МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Поиск ресурсов, созданных пользователем с данным SID**

КУРСОВАЯ РАБОТА

студентки 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Ежовой Елены Дмитриевны

|  |  |  |
| --- | --- | --- |
| Научный руководитель  доцент к. ф.-м. н. | \_\_\_\_\_\_\_\_\_\_ | А. В. Гортинский |
|  | подпись, дата |  |
| Заведующий кафедрой  д. ф.-м. н., доцент | \_\_\_\_\_\_\_\_\_\_ | М. Б. Абросимов |
|  | подпись, дата |  |

Саратов 2023

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение………………………………………………………………………….... | 3 |
| 1. Дескриптор безопасности……………………………………………………… | 4 |
| 1. Архитектура файловой системы NTFS……………………………………….. | 6 |
| 1. Программная реализация извлечения SID (полного) пользователя из $STANDARD\_INFORMATION…………………………………………………. | 19 |
| Заключение………………………………………………………………………... | 23 |
| Список использованных источников……………………………………………. | 24 |
| Приложение А. Листинг программы…………………………………………….. | 25 |

**ВВЕДЕНИЕ**

Актуальность темы состоит в том, что поиск ресурсов, созданных пользователем с определенным SID, имеет широкий спектр применений. Такой поиск может применяться при выявлении возможных нарушений безопасности, таких как несанкционированный доступ или использование привилегий другими пользователями. Также его можно использовать в устранении проблем с доступом или другими нарушениями прав доступа, в управлении ресурсами, например, определении того, какие файлы и папки находятся под контролем определенного пользователя и какие права доступа к ним у него есть.

Целью данной курсовой работы является поиск ресурсов, созданных пользователем с данным SID, в результате чего требуется написать программу, реализующую извлечение SID (полного) пользователя из $STANDARD\_INFORMATION.

Исходя из поставленной цели, необходимо решить следующие задачи:

1. осветить в работе теоретические аспекты, необходимые для реализации;
2. написать программу на высокоуровневом языке программирования, применив полученные теоретические знания на практике.

**1 Дескриптор безопасности**

Для полного понимания роли дескриптора безопасности в обеспечении безопасности Windows необходимо начать с определения консолидированной безопасности, так как дескриптор безопасности является непосредственно связанным с этой концепцией. Консолидированная безопасность в Windows является подходом к управлению безопасностью, который объединяет различные технологии и механизмы безопасности в единую систему управления безопасностью.

В свою очередь, дескриптором безопасности называется структура данных, содержащая информацию о безопасности, связанную с объектом, которая определяет, кто и какие действия с объектом может выполнять. Дескриптор безопасности является основой механизма контроля доступа в консолидированной безопасности Windows, так как он позволяет определить, кто и как может получать доступ к объектам в системе, и управлять этим доступом с помощью различных правил и политик безопасности. Так же стоит упомянуть, что он связан с другими элементами консолидированной безопасности Windows, такими как механизмы аутентификации, авторизации, шифрования и аудита, которые работают вместе для обеспечения безопасности системы и защиты данных и ресурсов.

В дескриптор безопасности входят следующие атрибуты:

1. номер версии – это версия модели безопасности SRM, использованная для создания дескриптора;
2. флаги – это дополнительные модификаторы, определяющие поведение или характеристики дескриптора;
3. SID владельца – это идентификатор безопасности владельца;
4. SID группы – это идентификатор безопасности основной группы объекта;
5. избирательный список управления доступом (DACL) – это список, который указывает, кто и какой доступ имеет к объекту;
6. системный список управления доступом (SACL) – это список, в котором содержатся сведения о том, какие операции какими пользователями должны регистрироваться в журнале аудита безопасности и конкретный уровень целостности объекта.

Список управления доступом (ACL) состоит из заголовка и нескольких элементов управления доступом (ACE), которых может и не быть. Существует два типа ACL-списков: DACL и SACL. В DACL в каждом ACE-элементе содержится SID и маска доступа, которая обычно определяет права доступа (чтение, запись, удаление и т. д.), предоставленные или запрещенные держателю SID. Аккумулирование прав доступа, предоставленных отдельными ACE- элементами, формирует набор прав доступа, предоставленных ACL-списком. Если в дескрипторе безопасности отсутствует DACL (то есть имеется нулевой DACL), полный доступ к объекту предоставляется кому угодно. Если DACL пуст (то есть в нем нуль ACE-элементов), никто не имеет доступа к объекту.

В SACL содержатся два типа ACE-элементов, ACE-элементы системного аудита и ACE-элементы объекта системного аудита. Эти ACE-элементы определяют, какие операции выполняются в отношении объекта путем указания подвергаемых аудиту пользователей или групп.

Таким образом, дескриптор безопасности Windows играет ключевую роль в обеспечении безопасности операционной системы.

**2 Архитектура файловой системы NTFS**

1. *Master File Table.*

NTFS (New Technology File System) – это файловая система, используемая операционной системой Windows для хранения и управления файлами на жестких дисках и других носителях данных. Она была разработана корпорацией Microsoft и является одной из наиболее распространенных файловых систем в мире.

