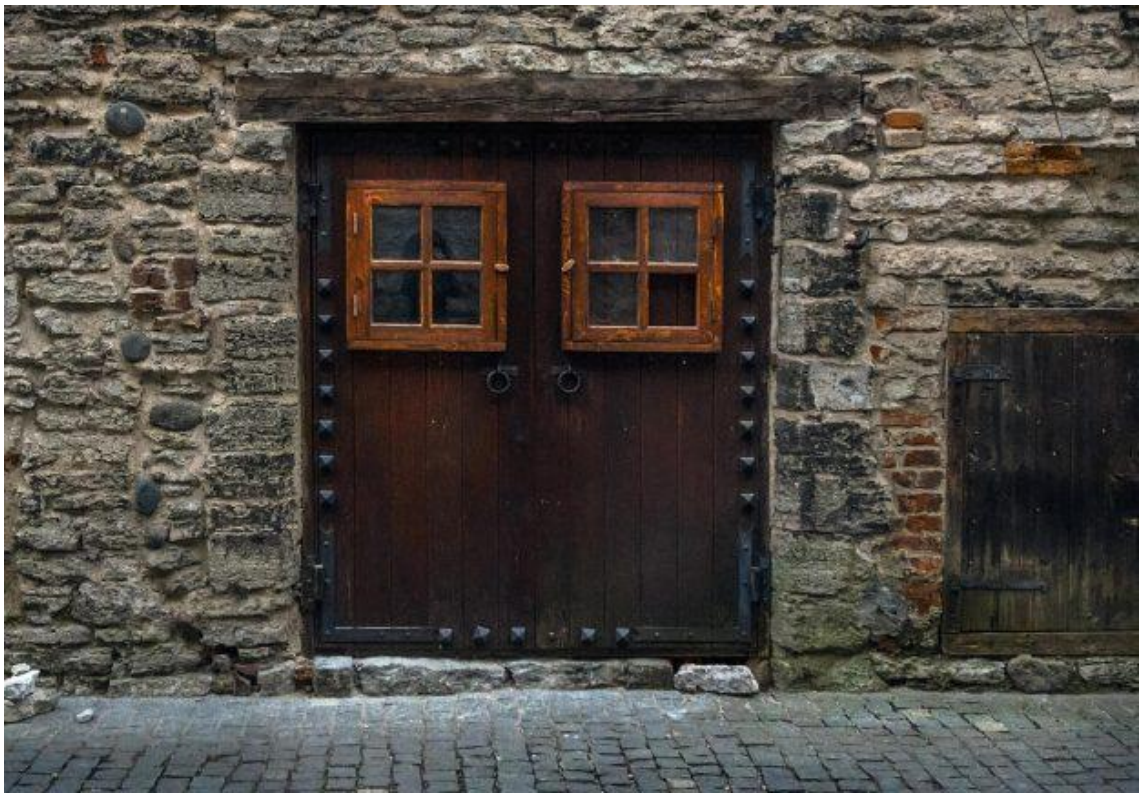
АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

OceanLotus: обновление малвари для macOS

10 апреля 2019 года

В марте 2019 года в VirusTotal, популярный онлайн-сервис сканирования, был загружен новый образец вредоносного ПО для macOS кибергруппы OceanLotus. Исполняемый файл бэкдора обладает теми же возможностями, что и предыдущая изученная нами версия малвари для macOS, но его структура изменилась и его стало сложнее обнаружить. К сожалению, мы не смогли найти дроппер, связанный с этим образцом, поэтому пока не знаем вектор заражения.

Недавно мы опубликовали пост про OceanLotus и о том, как операторы пытаются обеспечить персистентность, ускорить выполнение кода и свести к минимуму следы присутствия в системах Windows. Известно также, что у этой кибергруппы есть и компонент для macOS. В данном посте подробно описываются изменения в новейшей версии малвари для macOS в сравнении с предыдущим вариантом (описанным Trend Micro), а также рассказывается, как при анализе можно автоматизировать расшифровку строк с помощью IDA Hex-Rays API.



Анализ

В следующих трех частях описывается анализ образца с хэшем SHA-1 `E615632C9998E4D3E5ACD8851864ED09B02C77D2`. Файл называется **flashlightd**, антивирусные продукты ESET детектируют его как OSX/OceanLotus.D.

