который содержит ключ 3DES. Blob зашифрован с использованием RSA и может быть расшифрован с помощью ресурса 101.
- SignatureBlob: вторая часть данных – 1024-битный blob, содержащий подпись последней части данных, которая была расшифрована.
- 3DESBlob: последняя часть – актуальные данные, которые зашифрованы с ключом 3DES из первого blob.

Каждый ресурс расшифровывается динамически. Подпись сверяется с расшифрованными данными для проверки целостности. Далее ресурс будет зашифрован новым ключом RSA, созданным в коде оркестратора случайным образом. Закрытый ключ и зашифрованный ресурс сохраняются в реестре под определенным подключом GUID.

Список ресурсов:

- 101: ресурс содержит закрытый RSA ключ. Он используется для расшифровки других ресурсов.
- 102: открытый ключ RSA.
- 103: пустой
- 104: не определено
- 105: информация о сохранении данных
- 106: список процессов, которые используются для внедрения коммуникационного модуля
- 107: DLL для связи с C&C-серверами

- 108: список C&C-серверов
- 109: путь к рабочему каталогу Gazer
- 110: список плагинов
- 111: информация о локальной передаче данных

Выполнение задачи

Задача, полученная с C&C-сервера, выполняется зараженной машиной или другим компьютером сети через механизм P2P (как в Carbon и Snake).

Возможные задачи:

- отправить файл
- загрузить файл
- обновить конфигурацию
- выполнить команду

Результат сохраняется в очереди и будет отправлен модулю, который обратится к C&C-серверу, когда будет доступен интернет.

Иерархия классов

Вредоносная программа написана на C++ и RTTI, который содержит информацию об объектах, используемых в коде, не перезаписывается.

Существует пять абстрактных классов:

Table 1. **Abstract Class Autorun**

Class Name
LinkAutorun
StartupAutorun
ShellAutorun
ScreenSaverAutorun
TaskSchedulerAutorun
HiddenTaskAutorun

Table 2. **Abstract Class Queue**

Class Name
TaskQueue
ResultQueue

Table 3. **Abstract Class Storage**

Class Name
ExeStorage
FSStorage
RegStorage

Table 4. **Abstract Class TListenerInterface**

Class Name
LTMessageProcessing
CMessageProcessingSystem

Table 5. **Abstract Class TAbstractTransport**

Class Name
LTNamedPipe
TNPTransport

Коммуникационный модуль

Коммуникационный модуль используется для извлечения задач с C&C-сервера и передачи их оркестратору.

Библиотека внедряется в процесс, который может легитимно обращаться к интернету. Библиотека такая же, как в загрузчике, которая используется для ввода оркестратора в explorer.exe.

Инициализация связи

Если прокси-сервер существует, он извлекается и используется Gazer для выполнения HTTP-запросов. Для получения этого значения есть два метода:
запросить данный раздел реестра HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings или использовать функцию InternetQueryOption с признаком INTERNET_OPTION_PROXY, если прокси-сервер не может быть получен через реестр.

Далее настраивается агент пользователя системы:

- извлекается значение ключа данных User Agent для ключа HKCU\Software\Microsoft\Windows\Current Version\Internet Settings;
- перечислены значения ключей в разделе HKLM\Software\Microsoft\Windows\Current Version\Internet Settings\5.0\User Agent\Post Platform; те из них, которые имеют в качестве данных IEAK, соединены с агентом пользователя;
- если в реестре не обнаружен агент пользователя, используется жестко кодированный UA Mozilla/4.0 (compatible; MSIE 6.0)

До попытки обращения к C&C-серверу проверяется подключение к интернету. Один за другим запрашиваются следующие серверы:

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

Связь с C&C-сервером

Вредоносная программа взаимодействует с C&C-сервером для извлечения задач (через HTTP GET запросы) и отправки результатов выполнения (через HTTP POST запросы).