Одна из важнейших концепций архитектуры NTFS гласит, что все служебные данные хранятся в файлах. В частности, это относится к административным данным базовой файловой системы, обычно скрываемым в других файловых системах. Файлы с административными данными могут находиться в любом месте тома, как обычные файлы. Таким образом, в отличие от других файловых систем NTFS не обладает жестко заданной структурой. Вся файловая система считается областью данных, и любой сектор может быть выделен файлу. Единственное фиксированное требование гласит, что первые секторы тома содержат загрузочный сектор и загрузочный код.

«Сердцем» NTFS является главная файловая таблица MFT (Master File Table), содержащая информацию обо всех файлах и каталогах. Каждый файл и каталог представлен как минимум одной записью таблицы, причем записи сами по себе очень просты. Их размер составляет 1 Кбайт, но только первые 42 байта имеют определенное предназначение. В остальных байтах хранятся атрибуты – небольшие структуры данных, выполняющие строго специализированную функцию. Если атрибуты не помещаются в одной записи, для файла создаются несколько записей MFT. В этом случае первая запись называется базовой записью MFT, а ее адрес сохраняется в одном из фиксированных полей всех остальных записей.

MFT содержит метаданные, такие как размер файла, дата создания и модификации, атрибуты и права доступа. Она также содержит ссылки на фрагменты файлов, которые разбросаны по диску.

Каждая запись в MFT имеет свой уникальный номер, который называется MFT-индекс. Файловая система использует MFT-индексы для быстрого поиска файлов и папок на диске. Как ни странно, нулевой записью MFT является запись для представления себя самой. Эта запись таблицы называется $MFT – она является заголовком для всей таблицы MFT. Эта запись содержит информацию о таблице MFT, такую как ее размер, номера первого и последнего кластеров на диске, сигнатуру файловой системы, а также информацию о специальных файлах и папках, которые являются ключевыми компонентами файловой системы NTFS. Нулевая запись также содержит в себе указатель на первую запись MFT, что позволяет системе быстро найти ее на диске и начать работу с файловой системой.

В первом поле каждой записи MFT хранится сигнатура; у стандартных записей это ASCII-строка «FILE». Если в записи обнаружена ошибка, в качестве сигнатуры может использоваться строка «BAAD». Поле флагов указывает, используется ли запись и представляет ли она каталог.

1. *Атрибуты записей MFT.*

Большая часть записей MFT используется для хранения объектов-атрибутов с определенным типом данных. Существует множество различных атрибутов, каждый из которых обладает своей собственной внутренней структурой. Например, атрибуты могут содержать информацию об имени файла, дате и времени, а также содержимом файла. Это является одним из ключевых отличий NTFS от других файловых систем, которые обычно читают и записывают содержимое файлов, в то время как NTFS читает и записывает атрибуты, включая те, которые инкапсулируют содержимое файлов.

Все атрибуты содержат две общие части: заголовок и содержимое. Заголовок определяет тип атрибута, его размер и имя. Он также содержит флаги, указывающие на сжатие или шифрование значения. Тип атрибута представляет собой числовой код, ассоциированный с типом хранящихся данных. Запись MFT может содержать несколько однотипных атрибутов. Атрибуту также назначается идентификатор, который обеспечивает его уникальность в рамках записи МFТ. Если запись содержит несколько однотипных атрибутов, они различаются по значению идентификатора.

Содержимое атрибута имеет произвольный формат и произвольный размер. Так как содержимое атрибутов может быть произвольное, то размеры могут достигать нескольких мегабайт или даже гигабайт. Сохранять такое количество данных в 1024-байтовых записях MFT неудобно, поэтому для решения этой проблемы в NTFS было принято решение разделить атрибуты на предусмотрена возможность хранения содержимого атрибутов в двух местах. Содержимое резидентных атрибутов хранится в записях MFT с заголовками. Содержимое нерезидентных атрибутов хранится во внешнем кластере файловой системы. В заголовке атрибута указано, является атрибут резидентным или нерезидентным. У резидентных атрибутов содержимое следует непосредственно за заголовком. Для нерезидентных атрибутов в заголовке содержится адрес кластера.

Так как каждый тип атрибута представляется неким числом, то стандартным атрибутам присваивается значение по умолчанию, но это значение можно переопределить при помощи файла метаданных файловой системы $AttrDef. Кроме числового идентификатора каждый тип атрибута обладает именем, которое состоит только из прописных букв и начинается со знака «$». В таблице 1 представлены некоторые стандартные атрибуты.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Идентификатор типа | Имя | Описание |
| 16 | $STANDARD\_INFORMATION | Общая информация (флаги; время создания, последнего обращения и модификации; владелец и идентификатор системы безопасности) |
| 32 | $ATTRIBUTE LIST | Список других атрибутов файла |
| 48 | $FILENAME | Имя файла в Unicode; время создания, последнего обращения и модификации |
| 80 | $SECURITY\_DESCRIPTOR | Время обращения и свойства безопасности файла |
| 128 | $DATA | Содержимое файла |
| 144 | $INDEX\_ROOT | Корневой узел индексного дерева |
| 160 | $INDEX\_ALLOCATION | Узлы индексного дерева, корень которого определяется атрибутом  $INDEX\_ROOT |
| 176 | $BITMAP | Битовая карта файла $MFT и его индексов |

Почти все записи MFT содержат атрибуты типов $FILE\_NAME и $STANDARD\_INFORMATION. Последний необходимо рассмотреть подробно, так как его использование является ключевым в практической части.

