# OceanLotus: новый бэкдор, старые схемы

*30 августа 2018 года*

Группа OceanLotus (она же APT32 и APT-C-00) известна благодаря атакам в Восточной Азии. В прошлом году опубликован ряд исследований о работе группы, включая документы CyberReason, обзор FireEye и описание watering-hole атаки Volexity. Как мы видим, группа обновляет бэкдоры, инфраструктуру и векторы заражения.

OceanLotus продолжает деятельность, нацеленную на компании и госучреждения в странах Восточной Азии. По данным телеметрии ESET, приоритетные цели OceanLotus – во Вьетнаме, Лаосе, Камбодже и на Филиппинах.

Несколько месяцев назад мы обнаружили и проанализировали один из их новейших бэкдоров. В нем реализовано несколько инструментов, позволяющих затруднить анализ и избежать обнаружения – их и обсудим в посте.

## Распространение

Чтобы убедить жертву запустить вредоносный дроппер, атакующие используют разные методы.

## Двойные расширения и фейковые иконки приложений (Word, PDF и т.д.)

Есть вероятность, что дропперы распространяются через вложения по электронной почте. Мы наблюдали следующие имена файлов:
— `Mi17 Technical issues - Phonesack Grp.exe` (Ми-17 – российская модель вертолета)
— `Chi tiet don khieu nai gui saigontel.exe` (в переводе с вьетнамского – «детали претензии, направленной в Сайгонтел», Сайгонтел – вьетнамская телекоммуникационная компания)
— `Updated AF MOD contract - Jan 2018.exe`
— `remove_pw_Reschedule of CISD Regular Meeting.exe`
— `Sorchornor_with_PM_-_Sep_2017.exe`
— `20170905-Evaluation Table.xls.exe`
— `CV_LeHoangThing.doc.exe` (фейковые резюме также были обнаружены в Канаде)

У всех этих файлов есть нечто общее – запуск документа-приманки, защищенного паролем. Неясно, содержится ли где-то в данных передаваемого письма пароль, либо документ не должен открываться.

## Фейковые установщики

Несколько фейковых инсталлеров, выдающих себя за установщики или обновления ПО, были замечены в watering hole кампаниях. Один из примеров – переупакованный установщик Firefox, описанный 360 Labs на Freebuf (на китайском языке).

Другой образец, который мы видели, назывался `RobotofontUpdate.exe`. Он, вероятно, распространялся и через скомпрометированные сайты, но у нас нет достаточных доказательств этого.

Все описываемые файлы, распространялись ли они через почту или скачивались при посещении скомпрометированного сайта, доставляли один и тот же компонент бэкдора. В посте мы разберем образец `RobotofontUpdate.exe` и покажем, как ему удается выполнить вредоносную полезную нагрузку в системе.

# Технический анализ

Процесс установки и выполнения зависит от многослойной обфускации, а именно шифрования компонентов, реконструкции файлов PE, загрузки шелл-кода и техники side-loading (заливка). Последняя была описана в предыдущем исследовании ESET о Korplug.

## Обзор хода выполнения

Атака состоит из двух частей: дроппера и средства запуска. Каждый шаг каждой части процесса мы объясним детально в соответствующем разделе. Две схемы ниже дают краткое представление об общем ходе выполнения вредоносного ПО.