Перед отправкой запроса на сервер, в оркестратор передается команда CMC_GIVE_SETTINGS через канал связи (именованный канал, подробнее в следующем разделе). Сообщение (MSG), содержащееся в пакете, в этом случае представляет собой один байт, установленный оркестратором для статуса

Перед отправкой запроса на C & C команда CMC_GIVE_SETTINGS отправляется в оркестр через их канал связи (именованный канал, подробнее об этом в следующем разделе). Сообщение CMC_GIVE_SETTINGS содержащееся в пакете, в этом случае представляет собой один байт, установленный оркестром для статуса о результате выполнения команды.

Оркестратор отвечает в том же канале с настройками, полученными из рабочего каталога с идентификатором объекта, списком C&C-серверов и датой последнего соединения.
Для получения задачи с C&C-сервера выполняется запрос GET.

Параметры запроса GET выбираются среди жестко закодированного списка ключевых слов, который не выглядит подозрительным. Значения генерируются случайным образом в кодировке [a-z0-9] со случайным размером для каждого параметра:
• id [6-12]
• hash [10-15]
• session [10-15]
• photo [6-10]
• video [6-10]
• album [6-10]
• client [5-10]
• key [5-10]
• account [6-12]
• member [6-12]
• partners [5-10]
• adm [6-12]

• author [6-12]
• contact [6-12]
• content [6-12]
• user [6-12]

Вот несколько примеров таких запросов:
xxx.php?album=2ildzq&key=hdr2a&partners=d2lic33f&session=nurvxd2x0z8bztz&video=sg508tujm&photo=4d4idgk
xxx.php?photo=he29zms5fc&user=hvbc2a&author=xvfj5r0q9c&client=7mvvc&partners=t4mgmuy&adm=lo3r6v4
xxx.php?member=ectwzo820&contact=2qwi15&album=f1qzoxuef4&session=x0z8bztz8hrs65f&id=t3x0ftu9
xxx.php?partners=ha9hz9sn12&hash=5740kptk3acmu&album=uef4nm5d&session=dpeb67ip65f&member=arj6x3ljj
xxx.php?video=nfqsz570&client=28c7lu2&partners=818eguh70&contact=ibj3xch&content=1udm9t799ixr&session=5fjjt61qred9uo

Для каждого запроса (отправить, получить, соединение) задается десятиминутный таймаут с помощью InternetSetOption.

После отправки запроса ответ обрабатывается только в том случае, если возвращаемый идентификатор состояния HTTP равен 404.

Содержание ответа шифруется и может быть расшифровано с помощью закрытого RSA-ключа, сгенерированного оркестратором. Он содержит blob и MD5-хеш данных.

Если размер ответа составляет 20 байт (4-байтный blob + хеш), для извлечения задач нет. Команда CMC_TAKE_TASK отправляется оркестратору с зашифрованной задачей, полученной от C&C-сервера, и ее размером. Оркестратор будет отвечать за выполнения задачи и отправит результаты в коммуникационный модуль. Полученный результат выполнения (зашифрованный оркестратором) будет передан на C&C-сервер с помощью запроса POST тем же способом, что и для запроса GET (используя параметры со случайными значениями).

Сообщения между компонентами

Глобальный именованный канал используется для связи между различными компонентами. Данные, отправленные через него, форматируются следующим образом:

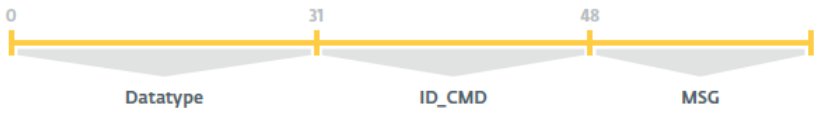


Рисунок 4. Формат сообщений

DATATYPE: те же константы используются для ресурсов (секция File Format в главе «Ресурсы»)
ID_CMD: имя команды (см. ниже)
MSG: данные для отправки

Список сообщений:

CMC_TAKE_TASK (ID_CMD: 1)

Задача, полученная с C&C-сервера, отправляется в оркестратор, который сохраняет ее в очереди задач.

CMC_TAKE_LOADER_BODY (ID_CMD: 2)
Удалить файл загрузчика Gazer, настроить копию загрузчика и его постоянное хранение в соответствии с одним из ресурсов

CMC_GIVE_RESULT (ID_CMD: 4)
Получив сообщение, оркестратор извлечет результат задачи из очереди результатов, упакует и зашифрует его с помощью открытого RSA ключа (ресурс 102) и отправит blob в коммуникационный модуль, который пошлет его на сервер (POST запрос).