1. *Атрибут $STANDARD\_INFORMATION.*

Атрибут $STANDARD\_INFORMATION существует у всех файлов и каталогов; в нем хранятся основные метаданные. Именно здесь находятся основные временные штампы, а также информация о владельце, безопасности и квотах. Содержимое этого атрибута не является строго необходимым для хранения файлов, но многие функции прикладного уровня, предоставляемые Microsoft, зависят от него.

В последней версии NTFS, атрибут $STANDARD\_INFORMATION содержит следующие значения:

1. идентификатор владельца (Owner ID) – это числовой идентификатор пользователя, который является владельцем файла или папки;
2. идентификатор безопасности (Security ID) – это числовой идентификатор, который связывает пользователя или группу с соответствующей записью в списке управления доступом (Access Control List, ACL);
3. время создания файла или папки (Creation Time) – это дата и время, когда объект был создан;
4. время последнего доступа к файлу или папке (Last Access Time) – это дата и время последнего чтения или записи файла или папки;
5. время последнего изменения файла или папки (Last Modification Time) – это дата и время последнего изменения содержимого файла или папки;
6. флаги (Flags) – это битовые флаги, которые указывают на различные свойства файла или папки, такие как "скрытый", "системный" или "архивный";
7. размер файла (File Size) – это размер файла в байтах.

Разберем понятия идентификатор владельца (Owner ID) и идентификатор безопасности (Security ID) отдельно, чтобы получить более полное представление о их значении и использовании.

Начнем с идентификатора владельца (Owner ID). Он обычно всегда заполнен в файловой системе NTFS. Когда файл или папка создаются, система NTFS автоматически назначает владельца, который обычно является пользователем, создавшим объект. Однако, в редких случаях, возможно, что идентификатор владельца не будет заполнен или будет содержать некорректные данные. Это может произойти, например, если файл был создан или скопирован с другой файловой системы, которая не поддерживает идентификатор владельца или если файл был создан без авторизованного пользователя. Если идентификатор владельца не заполнен, это может привести к проблемам при работе с файлами и папками, такими как ограничения доступа и проблемы с безопасностью.

Перейдем к идентификатору безопасности (Security ID). Вместо использования имен, для идентификации всего, что производит в системе действия, Windows использует идентификаторы безопасности (SID).

SID представляет собой числовое значение переменной длины, состоящее из номера версии SID-структуры, 48-разрядного значения идентификатора полномочий и переменного количества 32-разрядных кодов значений подчиненных полномочий или относительных идентификаторов (RID). Значение полномочий идентифицирует агента, выдавшего SID, и этим агентом обычно является локальная система Windows или домен. Значения подчиненных полномочий идентифицируют представителей, имеющих отношение к выдавшему полномочия, а RID-идентификаторы являются просто способом, применяемым в Windows для создания уникальных SID на основе общего базового SID.

Из-за большой длины SID-идентификаторов Windows старается сгенерировать внутри каждого SID по-настоящему случайное значение, практически невозможно, чтобы Windows выдала один и тот же SID на машине или домене или где-либо еще дважды.

При текстуальном отображении каждый SID содержит префикс S, и его различные компоненты отделены друг от друга дефисами:

S-1-5-21-1463437245-1224812800-863842198-1128

В данном SID номером версии служит цифра 1, значением идентификатора полномочий служит цифра 5 (полномочия безопасности Windows), а затем следуют четыре значения подчиненных полномочий плюс один RID (1128), который составляет оставшуюся часть SID.

При установке Windows программа Windows Setup выдает компьютеру SID машины. Windows назначает SID-идентификаторы локальным учетным записям, имеющимся на компьютере. Каждый SID локальной учетной записи создается на основе исходного компьютерного SID и в конце имеет RID. RID-идентификатор для пользовательских учетных записей и групп начинается с 1000 и становится больше на 1 с каждым новым пользователем или группой.

В операционных системах семейства Windows можно получить идентификаторы безопасности с помощью командной строки или PowerShell, но пользователь для этого должен обладать правами администратора. Например, чтобы получить список всех SID в командной строке, нужно выполнить команду: whoami /all. На рисунке 1 представлен ответ на команду whoami /all.