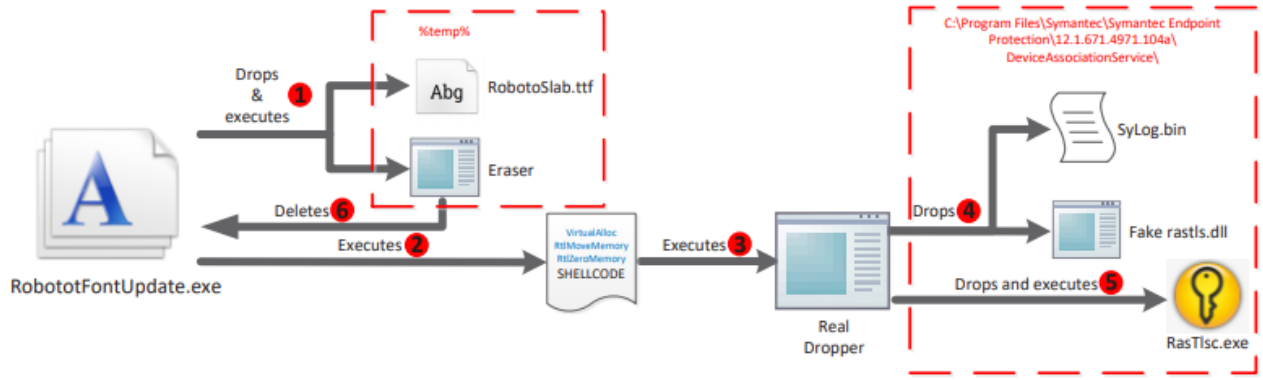



*Рисунок 1. Ход выполнения дроппера*

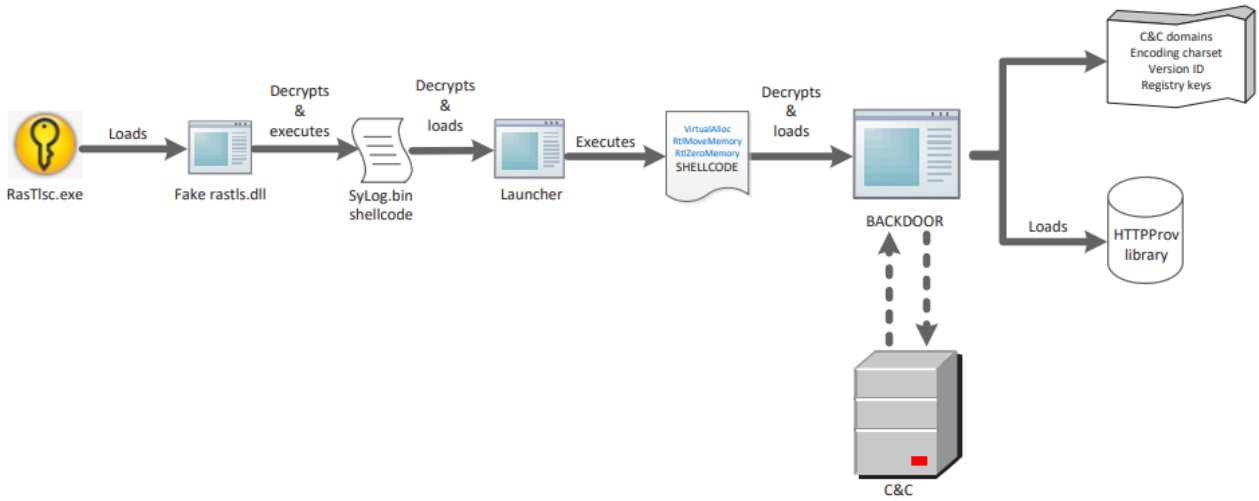
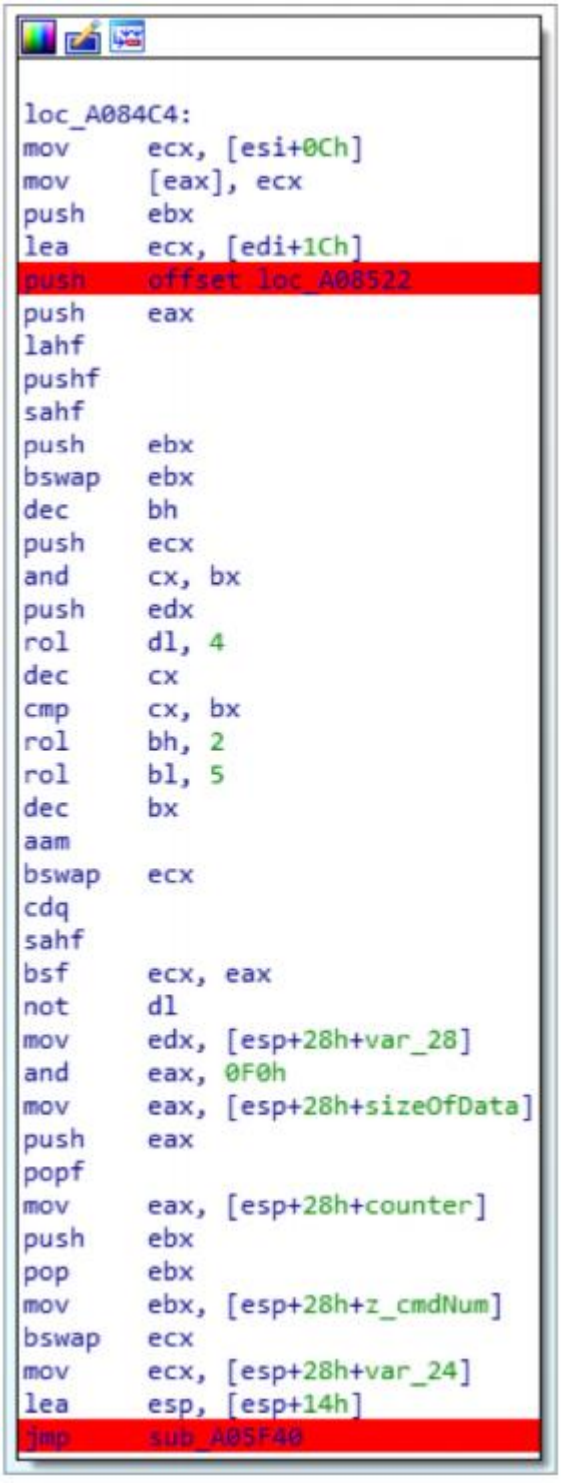



*Рисунок 2. Ход выполнения бэкдора*

Почти все эти компоненты обфусцированы. Обфускация основана на командах парного комплементарного условного перехода. По каждой из форм: JZ/JNZ, JP/JNP, JO/JNO и так далее, каждая пара выполняет переход к одной и той же цели. Последовательность перемежается с мусорным кодом, использующим указатель стека, но не меняющим значение условного флага. Получается, что переход совершается в пределах одной ветки. Это приводит к проблемам в процессе декомпиляции из-за применения положительных значений указателя стека.