CMC_GIVE_SETTINGS (ID_CMD: 5)
Коммуникационный модуль отправляет это сообщение оркестратору, чтобы запросить информацию, необходимую для контакта с сервером (список C&C-серверов, время последнего соединения, идентификатор жертвы).

CMC_TAKE_CONFIRM_RESULT (ID_CMD: 6)
Когда коммуникационный модуль отправляет результат выполнения задачи на сервер, в оркестратор отправляется сообщение, предписывающее удалить результат из очереди.

CMC_TAKE_CAN_NOT_WORK (ID_CMD: 7)
Если операция завершена неудачно (например, если коммуникационный модуль не может корректно интерпретировать данные, полученные от оркестратора), на оркестратор отправляется сообщение с кодом последней ошибки. Код можно добавить в журнальный файл.

CMC_TAKE_UNINSTALL (ID_CMD: 8)
Используется для удаления файлов с диска.

CMC_TAKE_NOP (ID_CMD: 9)
Нет операций

CMC_NO_CONNECT_TO_GAZER (ID_CMD: 0xA)
Команда отправляется в оркестратор, когда коммуникационный модуль не может связаться ни с одним сервером. В этом случае, если результаты задачи находятся в очереди, они будут зашифрованы и сохранены в систему Gazer.

CMC_TAKE_LAST_CONNECTION (ID_CMD: 0xB)
Команда направляется из коммуникационного модуля в оркестратор каждый раз, когда атакующие устанавливают связь с сервером контроллера. Содержит структуру SystemTime (с текущим системным временем). Когда сообщения получено оркестратором, время последнего соединения будет упаковано и зашифровано в хранилище Gazer (в реестре или в ADS).

CMC_GIVE_CACHE / CMC_TAKE_CACHE (ID_CMD: 0xC / 0xD)
Не реализовано

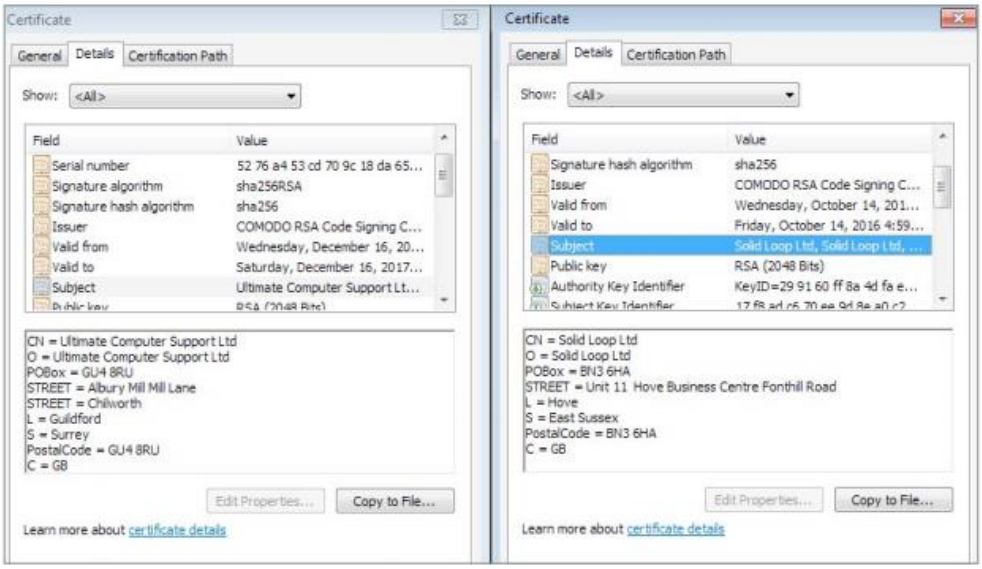
Версии Gazer

Обнаружено четыре версии вредоносной программы.