Антиотладка и защита от песочниц

Как все macOS-бинарники OceanLotus, образец упакован с UPX, но большинство средств идентификации упаковщиков не распознают его как таковой. Вероятно, потому, что они в основном содержат подпись, зависящую от наличия строки “UPX”, кроме того, Mach-O сигнатуры встречаются реже и не так часто обновляются. Эта особенность затрудняет статическое обнаружение. Интересно, что после распаковки точка входа находится в начале раздела `__cfstring` в сегменте `.TEXT`. В этом разделе есть атрибуты flag, как показано на рисунке ниже.

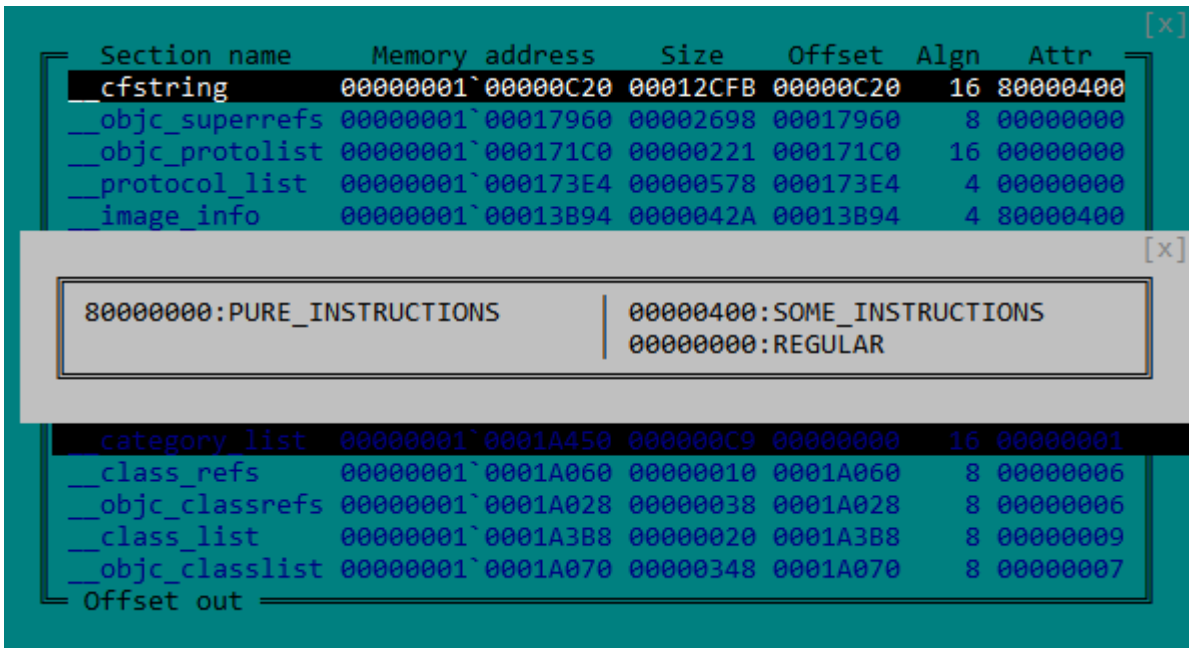


Рисунок 1. Атрибуты раздела MACH-O __cfstring

Как показано на рисунке 2, расположения кода в разделе `__cfstring` позволяет обмануть некоторые инструменты дизассемблирования, отображая код в виде строк.

```
__cfstring:0000000100000C20 start           __CFString <0E48348E58948006Ah, 758D48087D8B48F0h, 0E2C101C283FA8910h,\
__cfstring:0000000100000C20                             0EBD18948F2014803h>
__cfstring:0000000100000C40                 __CFString <39834808C1834804h, 0E808C18348F67500h, 0FFE8C7890000D626h,\
__cfstring:0000000100000C40                             0E5894855F400012Dh>
__cfstring:0000000100000C60                 __CFString <5441554156415741h, 0FD8949F689495053h, 0C4894900012EC9E8h,\
__cfstring:0000000100000C60                             0F8D1E001441FE8C1h>
```

Рисунок 2. Код бэкдора определяется IDA как данные

После запуска бинарный файл создает поток в качестве средства защиты от отладки, единственной целью которого является постоянная проверка наличия дебаггера. Для этого поток:

- Пытается отцепить любой дебаггер, вызывая `ptrace` с `PT_DENY_ATTACH` в качестве параметра запроса
- Проверяет, открыты ли некоторые исключительные порты, вызывая функцию `task_get_exception_ports`
- Проверяет, подключен ли дебаггер, как показано на рисунке ниже, путем проверки наличия флага `P_TRACED` в текущем процессе

```
LABEL_6:
    info.kp_proc.p_flag = 0;
    mib[0] = CTL_KERN;
    mib[1] = KERN_PROC;
    mib[2] = KERN_PROC_PID;
    mib[3] = getpid();
    size = 0x288LL;
    sysctl(mib, 4u, &info, &size, 0LL, 0LL);
    v1 = (unsigned __int16)(info.kp_proc.p_flag & P_TRACED) >> 11;
```