*Рисунок 3. Комплементарный условный переход*

Кроме того, некоторые основные элементы программы добавляют один адрес в стек, после чего завершаются на JMP/CALL, в то время как другие базовые элементы добавляют два адреса и оканчиваются командой RET. Второе добавление элемента — это вызываемая функция, а первое — адрес следующего базового элемента программы, куда будет совершен переход. Таким образом создаются базовые элементы программы без родительских объектов.



```
loc_A084C4:
mov     ecx, [esi+0Ch]
mov     [eax], ecx
push    ebx
lea     ecx, [edi+1Ch]
push    offset loc_A08522
push    eax
lahf
pushf
sahf
push    ebx
bswap   ebx
dec     bh
push    ecx
and     cx, bx
push    edx
rol     dl, 4
dec     cx
cmp     cx, bx
rol     bh, 2
rol     bl, 5
dec     bx
aam
bswap   ecx
cdq
sahf
bsf     ecx, eax
not     dl
mov     edx, [esp+28h+var_28]
and     eax, 0F0h
mov     eax, [esp+28h+sizeOfData]
push    eax
popf
mov     eax, [esp+28h+counter]
push    ebx
pop     ebx
mov     ebx, [esp+28h+z_cmdNum]
bswap   ecx
mov     ecx, [esp+28h+var_24]
lea     esp, [esp+14h]
jmp     sub_A05F40
```

*Рисунок 4. Техника PUSH/JMP*

В результате сочетания двух техник обфускации получаются «красивые» графики:

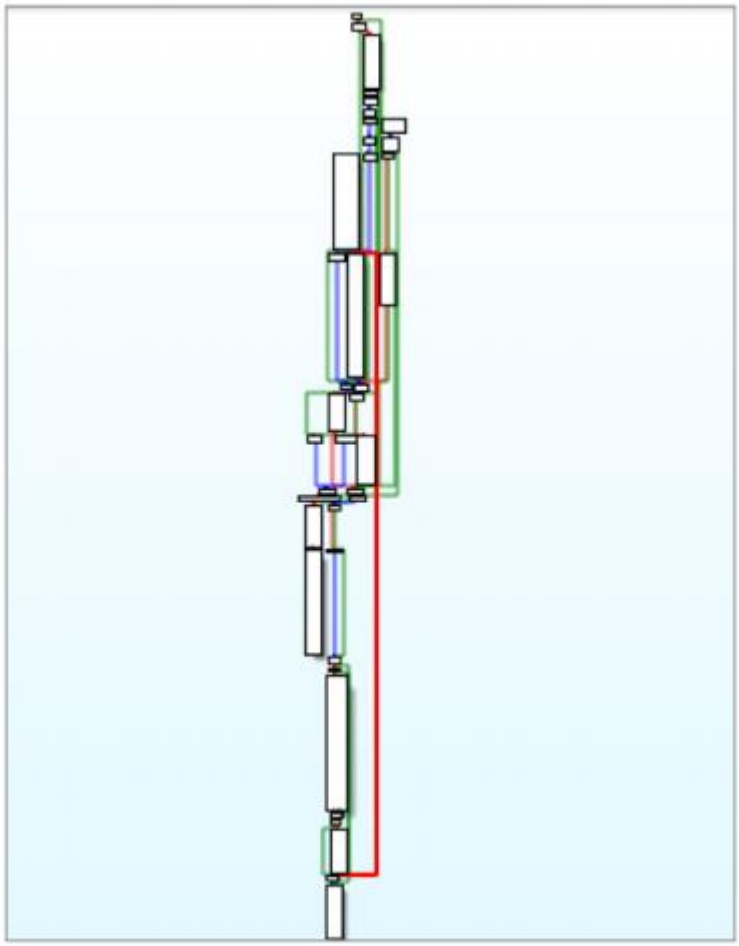


*Рисунок 5. Обфускация последовательности выполнения*

Заметить мусорный код довольно просто. Его можно игнорировать при анализе образцов, если знать схему применения.

# Дроппер

## Этап 1. Документ-приманка

В последние месяцы OceanLotus использовали несколько приманок. Один из них – фейковое ПО обновления шрифта `Roboto Slab regular` TrueType. Выбор шрифта кажется немного странным, так как он не поддерживает многие восточноазиатские языки.

*Рисунок 6. Иконка обновления шрифта RobototFontUpdate*

При исполнении бинарный файл расшифровывает свои ресурсы (XOR, 128 байт, жестко

закодированный ключ) и восстанавливает расшифрованные данные (LZMA). Легитимный файл RobotoSlab-Regular.ttf (SHA1:`912895e6bb9e05af3a1e58a1da417e992a71a324`) записывается в папку `%temp%` и запускается с помощью функции Win32 API `ShellExecute`.

Выполняется шелл-код, расшифрованный из ресурсов. После выполнения фейковое обновление шрифта внедряет другое приложение, единственной целью которого является удаление дроппера. Это «стирающее» приложение внедряется в виде `%temp%\[0-9].tmp.exe`.

## Этап 2. Шелл-код

На каждом этапе применяется один и тот же шелл-код.

Шелл-код – кастомный PE-загрузчик. Он восстанавливает исполняемый файл в памяти – расшифровывает все разделы и рассчитывает необходимые перемещения и иные отступы. Шелл-код применяет три функции Windows API: `VirtualAlloc`, `RtlMoveMemory` и `RtlZeroMemory`.

Функция `RtlZeroMemory` используется для обнуления полей в PE-заголовке. Полагаться на автоматический дамп памяти не получится, так как заголовки MZ/PE повреждены.
Шелл-код вызывает функцию входа в дешифрованный PE, а затем функцию экспорта `DLLEntry`.

## Этап 3. Настоящий дроппер

`{103004A5-829C-418E-ACE9-A7615D30E125}.dll`
Этот исполняемый файл расшифровывает ресурсы, используя алгоритм AES в режиме CBC через Windows API. Жестко закодированный ключ имеет размер 256 бит. После расшифровки сжатые данные распаковываются (алгоритм LZMA).