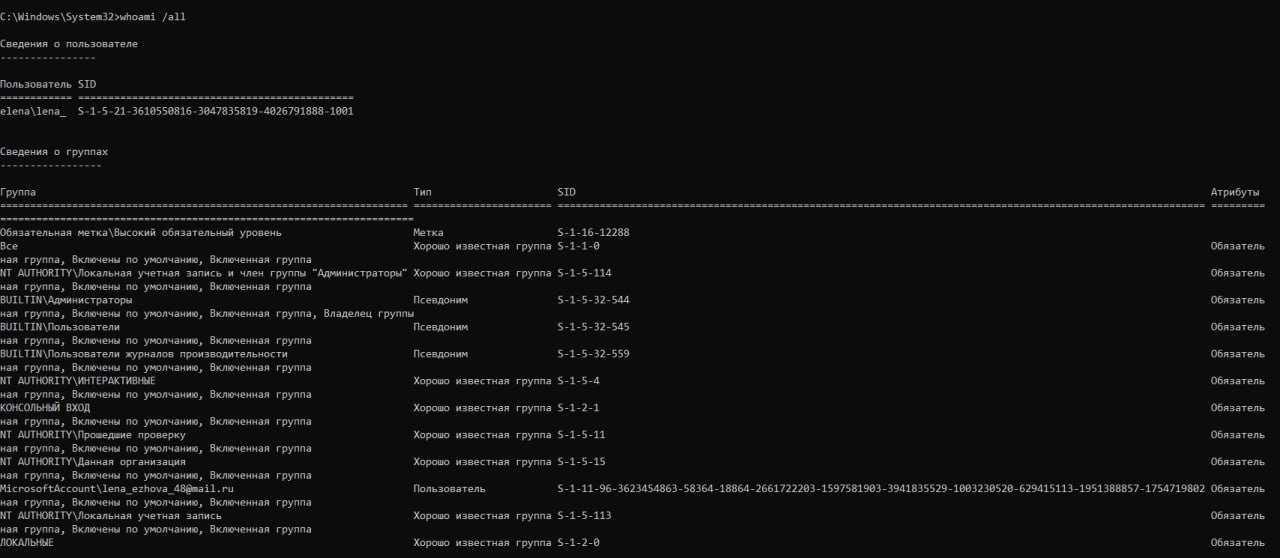


Рисунок 1 – вывод списка SID через командную строку

Чтобы вывести SID текущего пользователя нужно воспользоваться командой whoami /user, и ответ программы на эту команду будет выглядеть как на рисунке 2.

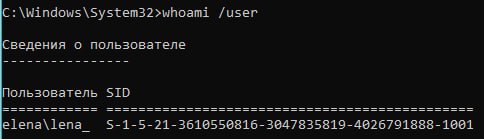


Рисунок 2 – вывод SID текущего пользователя через командную строку

Если требуется вывести SID конкретного пользователя через командную строку, то необходимо воспользоваться командой wmic useraccount where name='username' get sid, где вместо 'username' необходимо вставить имя пользователя, SID которого необходимо получить. На рисунке 3 изображен ответ командной строки на данную команду.

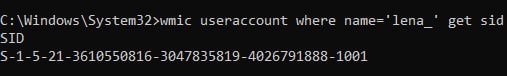


Рисунок 3 – вывод SID конкретного пользователя через командную строку

Все данные, полученные выше, можно получить и через PowerShell. Чтобы получить список SID всех пользователей воспользуемся командой Get-WmiObject -Class Win32\_Account | Select-Object Name, SID. Ответ PowerShell на этот запрос изображен на рисунке 4.

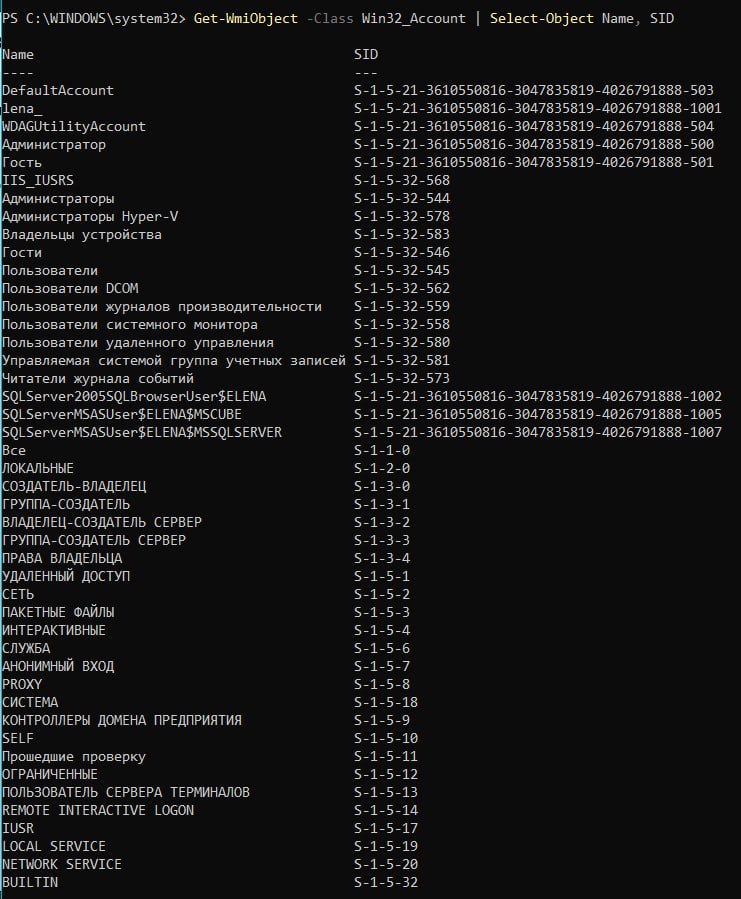


Рисунок 4 – вывод списка SID через PowerShell

В случае, если требуется показать SID текущего пользователя, то нужно применить команду (Get-WmiObject -Class Win32\_UserAccount -Filter "Name='$env:UserName'").SID. Информация, полученная в результате команды, изображена на рисунке 5.