Рисунок 3. Проверка подключения дебаггера посредством функции sysctl

Если сторожевая схема обнаруживает присутствие дебаггера, вызывается функция `exit`. Кроме того, затем образец проверяет среду, выполняя две команды:

```
ioreg -l | grep -e "Manufacturer" и sysctl hw.model
```

После этого образец проверяет возвращаемое значение по жестко закодированному списку строк известных систем виртуализации: **acle**, **vmware**, **virtualbox** или **parallels**. Наконец, следующая команда проверяет, является ли машина одной из следующих “MBP”, “MBA”, “MB”, “MM”, “IM”, “MP” and “XS”. Это коды модели системы, например, “MBP” означает MacBook Pro, “MBA” – MacBook Air и т. д.

```
system_profiler SPHardwareDataType 2>/dev/null | awk '/Boot ROM Version/
{split($0, line, ":");printf("%s", line[2]);}
```

Основные дополнения

Несмотря на то, что команды бэкдора не изменились со времен исследования Trend Micro, мы заметили несколько других модификаций. C&C-серверы, используемые в этом образце, довольно новые, дата их создания – 22.10.2018.

- **daff.faybilodeau[.]com**
- **sarc.onteagleroad[.]com**
- **au.charlineopkesston[.]com**

Ресурс URL изменился на `/dp/B074WC4NHW/ref=gbps_img_m-9_62c3_750e6b35`.
Первый пакет, отправляемый на C&C-сервер, содержит больше информации о хост-машине, включая все данные, собираемые командами из таблицы ниже.

Команды	Описание
`system_profiler SPHardwareDataType 2>/dev/null | awk '/Processor /{split($0,line,":");printf("%s",line[2]);}' machdep.cpu.brand_string`	Информация о процессоре
`system_profiler SPHardwareDataType 2>/dev/null | awk '/Memory/{split($0,line, ":"); printf("%s", line[2]);}'`	Информация о памяти
`ifconfig -l`	Сетевой интерфейс MAC
`ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ {split($0, line, "\""); printf("%s", line[4]); }'`	Запрос серийного номера устройства

Помимо этого, изменения конфигурации, образец использует для сетевой фильтрации не библиотеку libcurl, а внешнюю библиотеку. Чтобы найти ее, бэкдор пытается расшифровать каждый файл в текущем каталоге, используя AES-256-CBC с ключом `gFjMXBgyXWULmVVVzyxy`, дополненным нулями. Каждый файл расшифровывается и сохраняется как /tmp/store, а попытка его загрузить как библиотеку сделана с использованием функции dlopen. Когда попытка расшифровки приводит к успешному вызову `dlopen`, бэкдор извлекает экспортированные функции `Boriry` и `ChadylonV`, которые, по всей видимости, отвечают за сетевое взаимодействие с сервером. У нас нет дроппера или других файлов из исходного местоположения образца, поэтому мы не можем проанализировать эту библиотеку. Более того, поскольку компонент зашифрован, YARA-правило, основанное на этих строках, не будет соответствовать файлу, найденному на диске.

Как описано в вышеупомянутой статье, создается *cliendID*. Этот идентификатор является хешем MD5 возвращаемого значения одной из следующих команд:

— `ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ { split($0, line, "\""); printf("%s", line[4]); }'`
— `ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ { split($0, line, "\""); printf("%s", line[4]); }'`
— `ifconfig en0 | awk \'/ether /{print $2}\'` (получить MAC адрес)
— неизвестная команда ("`\x1e\x72\x0a`"), которая используется в предыдущих образцах

Перед хешированием к возвращаемому значению добавляется символ «0» или «1», указывающий на наличие root привилегий. Этот *clientID* хранится в `/Library/Storage/File System/HFS/25cf5d02-e50b-4288-870a-528d56c3cf6e/pivtoken.appex`, если код запущен от root или в ~/Library/SmartCardsServices/Technology/PlugIns/drivers/snippets.ecgML во всех остальных случаях. Файл обычно скрыт с помощью функции _chflags, его временная метка изменяется с помощью команды `touch -t` со случайным значением.