Если процесс запущен с правами администратора, вредоносное ПО обеспечивает персистентность, создавая службу. В ином случае используется классический ключ реестра Run (`HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run;DeviceAssociationService;rastlsc.exe`).

Если код дроппера исполняется с правами администратора, он пробует записать файлы, указанные ниже, в папку `C:\Program Files\Symantec\Symantec Endpoint Protection\12.1.671.4971.104a\DeviceAssociationService\`, если нет — пишет их в папку `%APPDATA%\Symantec\Symantec Endpoint Protection\12.1.671.4971.104a\DeviceAssociationService\`:

— `rastlsc.exe` (SHA1:`2616da1697f7c764ee7fb558887a6a3279861fac`, копия легитимного приложения Symantec Network Access Control, `dot1xtra.exe`)
— `SyLog.bin` (SHA1:`5689448b4b6260ec9c35f129df8b8f2622c66a45`, зашифрованный бэкдор)
— `rastls.dll` (SHA1:`82e579bd49d69845133c9aa8585f8bd26736437b`, вредоносный DLL, заливку которого выполняет `rastlsc.exe`)

Путь меняется от образца к образцу, но схема похожа. В зависимости от прав малварь сбрасывает файлы в `%ProgramFiles%` или `%appdata%`. Также мы наблюдали:

— `\Symantec\CNG Key Isolation\`
— `\Symantec\Connected User Experiences and Telemetry\`
— `\Symantec\DevQuery Background Discovery Broker Tasks\`

Эти пути используются различными продуктами Symantec.

После достижения персистентности и внедрения исполняемого файла легитимный файл, `rastlsc.exe`, исполняется с помощью `CreateProcessW`.

Также мы наблюдали версию (`{BB7BDEC9-B59D-492E-A4AF-4C7B1C9E646B}.dll`), которая исполняет `rastlsc.exe` с параметром `krv`. Детально обсудим далее.

## Бэкдор-компонент: заливка rastlsc.exe

Группа OceanLotus использует старую и широко известную технику в одном из исполняемых файлов продукта Symantec. Суть в том, чтобы использовать процесс загрузки библиотеки легитимного и подписанного файла .exe, путем записи вредоносной библиотеки в ту же папку. Это заставит вредоносное поведение выглядеть легитимным, поскольку эти действия выполняются в процессе работы доверенного исполняемого файла.

Как говорилось выше, сбрасывается и выполняется легитимный файл `rastlsc.exe`.
Он импортирует файл `rastls.dll`, который в этом случае имеет вредоносное содержимое.