Рисунок 5 – вывод SID текущего пользователя через PowerShell

Чтобы вывести SID конкретного пользователя через PowerShell следует применить команду (Get-WmiObject Win32\_UserAccount -Filter "Name='username'").SID, где вместо 'username' нужно вставить имя пользователя, SID которого требуется получить. На рисунке 6 продемонстрирован результат выполнения команды.



Рисунок 6 – вывод SID конкретного пользователя через PowerShell

Еще одна полезная команда в PowerShell – это (Get-Acl -Path 'way').GetOwner([System.Security.Principal.SecurityIdentifier]).Value. Ответом этой команды является вывод SID пользователя, который создал файл, путь к которому указан в одинарных кавычках. Пример использования этой команды изображен на рисунке 7.



Рисунок 7 – вывод SID пользователя, создавшего указанный файл

1. *Файлы метаданных файловой системы.*

Поскольку все данные в NTFS хранятся в виде файлов, то административные данные файловой системы также должны храниться в файлах. Такие файлы называются файлами метаданных.

Microsoft резервирует для файлов метаданных файловой системы первые 16 записей MFT. Неиспользуемые зарезервированные записи находятся в выделенном состоянии и содержат только базовую и общую информацию. Все файлы метаданных файловой системы отображаются в корневом каталоге, хотя обычно они скрываются от большинства пользователей. Имена файлов метаданных файловой системы начинаются с символа «$», а первая буква является прописной. В таблице 2 представлены основные файлы метаданных.

Таблица 2.

|  |  |  |
| --- | --- | --- |
| Запись | Имя файла | Описание |
| 0 | $MFT | Запись для самой таблицы MFT |
| 1 | $MFTMirr | Содержит резервную копию первых записей MFT |
| 2 | $LogFile | Содержит журнал транзакций метаданных |
| 3 | $Volume | Содержит информацию о томе – метка, идентификатор и версии |
| 4 | $AttrDef | Содержит информацию об атрибутах - значения идентификатора, имена, размеры |
| 5 | . | Содержит корневой каталог файловой системы |
| 6 | $Bitmap | Содержит признак выделения для каждого кластера файловой системы |
| 7 | $Boot | Содержит загрузочный сектор и загрузочный код файловой системы |
| 8 | $BadClus | Содержит кластеры, содержащие поврежденные секторы |
| 9 | $Secure | Содержит информацию системы безопасности и управления доступом к файлам |
| 10 | $Upcase | Содержит все символы Unicode в верхнем регистре |
| 11 | $Extend | Каталог с файлами необязательных расширений |

Рассмотрим подробнее файл метаданных $Secure, так как практическая часть напрямую связана с этим файлом.

1. *Файл метаданных $Secure.*

Дескрипторы безопасности используются для определения политики контроля доступа к файлам. B NTFS дескрипторы безопасности хранятся в файле метаданных файловой системы $Secure.

Файл $Secure содержит два индекса $SDH и $SII и один атрибут $SDS. Атрибут $SDS содержит дескрипторы безопасности, а два индекса используются при ссылках на дескрипторы. NTFS назначает каждому уникальному дескриптору на томе внутренний идентификатор безопасности NTFS и хеширует дескриптор безопасности в соответствии с простым хеш-алгоритмом, где хеш-значение – потенциально неуникальное сокращенное представление дескриптора. Стоит отметить, что внутренний идентификатор безопасности и SID не одно и то же, так как внутренние идентификаторы безопасности уникальны только в рамках файловой системы, тогда как коды SID глобально-уникальны. Рассмотрим индексы $SDH и $SII отдельно, чтобы получить более полное представление о их значении в механизме работы файла $Secure.

Файл $SDH (Security Descriptor Hash) – это индексный атрибут, который содержит хэш-таблицу, используемую для оптимизации поиска дескрипторов безопасности. Каждая запись в хэш-таблице содержит хэш-значение дескриптора безопасности и ссылку на соответствующий дескриптор в файле $SDS.

В свою очередь, индекс $SII – это индексный атрибут, который содержит таблицу, которая связывает SID с ссылкой на соответствующий дескриптор в файле $SDS. То есть можно сделать вывод о том, что атрибут $STANDARD\_INFORMATION и файл метаданных $Secure связаны напрямую, так как атрибут $STANDARD\_INFORMATION любого файла содержит идентификатор безопасности (SID) и именно по этому значению сортируется индекс $SII.