Расшифровка строк

Как и в предыдущих вариантах, строки зашифрованы с использованием AES-256-CBC (шестнадцатеричный ключ: `9D7274AD7BCEF0DED29BDBB428C251DF8B350B92` дополнен нулями, а IV заполнен нулями) посредством функции CCCrypt. Ключ изменен в сравнении с предыдущими версиями, но, поскольку группа все еще использует тот же алгоритм шифрования строк, расшифровка может быть автоматизирована. Помимо этого поста мы выпускаем скрипт IDA, использующий API Hex-Rays для расшифровки строк, присутствующих в бинарном файле. Этот скрипт может помочь в будущем анализе OceanLotus и анализе существующих образцов, которые мы пока не смогли получить. В основе сценария – универсальный метод получения аргументов, переданных функции. Кроме того, он ищет назначения параметров. Метод можно использовать повторно, чтобы получить список аргументов функции и затем передать на обратный вызов (callback).

Зная прототип функции *decrypt*, скрипт находит все перекрестные ссылки на эту функцию, все аргументы, затем расшифровывает данные и помещает простой текст внутри комментария по адресу перекрестной ссылки. Чтобы скрипт работал правильно, в нем должен быть установлен пользовательский алфавит, используемый функцией декодирования base64, и должна быть определена глобальная переменная, содержащая длину ключа (в данном случае DWORD, см. рисунок 4).

```
000000010001A408 00 00 00 00 00 00 00 00                  dq 0
000000010001A410 9D 72 74 AD 7B CE F0 DE D2 9B+key        db 9Dh, 72h, 74h, 0ADh, 7Bh, 0CEh, 0F0h, 0DEh, 0D2h, 9Bh
000000010001A410 DB B4 28 C2 51 DF 8B 35 0B 92                                    ; DATA XREF: f_MachineCheck+49↑o
000000010001A410                                                                  ; f_MachineCheck+424↑o ...
000000010001A410                                          db 0DBh, 0B4h, 28h, 0C2h, 51h, 0DFh, 8Bh, 35h, 0Bh, 92h
000000010001A424                          ; unsigned int key_len
000000010001A424 14 00 00 00              key_len         dd 14h                  ; DATA XREF: f_MachineCheck+3E↑o
000000010001A424                                                                  ; f_CheckMachineType+4B↑o ...
000000010001A428 00 00 00 00 00 00 00 00                  dq 0
```

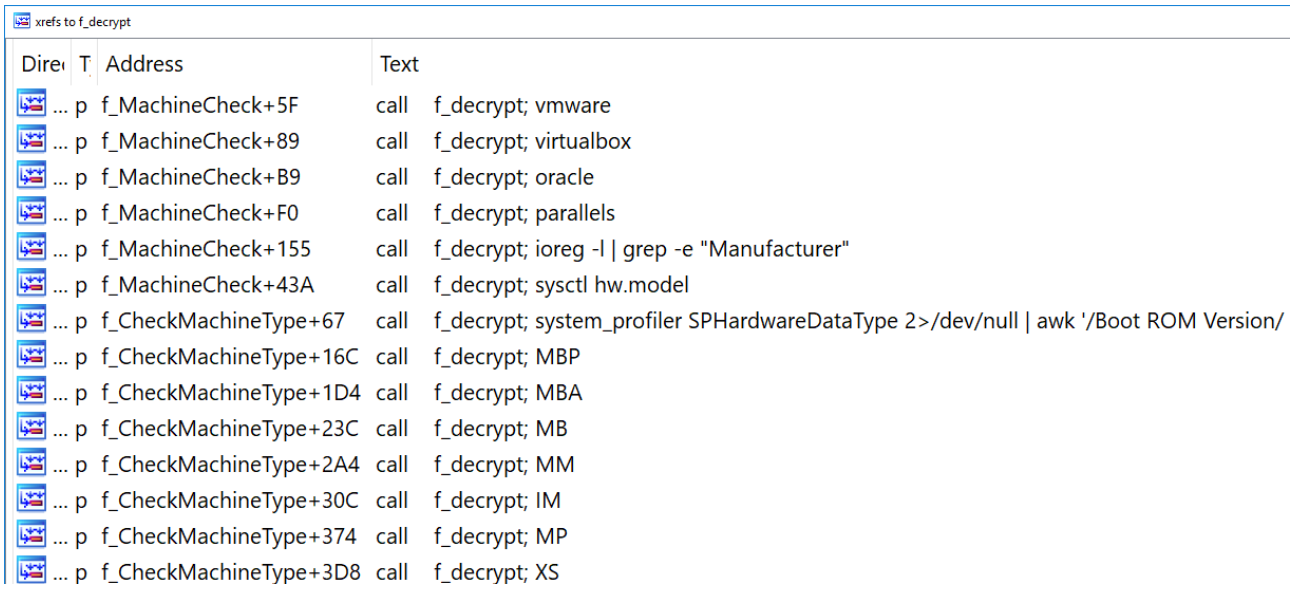
Рисунок 4. Определение глобальной переменной key_len