В первой версии функция, используемая для записи журналов, имеет в качестве параметра реальное имя функции. Существуют различные методы внедрения кода.

Во второй версии имена функций, используемые в качестве параметров, заменены идентификатором. Для инъекции кода используется только один метод.

Некоторые образцы из первых версий подписаны действующим сертификатом Comodo, выданным компании Solid Loop Ltd. Компиляция датирована 2002 годом, но может быть подделана, поскольку сертификат выпущен в 2015 году. Ниже сертификаты, используемые для подписи версий Gazer:



Последние версии подписаны другим сертификатом – Ultimate Computer Support Ltd. Были приняты некоторые усилия для обфускации строк, которые могут использоваться в качестве индикаторов компрометации. Имя мьютекса и именованный канал больше не отображаются в открытом тексте, теперь он кодируется ключом XOR.

В прошлых версиях имена журнальных файлов были жестко закодированы в двоичном файле. Сейчас для генерации случайного имени файла используется функция GetTempFileNameA.
В последних версиях, собранных в 2017 году, отличаются сообщения журнала (хотя и имеют одинаковое значение). Например, PE STORAGE заменено на EXE SHELTER, PE CRYPTO на EXE CIPHER и др.

Наконец, больше не используется поддельная временная отметка компиляции.

Полный список индикаторов компрометации доступен в нашем аккаунте на GitHub. По любым вопросам, связанным с Turla/Gazer, пишите на threatintel@eset.com

Индикаторы компрометации (IoCs)

Имена файлов:
%TEMP%\KB943729.log
%TEMP%\CVRG72B5.tmp.cvr
%TEMP%\CVRG1A6B.tmp.cvr
%TEMP%\CVRG38D9.tmp.cvr

%TEMP%\~DF1E06.tmp
%HOMEPATH%\ntuser.dat.LOG3
%HOMEPATH%\AppData\Local\Adobe\AdobeUpdater.exe

Разделы реестра:
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ScreenSaver
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Explorer\ScreenSaver

C&C:
daybreakhealthcare.co.uk/wp-includes/themees.php
simplecreative.design/wp-content/plugins/calculated-fields-form/single.php
169.255.137.203/rss_0.php
outletpiumini.springwaterfeatures.com/wp-includes/pomo/settings.php
zerogov.com/wp-content/plugins.deactivate/paypal-donations/src/PaypalDonations/SimpleSubsribe.php
ales.ball-mill.es/ckfinder/core/connector/php/php4/CommandHandler/CommandHandler.php
dyskurs.com.ua/wp-admin/includes/map-menu.php
warrixmalaysia.com.my/wp-content/plugins/jetpack/modules/contact-form/grunion-table-form.php
217.171.86.137/config.php
217.171.86.137/rss_0.php
shinestars-lifestyle.com/old_shinstar/includes/old/front_footer.old.php
www.aviasiya.com/murad.by/life/wp-content/plugins/wp-accounting/inc/pages/page-search.php
baby.greenweb.co.il/wp-content/themes/san-kloud/admin.php
soligro.com/wp-includes/pomo/db.php
giadinhvabe.net/wp-content/themes/viettemp/out/css/class.php
tekfordummies.com/wp-content/plugins/social-auto-poster/includes/libraries/delicious/Delicious.php
kennynguyen.esy.es/wp-content/plugins/wp-statistics/vendor/maxmind-db/reader/tests/MaxMind/Db/test/Reader/BuildTest.php
sonneteck.com/wp-content/plugins/yith-woocommerce-wishlist/plugin-fw/licence/templates/panel/activation/activation.php
chagiocaxuanson.esy.es/wp-content/plugins/nextgen-gallery/products/photocrati_nextgen/modules/ngglegacy/admin/templates/manage_gallery/gallery_preview_page_field.old.php
hotnews.16mb.com/wp-content/themes/twentysixteen/template-parts/content-header.php
zszinhyosz.pe.hu/wp-content/themes/twentyfourteen/page-templates/full-hight.php
weandcats.com/wp-content/plugins/broken-link-checker/modules/checkers/http-module.php

Мьютексы:
{531511FA-190D-5D85-8A4A-279F2F592CC7}