Изучим процесс назначения дескриптора безопасности файлу с использованием индекса $SDH и атрибута $SDS в NTFS с помощью файла метаданных $Secure. Назначив дескриптор безопасности файлу, NTFS получает хеш-значение дескриптора и просматривает индекс $SDH в поисках совпадений. NTFS сортирует элементы индекса $SDH согласно хеш-значению соответствующего дескриптора безопасности и сохраняет элементы в B+дереве (форма двоичного дерева, в каждом узле которого хранится несколько элементов). Обнаружив для дескриптора совпадение в индексе $SDH, NTFS определяет смещение дескриптора безопасности элемента из записи $SDS Offset и считывает дескриптор безопасности из атрибута $SDS. Если совпадают хеш-значения, но не дескрипторы безопасности, то NTFS ищет еще один совпадающий элемент в индексе $SDH. Если NTFS обнаруживает полное совпадение, то файл, которому назначен дескриптор безопасности, может установить связь с дескриптором безопасности в атрибуте $SDS. NTFS устанавливает связь, считывая идентификатор безопасности из элемента $SDH и сохраняя его в атрибуте $STANDARD\_INFORMATION файла. Если NTFS не обнаруживает в индексе $SDH элемента с дескриптором безопасности, совпадающим с назначаемым, значит, новый дескриптор уникален для тома, и NTFS назначает ему новый внутренний ID безопасности. Затем NTFS добавляет дескриптор безопасности в атрибут $SDS, который сортируется в B+ дереве по ID безопасности NTFS, и дополняет индексы $SDH и $SII элементами, указывающими на смещение дескриптора в массиве данных $SDS.

Перейдем к процессу проверки безопасности файлов и каталогов в NTFS с помощью файла метаданных $Secure. Когда приложение пытается открыть файл, NTFS отыскивает дескриптор безопасности файла с помощью индекса $SII. NTFS читает SID безопасности файла из атрибута $STANDARD\_INFORMATION, а затем NTFS использует запись $SII, чтобы найти указатель на дескриптор безопасности объекта в файле $SDS, который соответствует SID, найденному в атрибуте $STANDARD\_INFORMATION. По смещению в атрибуте $SDS система NTFS считывает дескриптор безопасности и завершает проверку безопасности.

NTFS не удаляет элементы файла $Secure, даже если с ним не связано ни одного файла на томе. Наличие неудаленных элементов не приводит к значительной потере дискового пространства, так как число уникальных дескрипторов безопасности на большинстве томов, даже используемых в течение длительного времени, сравнительно невелико. Благодаря универсальной индексации NTFS файлы и каталоги с одинаковыми параметрами безопасности эффективно используют общие дескрипторы. С помощью индекса $SII NTFS быстро отыскивает дескрипторы безопасности в файле $Secure в ходе проверок безопасности, а индекс $SDH позволяет быстро определить, имеется ли в файле $Secure ранее сохраненный дескриптор безопасности, пригодный для совместного использования с данным файлом.

1. **Программная реализация извлечения SID (полного) пользователя из $STANDARD\_INFORMATION**

В ходе работы была выполнена реализация извлечения SID (полного) пользователя из $STANDARD\_INFORMATION на языке программирования высокого уровня C++. Программа состоит из файла main.cpp – файла для запуска работы программы. На вход программы подается путь до файла. Необходимо вывести SID пользователя, создавшего этот файл.

Идея алгоритма состоит в следующем: найти указатель на SID владельца файла из атрибута $STANDARD\_INFORMATION, а далее найти соответствие в $Secure:$SDS. Поиск соответствия нужен, чтобы убедиться, что SID, полученный из атрибута $STANDARD\_INFORMATION, действительно принадлежит пользователю, который является владельцем файла. В случае, если этот SID не найден в списке доступа (DACL) файла, то это может означать, что владелец файла был изменен или файл был создан другим пользователем.

Таким образом, алгоритм можно поделить на два этапа: поиск указателя из атрибута $STANDARD\_INFORMATION и поиск соответствия в $Secure:$SDS.

1. *Описание кода программы для поиска указателя из атрибута $STANDARD\_INFORMATION.*

После запроса у пользователя программа получает путь к файлу, который создал пользователь, SID, которого нужно получить. Далее этот путь конвертируется в широкий символьный тип LPCWSTR, и по этому пути открывается файл с помощью функции , которая возвращает дескриптор файла. Затем используя функцию , код получает указатель на SID владельца файла из атрибута $STANDARD\_INFORMATION и записывает его в переменную .

Далее, программа выполняет вызов функции , чтобы получить имя пользователя и домен, связанные с указанным SID в переменной . Поскольку размеры буферов и неизвестны, функция вызывается дважды: первый раз, чтобы получить необходимые размеры буферов, а второй раз, чтобы заполнить эти буферы. Размеры буферов хранятся в переменных и .

Затем программа выделяет память для буферов и , используя функцию , и вызывает функцию снова, чтобы получить имя пользователя и домен, связанные с указанным SID.

В результате, все необходимые данные получены и можно переходить ко 2 этапу.