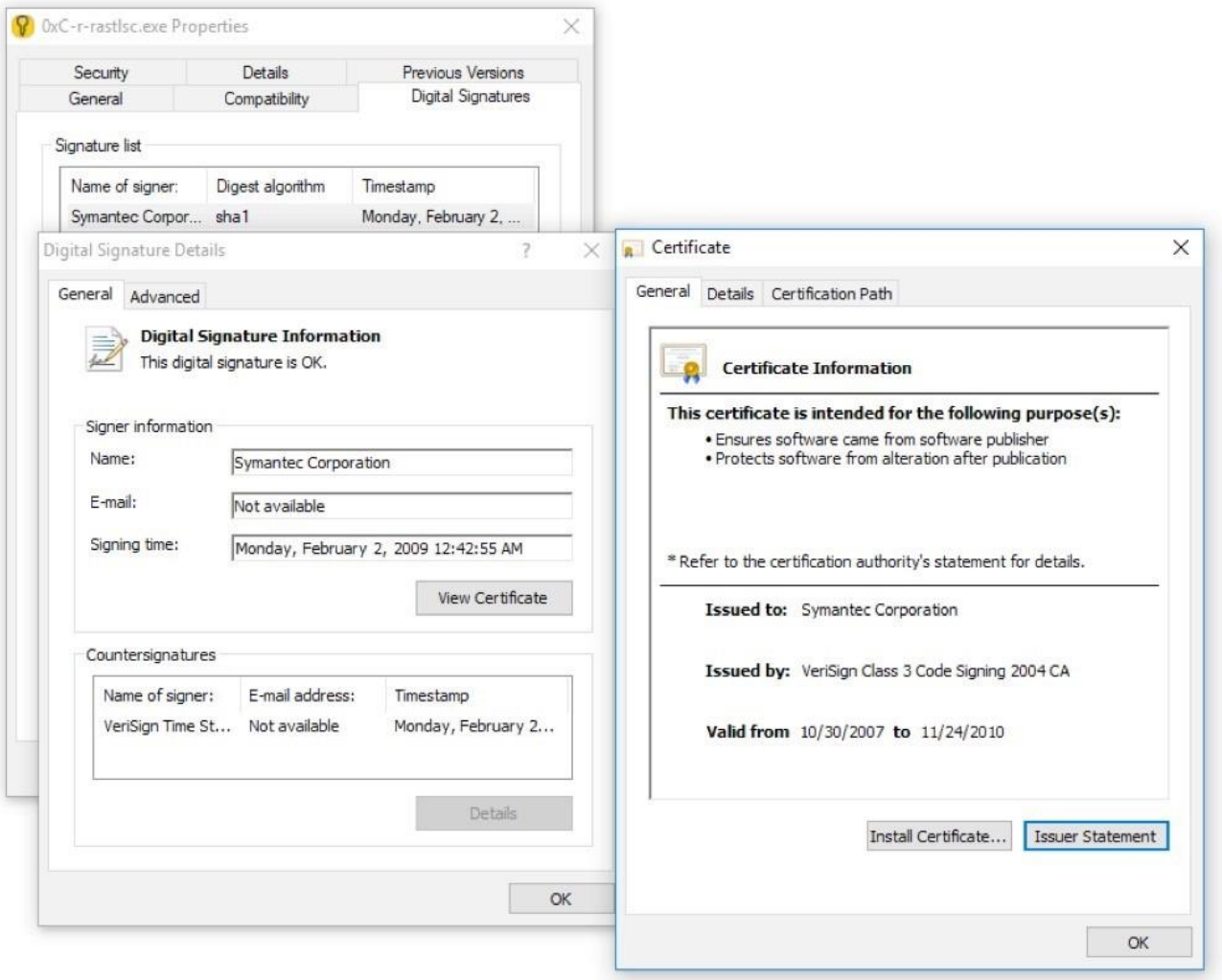



*Рисунок 7. Цифровая подпись rastlsc.exe от Symantec*

Также мы видели заливку с использованием других легитимных и подписанных исполняемых

файлов, в том числе `mcoemcpy.exe` от McAfee, который подгружает `McUtil.dll`. Данная техника ранее применялась PlugX, что привлекло внимание Vietnam CERT (на вьетнамском языке).

## Этап 1. Заливка библиотеки, rastls.dll

Внутреннее имя файла dll — `{7032F494-0562-4422-9C39-14230E095C52}.dll`, но мы видели и другие версии, например, `{5248F13C-85F0-42DF-860D-1723EEAA4F90}.dll`. Все экспортируемые функции приводят к выполнению одной и той же функции.

```
; Exported entry   1. RasEapCreateConnectionProperties2
; Exported entry   2. RasEapCreateConnectionProperties
; Exported entry   3. RasEapCreateUserProperties2
; Exported entry   4. RasEapCreateUserProperties
; Exported entry   7. RasEapGetIdentity
; Exported entry   9. RasEapInvokeConfigUI
; Exported entry  11. RasEapQueryCredentialInputFields
; Exported entry  12. RasEapQueryInteractiveUIInputFields
; Exported entry  13. RasEapQueryUIBlobFromInteractiveUIInputFields
; Exported entry  14. RasEapQueryUserBlobFromCredentialInputFields
; Exported entry  15. RasEapUpdateServerConfig


; Attributes: thunk

public RasEapUpdateServerConfig
RasEapUpdateServerConfig proc near
jmp     RasEapUpdateServerConfig_0 ; RasEapCreateConnectionProperties2
RasEapUpdateServerConfig endp ; RasEapCreateConnectionProperties
                        ; RasEapCreateUserProperties2
                        ; RasEapCreateUserProperties
                        ; RasEapGetIdentity
                        ; RasEapInvokeConfigUI
                        ; RasEapQueryCredentialInputFields
                        ; RasEapQueryInteractiveUIInputFields
                        ; RasEapQueryUIBlobFromInteractiveUIInputFields
                        ; RasEapQueryUserBlobFromCredentialInputFields
```

*Рисунок 8. Все экспорты rasltls.dll ведут к выполнению одной функции*

Экспорт пробует прочитать файл `SyLog.bin`, расположенный в той же папке. Другие версии пытались открыть файл `OUTLFLTR.DAT`. Если файл существует, выполнится его расшифровка по алгоритму AES в режиме CBC с жестко закодированным 256-битным ключом, а затем полученные сжатые данные распаковываются (сжатие LZMA).

Вариант `McUtil.dll` использует другую технику. На первый взгляд, основная функция не выполняет ничего вредоносного, но по факту она заменяет раздел `.text` легитимного файла `mcoemcpy.exe`, бинарного файла. Он генерирует шелл-код, задачей которого является вызов функции для чтения зашифрованного шелл-кода второго этапа из файла `mcscentr.adf`.

Для создания шелл-кода применяется следующий псевдокод:

```
x = False i = 0
buff = genRandom()
opc1 = [0x58,0x59,0x5a,0x5b]
opc2 = [0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57]
opc3 = [0x90,0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,
0x49,0x4a,0x4b]
while i < len(buff):
currentChar = buff[i] if currentChar < 0xc8:
buff[i] = opc1[currentChar % len(opc1)]
else:
if x:
buff[i] = opc2[currentChar % len(opc2)]
else:
buff[i] = opc3[currentChar % len(opc3)] x = x == False
i+=1
```

Ниже на рисунке можно увидеть листинг результата работы ассемблера:

```
90              nop
4A              dec edx
43              inc ebx
4A              dec edx
90              nop
48              dec eax
48              dec eax
48              dec eax
4B              dec ebx
41              inc ecx
49              dec ecx
90              nop
40              inc eax
90              nop
4B              dec ebx
4B              dec ebx
4A              dec edx
49              dec ecx
53              push ebx
41              inc ecx
B8 D0 1E B6 73  mov eax,0xb-r-mcutil.73B61ED0
FF D0           call eax
C3              ret
```

*Рисунок 9. Генерируемый шелл-код*

## Этапы 2–4. Шелл-код, средство запуска и снова шелл-код

Шелл-код расшифровывает и загружает библиотеку `{E1E4CBED-5690-4749-819D-24FB660DF55F}.dll`. Библиотека загружает ресурсы и пробует запуск службы DeviceAssociationService. Дешифрованная информация также содержит шелл-код. Последний расшифровывает финальную стадию: бэкдор.