В окне Function можно вызвать правой кнопкой мыши функцию расшифровки и кликнуть «Извлечь и расшифровать аргументы». Сценарий должен помещать расшифрованные строки в комментарии, как показано на рисунке 5.

```
00000001000012C3 movaps  xmm0, xmmword ptr cs:a_vmware ; "Y:zC^"
00000001000012CA movaps  [rbp+var_50], xmm0
00000001000012CE mov     [rbp+var_40], 0
00000001000012D2 lea     r12, key_len
00000001000012D9 mov     ecx, [r12]
00000001000012DD lea     rbx, key
00000001000012E4 lea     rdi, [rbp+var_50]
00000001000012E8 mov     esi, 10h
00000001000012ED xor     r8d, r8d
00000001000012F0 mov     rdx, rbx
00000001000012F3 call    f_decrypt       ; vmware
00000001000012F8 mov     r13, rax
00000001000012FB movaps  xmm0, xmmword ptr cs:a_virtualbox ; "\x15\x7F"
0000000100001302 movaps  [rbp+var_70], xmm0
0000000100001306 mov     [rbp+var_60], 0
000000010000130A mov     ecx, [r12]
000000010000130E lea     rdi, [rbp+var_70]
0000000100001312 mov     esi, 10h
0000000100001317 xor     r8d, r8d
000000010000131A mov     rdx, rbx
000000010000131D call    f_decrypt       ; virtualbox
0000000100001322 mov     r15, rax
0000000100001325 movaps  xmm0, xmmword ptr cs:a_oracle ; "-,l\x05A"
000000010000132C movaps  [rbp+var_90], xmm0
0000000100001333 mov     [rbp+var_80], 0
0000000100001337 mov     ecx, [r12]
000000010000133B lea     rdi, [rbp+var_90]
0000000100001342 mov     esi, 10h
0000000100001347 xor     r8d, r8d
000000010000134A mov     rdx, rbx
000000010000134D call    f_decrypt       ; oracle
0000000100001352 mov     [rbp+var_170], rax
0000000100001359 movaps  xmm0, xmmword ptr cs:unk_1000171C0
```

Рисунок 5. Расшифрованный текст помещен в комментарии

Таким образом расшифрованные строки удобно размещаются вместе в окне IDA *xrefs* для этой функции, как показано на рисунке 6.


xrefs to f_decrypt

Dire	T	Address	Text
...	p	f_MachineCheck+5F	call f_decrypt; vmware
...	p	f_MachineCheck+89	call f_decrypt; virtualbox
...	p	f_MachineCheck+B9	call f_decrypt; oracle
...	p	f_MachineCheck+F0	call f_decrypt; parallels
...	p	f_MachineCheck+155	call f_decrypt; ioreg -l | grep -e "Manufacturer"
...	p	f_MachineCheck+43A	call f_decrypt; sysctl hw.model
...	p	f_CheckMachineType+67	call f_decrypt; system_profiler SPHardwareDataType 2>/dev/null | awk '/Boot ROM Version/
...	p	f_CheckMachineType+16C	call f_decrypt; MBP
...	p	f_CheckMachineType+1D4	call f_decrypt; MBA
...	p	f_CheckMachineType+23C	call f_decrypt; MB
...	p	f_CheckMachineType+2A4	call f_decrypt; MM
...	p	f_CheckMachineType+30C	call f_decrypt; IM
...	p	f_CheckMachineType+374	call f_decrypt; MP
...	p	f_CheckMachineType+3D8	call f_decrypt; XS

Рисунок 6. Xrefs to функции f_decrypt

Финальный вариант сценария можно найти на Github repository.

Вывод

Как уже было сказано, OceanLotus постоянно совершенствуют и обновляют свой набор инструментов. На этот раз кибергруппа усовершенствовала вредоносное ПО для работы с Mac-пользователями. Код не сильно изменился, но, поскольку многие Mac-пользователи игнорируют продукты для безопасности, защита малвари от обнаружения имеет второстепенное значение.

Продукты ESET уже детектировали этот файл на момент исследования. Поскольку сетевая библиотека, используемая для C&C-коммуникации, теперь зашифрована на диске, точный сетевой протокол, используемый атакующими, пока неизвестен.

Индикаторы компрометации

Индикаторы компрометации, а также атрибуты MITRE ATT&CK также доступны на GitHub.