1. *Описание кода программы для поиска соответствия в $Secure:$SDS.*

На предыдущем шаге были получены предполагаемый SID владельца файла и имя учетной записи этого владельца, теперь можно приступить к проверке, что найденный SID относится именно к владельцу файла. Для этого программа вызывает функцию , чтобы получить информацию о безопасности файла, включая список контроля доступа DACL. Для этого функции необходимо передать указатель на структуру SECURITY\_DESCRIPTOR, которую функция вернула в переменную . Функция вызывается с флагом DACL\_SECURITY\_INFORMATION, указывающим, что требуется только список контроля доступа.

Если вызов не выполняется успешно, программа проверяет, была ли ошибка вызвана недостаточной длиной буфера. Если это так, программа повторно вызывает с буфером, выделенным при помощи функции .

Далее программа вызывает функцию , чтобы получить указатель на DACL в переменную . Затем идет проход по списку ACE (Access Control Entries) с помощью цикла, и для каждой записи в списке ACE выполняется проверка совпадения SID в записи ACE и SID владельца файла, полученному ранее в переменную . Если произошло совпадение, то программа вызывает функцию , чтобы получить строковое представление SID владельца файла.

В итоге программа выводит строковое представление SID владельца файла и имя его учетной записи в консоль.

1. *Инструкция для пользователя.*
2. Запускать IDE необходимо от имени администратора.
3. Программа работает для любых файлов и папок.
4. Запустить программу пользователь может, нажав на клавиатуре клавишу «F5», или через отладчик.
5. После запуска программы откроется консоль, в которой пользователю необходимо ввести путь до нужного файла без пробелов и кавычек.
6. В зависимости от результата, пользователь получит в консоли либо вывод SID и имени учетной записи пользователя, создавшего указанный файл, либо сообщение о том, что не удалось найти пользователя, создавшего указанный файл, в $Secure:$SDS.
7. *Примеры работы программы.*

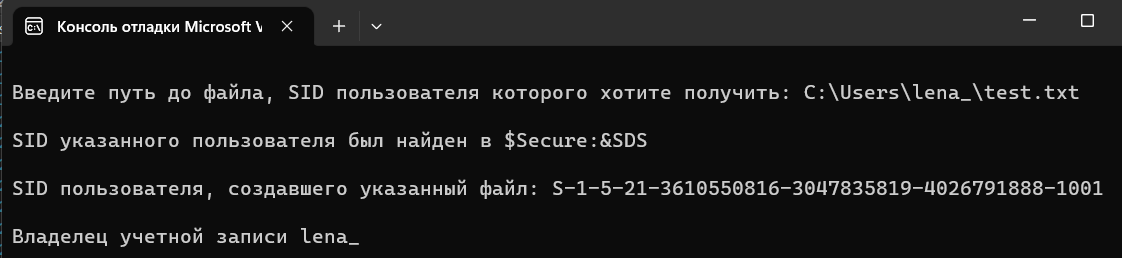


Рисунок 8 – вывод SID пользователя, создавшего указанный файл

Проверим ответ программы через PowerShell способом, описанным во главе 2 пункте 3.



Рисунок 9 – данные на рисунке доказывают правильность работы программы, так как SID-ы сошлись

**ЗАКЛЮЧЕНИЕ**

В ходе теоретической части работы были рассмотрены такие темы, как дескриптор безопасности и структура файловой системы NTFS.

В ходе практической части был реализован поиск SID пользователя, создавшего указанный файл. Умения, полученные в ходе выполнения практической части являются важным навыком для администраторов компьютерных систем и технических специалистов в области информационной безопасности, так как этот навык позволяет управлять безопасностью компьютерной среды, обнаруживать потенциальные угрозы и предотвращать несанкционированный доступ к информации.

Таким образом, все поставленные задачи решены, цель работы достигнута.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Кэрриэ, Б. Криминалистический анализ файловых систем / Б. Кэрриэ – СПб.: Питер, 2007. – 480 с.: ил. – дата обращения 08.05.2023, Яз. Рус
2. Русинович, М., Соломон, Д. Внутреннее устройство Microsoft Windows. – СПб.: Питер, 2013. – 800с.: ил. – (Серия «Мастер-класс»), дата обращения 08.05.2023, Яз. Рус.
3. «Техническая документация Microsoft», [Электронный ресурс] / URL: https://docs.microsoft.com/, дата обращения 08.09.2023, Яз. Рус.
4. Русинович, M. Возможности NTFS / М. Русинович // Журнал «Windows 2000 Magazine» – 2001. – дата обращения 08.05.2023.
5. Хорошо известные идентификаторы безопасности в операционных системах Windows [Электронный ресурс] : статья, открытый доступ. – URL: https://support.microsoft.com/ru-ru/help/243330/well-known-security-identifiers-in-windows-operating-systems (дата обращения 08.05.2023). – Загл. с экрана. – Яз. рус.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

*main.cpp*

#include <stdio.h>

#include <windows.h>

#include <tchar.h>

#include "accctrl.h"

#include "aclapi.h"

#include <iostream>

#include <sddl.h>

using namespace std;

#pragma comment(lib, "advapi32.lib")

int main(void) {

setlocale(LC\_ALL, "Russian");

DWORD dwRtnCode = 0;

PSID pSidOwner = NULL;