Вариант `{92BA1818-0119-4F79-874E-E3BF79C355B8}.dll` проверяет, был ли `rastlsc.exe` исполнен с `krv` в качестве первого параметра. Если да — создается задача, и `rastlsc.exe` исполняется еще раз, но без этого параметра.

## Этап 5. Бэкдор

`{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll`

Сначала вредоносная программа пытается загрузить свои ресурсы и расшифровать их по алгоритму RC4. Полученные ресурсы содержат данные, используемые для конфигурации бэкдора. Формат конфигурации узнать нетрудно. С помощью Kaitai struct и его дампера структуры мы получаем следующее:

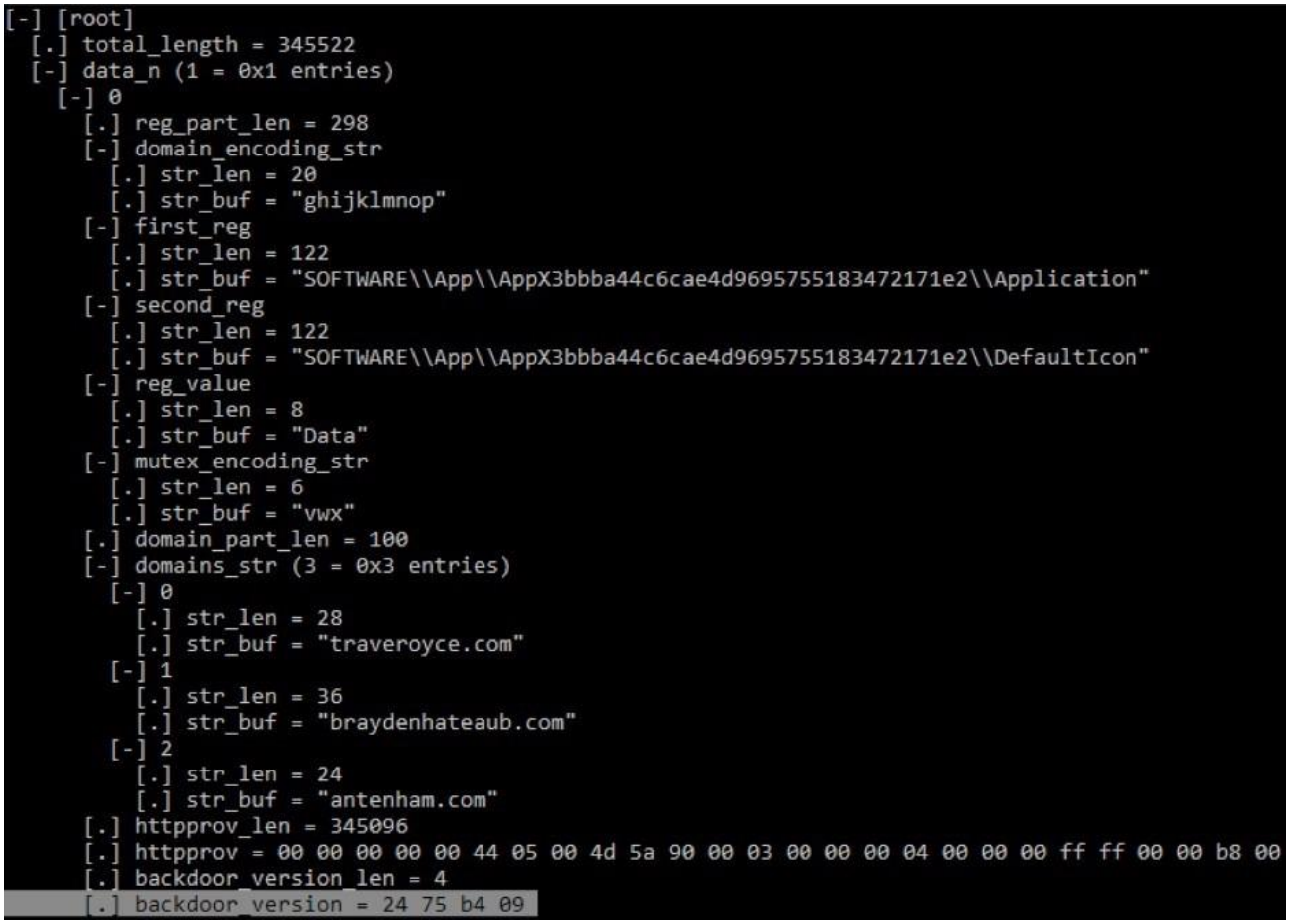



*Рисунок 10. Структура конфигурации*

**Примечание**: за исключением строки domain_encoding_str и библиотеки httpprov, данные меняются от образца к образцу. Ключи реестра почти одинаковы, но схема у них схожая: `\HKCU\SOFTWARE\Classes\AppX[a-f0-9]{32}`, ничего примечательного.

Вредоносная программа получает первые 10 байтов имени пользователя (UTF-16), ксорит их с трехбуквенной строкой `mutex_encoding_str` в UTF-16 и кодирует в шестнадцатеричной системе. Результат используется в качестве имени мьютекса. К примеру, для пользователя, чье имя начинается с `abc` и ключа в виде `vwx`, мьютексом будет `\Sessions\1\BaseNamedObjects\170015001b`.

Бэкдор включает PE-загрузчик, который загружает библиотеку `HTTPProv.dll` в память, вызывает точку входа и затем вызывает функцию экспорта `CreateInstance`.