BOOL bRtnBool = TRUE;

LPTSTR AcctName = NULL;

LPTSTR DomainName = NULL;

DWORD dwAcctName = 1, dwDomainName = 1;

SID\_NAME\_USE eUse = SidTypeUnknown;

HANDLE hFile;

PSECURITY\_DESCRIPTOR pSD = NULL;

cout << endl;

cout << "Введите путь до файла, SID пользователя которого хотите получить: ";

string str = "";

cin >> str;

wstring widestr = wstring(str.begin(), str.end());

const wchar\_t\* widecstr = widestr.c\_str();

LPCWSTR text = widecstr;

hFile = CreateFile(

text,

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

if (hFile == INVALID\_HANDLE\_VALUE) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

\_tprintf(TEXT("Не удалось открыть файл. Код ошибки %d\n"), dwErrorCode);

return -1;

}

dwRtnCode = GetSecurityInfo(

hFile,

SE\_FILE\_OBJECT,

OWNER\_SECURITY\_INFORMATION,

&pSidOwner,

NULL,

NULL,

NULL,

&pSD);

if (dwRtnCode != ERROR\_SUCCESS) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

\_tprintf(TEXT("Не удалось получить информацию из атрибута &StandartInformation. Код ошибки: %d\n"), dwErrorCode);

return -1;

}

bRtnBool = LookupAccountSid(

NULL,

pSidOwner,

AcctName,

(LPDWORD)&dwAcctName,

DomainName,

(LPDWORD)&dwDomainName,

&eUse);

AcctName = (LPTSTR)GlobalAlloc(

GMEM\_FIXED,

dwAcctName \* sizeof(wchar\_t));

if (AcctName == NULL) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

\_tprintf(TEXT("Не удалось выделить память в нужном размере. Код ошибки: %d\n"), dwErrorCode);

return -1;

}

DomainName = (LPTSTR)GlobalAlloc(

GMEM\_FIXED,

dwDomainName \* sizeof(wchar\_t));

if (DomainName == NULL) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

\_tprintf(TEXT("Не удалось выделить память в нужном размере. Код ошибки: %d\n"), dwErrorCode);

return -1;

}

bRtnBool = LookupAccountSid(

NULL,

pSidOwner,

AcctName,

(LPDWORD)&dwAcctName,

DomainName,

(LPDWORD)&dwDomainName,

&eUse);

if (bRtnBool == FALSE) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

if (dwErrorCode == ERROR\_NONE\_MAPPED) {

\_tprintf(TEXT("Владелец учетной записи не найден для указанного SID.\n"));

}

else

\_tprintf(TEXT("Не удалось получить имя учетной записи.Код ошибки: %d\n"), dwErrorCode);;

return -1;

}

DWORD dwLengthNeeded = 0;

if (!GetKernelObjectSecurity(hFile, DACL\_SECURITY\_INFORMATION, pSD, 0, &dwLengthNeeded)) {

if (GetLastError() == ERROR\_INSUFFICIENT\_BUFFER) {

pSD = (PSECURITY\_DESCRIPTOR)LocalAlloc(LPTR, dwLengthNeeded);

if (pSD == NULL){

cout << "Не удалось выделить память в нужном размере." << endl;

CloseHandle(hFile);

return 1;

}

if (!GetKernelObjectSecurity(hFile, DACL\_SECURITY\_INFORMATION, pSD, dwLengthNeeded, &dwLengthNeeded)){

cout << "Не удалось получить копию дескриптора безопасности объекта" << endl;

LocalFree(pSD);

CloseHandle(hFile);

return 1;

}

PACL pDacl = NULL;

BOOL bDaclPresent = FALSE;

BOOL bDaclDefaulted = FALSE;

if (!GetSecurityDescriptorDacl(pSD, &bDaclPresent, &pDacl, &bDaclDefaulted)) {

cout << "Не удалось получить DACL." << endl;

LocalFree(pSD);

CloseHandle(hFile);

return 1;

}

for (DWORD i = 0; i < pDacl->AceCount; i++) {

PACCESS\_ALLOWED\_ACE pAce = NULL;

if (GetAce(pDacl, i, (LPVOID\*)&pAce)) {

if (EqualSid(pSidOwner, &(pAce->SidStart))) {

LPTSTR sidstring;

cout << endl;

cout << "SID указанного пользователя был найден в $Secure:&SDS" << endl;

if (!ConvertSidToStringSid(pSidOwner, &sidstring)) {

return GetLastError();

}

cout << endl;

cout << "SID пользователя, создавшего указанный файл: ";

printf("%ws\n", sidstring);

if (bRtnBool == TRUE) {

cout << endl;

\_tprintf(TEXT("Владелец учетной записи %s\n"), AcctName);

}

LocalFree(pSD);

CloseHandle(hFile);

return 0;

}

}

}

}

else{

cout << "Не удалось получить размер буфера дескриптора безопасности" << endl;

CloseHandle(hFile);

return 1;

}

}

cout << "Не удалось найти SID указанного пользователя в $Secure:&SDS." << endl;

CloseHandle(hFile);

return 0;

}