## Коммуникация

Бэкдор использует стандартный коммуникационный протокол TCP через порт `25123`. Для получения IP-адреса сервера бэкдор сначала создает определенный DNS-запрос.

Вредоносная программа выбирает один из трех доменов из конфигурации и добавляет специальный поддомен, который генерируется с помощью двух значений. Первое значение — имя компьютера в пределах длины в 16 байтов. Второе — четырехбайтовый ID версии. Код на Python 2 ниже представляет собой алгоритм кодирования:

```
letters=domain_encoding_str # “ghijklmnop” hex_pc_name=pc_name.encode(“UTF-16LE”).encode(“hex”) s=’’
for c in hex_pc_name:
if 0x2f < ord(c) < 0x3a:
s+=letters[ord(c) - 0x30]
else:
s+=c
```

К примеру, если имя компьютера random-pc, а ID версии — 0x0a841523, то генерируется следующий домен:
`niggmhggmeggmkggmfggmdggidggngggmjgg.ijhlokga.dwarduong[.]com`

Для метки сервера C&C этого бэкдора можно использовать следующее регулярное выражение:
`[ghijklmnopabcdef]{4-60}\.[ghijklmnopabcdef]{8}\.[a-z]+\.[a-z]+`

Если адрес IP принадлежит конкретному домену, малварь пробует установить соединение по протоколу TCP через порт `25123`. У каждого из образцов есть три различных имени домена, которые используются для поиска C&C сервера.

Процесс коммуникации зашифрован по RC4 и сжат с помощью LZMA. Есть возможность расшифровать трафик, так как ключ добавлен к началу пакетов. Формат следующий:
`[ключ RC4 (4 байтов)][зашифрованная информация]`

Каждый байт ключа генерируется функцией `rand`. После расшифровки и распаковки пакета данные имеют следующий формат:
`[dw:неизвестно][dw:неизвестно][dw:номер команды][dw:размер данных][dw:неизвестно] [dw:данные]`

При первом подключении клиента к серверу происходит передача идентификатора UUID, который используется в качестве идентификатора сеанса. Последний хранится в разделе реестра в виде бинарных данных: `HKCU\SOFTWARE\Classes\AppXc52346ec40fb4061ad96be0e6cb7d16a\DefaultIcon`

Как мы говорили ранее, бэкдор также содержит библиотеку под названием `HTTPprov`. Она используется в качестве альтернативного способа связи с сервером. Файл DLL отправляет POST-запрос по протоколу HTTP. Он также поддерживает HTTPS и использование прокси SOCKS5, SOCKS4a и SOCKS4. Библиотека статически скомпонована с `libcurl`.

После инициализации в реестре будет создана запись – команда для бэкдора в дальнейшем использовать HTTP для связи с командным сервером: `HKCU\SOFTWARE\Classes\CLSID{E3517E26-8E93-458D-A6DF-8030BC80528B}`.

Используется стандартное клиентское приложение: `Mozilla/4.0 (с поддержкой; MSIE 8.0; Windows NT 6.0; Trident/4.0)`.

Основная характеристика этой библиотеки — специальный алгоритм шифрования универсального идентификатора ресурса. Ресурсная часть URI создается с помощью следующего псевдокода:

```
buffEnd = ((DWORD)genRand(4) % 20) + 10 + buff; while (buff < buffEnd){
b=genRand(16);
if (b[0] - 0x50 > 0x50)
t=0;
else
*buf++= UPPER(vowels[b[1] % 5]);
v=consonants[b[1]%21]); if (!t)
v=UPPER(v);
*buff++= v;
if (v!='h' && b[2] - 0x50 < 0x50)
*buff++= 'h';
*buff++= vowels[b[4] % 5];
if (b[5] < 0x60)
*buff++= vowels[b[6] % 5];
*buff++= consonants[b[7] % 21];
if (b[8] < 0x50)
*buff++= vowels[b[9] % 5];
*buff++= '-';
};
*buff='\0';
```

**Примечание**: для наглядности из кода убрана часть, отвечающая за проверку длины строки.

Для получения идентификатора из сгенерированной строки складываются два числа с помощью специального алгоритма проверки суммы:

```
checksum=crc32(buff)
num2=(checksum >> 16) + (checksum & 0xffff) * 2
num1=(num2 ^ 1) & 0xf
URL=GENERATED_DOMAIN+ "/" + num1 + "/" + num2 + "-" + buff
```

Добавив генератор URI библиотеки `HTTPprov` получаем следующий адрес URL:
`hXXp://niggmhggmeggmkggmfggmdggidggngggmjgg.ijhlokga.aisicoin[.]com/13/139756-Ses-Ufali-L`

## Команды

После получения идентификатора сеанса SESSIONID бэкдор делает цифровой отпечаток системы. Пакет построен следующим образом (отступ в пакете — описание):

*0x000* — байт: значение изменяется в каждой версии
*0x001* — 0x01: жестко закодированный байт
*0x002* — bool: наличие повышенных привилегий
*0x003* — dword: ID версии
*0x007* — строка (UTF-16), имя компьютера (макс. 0x20)
*0x027* — строка (UTF-16), имя пользователя
*0x079* — результат запроса по реестру в `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion`значения: `ProductName`, `CSDVersion`, `CurrentVersion`, `ReleaseId`, `CurrentBuildNumber` и результат вызова `IsWow64Process (x86|x64)`
*0x179* — последующий формат строки %s(%s); заменяется на (`GetVolumeInformationW:VolumeNameBuffer`), `VolumePathNames`
*0x279* — Физический диск, контроль «ввод-вывод» устройства PhysicalDrive deviceIOControl 0x2D1400 (IOCTL_STORAGE_QUERY_ PROPERTY) (VolSerialNumber)

*0x379* — wmi `SELECT SerialNumber FROM Win32_BaseBoard`
*0x3f9* — Получение текущей даты и времени GetSystemTimeAsFileTime
*0x400* — bool: неизвестно
*0x401* — dword: получено после дешифрования ресурса

Вот пример цифрового отпечатка системы:

```
Fingerprint
=======================
0x0:03
0x1:01
0x2:    isAdmin ?: 01
0x3:    Version: 0xa841523
0x7:    ComputerName: MOISEPC
0x27:   Username: Administrator
0x79:   RegistryQueries: Windows 7 Enterprise Service Pack 1 6.1.7601 x86
0x179:  Volumes: ( C : ) ;   ( D : ) ;

0x279:  VolumeSerialNumber: [redacted]

0x379:  BaseBoard Serial Number: [redacted]
0x3f9:  SystemTimeAsFileTime: 0x1BF545D479FA600
```

*Рисунок 11. Цифровой отпечаток системы*

Это полноценный бэкдор, который предоставляет своим операторам ряд возможностей: манипуляции с файлом, реестром и процессами, загрузка дополнительных компонентов, получение цифрового отпечатка системы. Ниже номера и описания поддерживаемых команд:

0 — цифровой отпечаток
1 — устанавливает ID сеанса
2 — создание процесса и получение результата (используя программные каналы)
3 — устанавливает счетчик попыток соединения
4 — откладывает время поллинга
5 — считывает файл или ключ реестра и считает MD5
6 — создание процесса
7 — создает файл, запись в реестре или поток в памяти
8 — пишет в реестр
9 — опрашивает реестр
10 — ищет файлы в системе
11 — переносит файлы в другую директорию
12 — удаляет файлы с диска
13 — получение списка дисков, размеченных в системе при помощи функции `GetLogicalDriveStringW`
14 — создает директорию
15 — удаляет директорию
16 — читает файл от смещения
17 — вызывает PE-загрузчик (переход к коммуникации по `HTTPprov`)
18 — [неизвестно]
19 — 0: опрос значения в реестре; 1: внедрение и исполнение программы
20 — устанавливает переменную среды
21 — запускает шелл-код в новом потоке
22 — возвращает переменную среды
+23 в новой версии — перезапускает себя, если переменная среды APPL не существует

# Заключение

OceanLotus сохраняет высокую активность и продолжает обновлять инструментарий.
Группа стремится скрывать свою деятельность, для этого атакующие тщательно выбирают жертв, ограничивают распространение вредоносных программ, используют несколько серверов, чтобы не привлекать внимание к одному домену или IP-адресу. Расшифровка внедряемого компонента и техника заливки (side-loading), несмотря на широкую известность, позволяет избежать обнаружения, поскольку работа злоумышленников в этом случае маскируется под легитимное приложение.

# Индикаторы компрометации (IoCs)

## Образцы

### Таблица 1: дроппер

| SHA1-1 | Имя файла | Обнаружение продуктами ESET |
|---|---|---|
| fdcb35cd9cb8dc1474cbcdf1c9bb03200dcf3f18 | RobototFontUpdate.exe | Win32/TrojanDropper.Agent.RUI |
| a40ee8ff313e59aa92d48592c494a4c3d81449af | Firefox Installer.exe | Win32/TrojanDropper.Agent.RUI |
| c2eb1033bc01ab0fd732a7ba4967be02c0690bf0 | 20170905-Evaluation Table.xls.exe | Win32/TrojanDropper.Agent.RUI |
| d35695f2366a43628231e73ffa83ca106306a8fa | CV_LeHoangThing.doc.exe | Win32/TrojanDropper.Agent.RUI |
| fe0161fb8a26a0bf4afad746c7ebf89499dcd3a7 | Chi tiet don khieu nai gui saigontel.exe | Win32/TrojanDropper.Agent.RUI |
| 032ef58b7978d079287874044dc516af624ae5f5 | Mi17 Technical issues - Phonesack Grp.exe | Win32/TrojanDropper.Agent.RUI |
| 2a387d7d47a63d6e47d9cc92d3dc69a53816c2c0 | Sorchornor_with_PM_-_Sep_2017.exe | Win32/TrojanDropper.Agent.RUI |
| 7105caa6d4fd8a2c67523d385277528e556ae4f6 | Updated AF MOD contract - Jan 2018.exe | Win32/TrojanDropper.Agent.RUI |
| f96bcd875836da89800912de1e557891697c7cf4 | remove_pw_Reschedule of CISD Regular Meeting.ex_ | Win32/TrojanDropper.Agent.RUI |

### Таблица 2: библиотеки

| SHA1-1 | Имя файла | Обнаружение продуктами ESET |
|---|---|---|
| 82e579bd49d69845133c9aa8585f8bd26736437b | rastls.dll | Win32/Salgorea.BD |
| 202fb56edb2fb542e05c845d62ffbdcfbebed9ec | McUtil.dll | Win32/Korplug.MK |

## Сеть

### IP-адреса

```
46.183.220.81
46.183.220.82
46.183.222.82
46.183.222.83
46.183.222.84
46.183.223.106
46.183.223.107
74.121.190.130
74.121.190.150
79.143.87.230
```