Лабораторная работа 1 (дополнительная)

ИССЛЕДОВАНИЕ ТЕХНОЛОГИИ WINDOWS MANAGEMENT INSTRUMENTATION

Цель работы:

Получить навыки работы с инструментарием управления Windows

Задача:

Написать клиент-серверное приложение под ОС Windows 7-10. Программа-клиент должна запрашивать по интерфейсу WMI информацию об удаленной системе, на которой работает программа-сервер. Возможна архитектура приложения, где роль сервера играет WMI API.

Требования:

- Все данные о системе должны быть получены с помощью обращения к WMI API;

- При установке соединения должна производится имперсонация клиента (через WMI);

- Необходимо запросить минимум 10 параметров системы, из которых обязательными является информация об установленных приложениях, а также информация из центра безопасности Windows (Windows Security Center, WSC) об установленном антивирусе, файерволле, противошпионском ПО;

- Программа должна быть написана на C/C++.

При сдаче лабораторной работы необходимо запускать клиента и сервер на разных хостах.

Теоретические сведения

Windows Management Instrumentation (WMI) — это одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows.

Технология WMI — это расширенная и адаптированная под Windows реализация стандарта WBEM, в основе которого лежит идея создания универсального интерфейса мониторинга и управления различными системами и компонентами распределенной информационной среды предприятия с использованием объектно-ориентированных идеологий и протоколов HTML и XML.

В основе структуры данных в WBEM лежит Common Information Model (CIM), реализующая объектно-ориентированный подход к представлению компонентов системы. CIM является расширяемой моделью, что позволяет программам, системам и драйверам добавлять в неё свои классы, объекты, методы и свойства.

WMI, основанный на CIM, также является открытой унифицированной системой интерфейсов доступа к любым параметрам операционной системы, устройствам и приложениям, которые функционируют в ней.

Важной особенностью WMI является то, что хранящиеся в нём объекты соответствуют динамическим ресурсам, то есть параметры этих ресурсов постоянно меняются, поэтому параметры таких объектов не хранятся постоянно, а создаются по запросу потребителя данных.

Так как WMI построен по объектно-ориентированному принципу, то все данные операционной системы представлены в виде объектов и их свойств и методов.

Все классы группируются в пространства имен, которые иерархически упорядочены и логически связаны друг с другом по определенной технологии или области управления. В WMI имеется одно корневое пространство имен Root, которое в свою очередь имеет 4 подпространства: CIMv2, Default, Security и WMI.

Классы имеют свойства и методы и находятся в иерархической зависимости друг от друга, то есть классы-потомки могут наследовать или переопределять свойства классов-родителей, а также добавлять свои свойства.

Свойства классов используются для однозначной идентификации экземпляра класса и для описания состояния используемого ресурса. Обычно все свойства классов доступны только для чтения, хотя некоторые из них можно модифицировать определенным методом. Методы классов позволяют выполнить действия над управляемым ресурсом.

К каждому экземпляру класса можно обратиться по полному пути, который имеет следующую структуру:

[\\ComputerName\NameSpace][:ClassName][.KeyProperty1=Value1][,KeyProperty2=Value2]…]

где

ComputerName – имя компьютера

NameSpace – название пространства имен

ClassName – имя класса

KeyProperty1=Value1, KeyProperty2=Value2 – свойства объекта и значение, по которому он идентифицируется.

Пример обращения к процессу с именем «Calc.exe», который запущен на локальной машине:

\\.\CIMv2:Win32_Process.Name="Calc.exe"

Экземпляры классов могут генерировать события, к которым можно подписываться. При наступлении события WMI автоматически создает экземпляр того класса, которому соответствует это событие. Такой механизм удобно использовать для выполнения определенной команды при наступлении определенного события, то есть следить за состоянием объектов операционной системы.

Общая безопасность в WMI реализуется на уровне операционной системы, а дополнительная политика безопасности основана на уровнях пространств имен и протокола DCOM. То есть если пользователь не имеет права делать какое-то действие через операционную систему, он не сможет это сделать и через WMI. Если же пользователю дано какое-то право в операционной системе, то это ещё не означает, что это право будет и в WMI, так как в WMI действуют дополнительные параметры безопасности на уровне пространств имен.

Каждый объект операционной системы имеет своё описание безопасности (SD) со своим списком доступа (ACL), в котором перечислены идентификаторы пользователей (SID) и их привилегии. Каждое пространство имен может иметь собственное SD со своим ACL, где пользователям могут быть назначены разрешения на чтение данных, выполнение методов, запись классов и данных и другие. Данные о дополнительных разрешениях хранятся в репозитории WMI. Отдельные классы из пространств имен не имеют собственных описаний безопасности, они наследуют их от своего пространства имен.

По умолчанию администратор компьютера имеет полные права на использование WMI, а остальные пользователи могут лишь вызывать методы, считывать данные и записывать в репозиторий экземпляры классов провайдеров WMI.

Для доступа к инфраструктуре WMI используется протокол DCOM, через который пользователь подключается к WMI. Чтобы определить, какие права будут у подключившегося пользователя, используются механизмы олицетворения и аутентификации протокола DCOM.

Уровни олицетворения могут принимать следующие значения:

Anonymous	Анонимный	WMI-объект не может получить информацию о пользователе — доступ по такому типу не предоставляется
Identify	Идентификация	WMI-объект запрашивает маркер доступа пользователя — доступ предоставляется только локально
Impersonate	Имперсонация	WMI-объект имеет такие же права, какие имеет пользователь — рекомендуемый уровень для выполнения команд на удаленном компьютере
Delegate	Делегирование	WMI-объект может обратиться от имени пользователя к другому WMI-объекту — нерекомендуемый уровень, так

		как команды можно выполнять удаленно через цепочку из нескольких компьютеров

Уровни аутентификации (подлинности) могут принимать следующие значения:

None	Отсутствует	Проверка подлинности отсутствует
Default	По умолчанию	Стандартные настройки безопасности, которые задаются компьютером-целью команды
Connect	Подключение	Проверка только во время подключения к компьютеру-цели команды, проверка в ходе работы отсутствует
Call	Вызов	Проверка подлинности при каждом запросе к компьютеру-цели команды, заголовки пакетов подписываются, но содержимое не шифруется
Pkt	Пакет	Проверка подлинности всех пакетов к компьютеру-цели команды, заголовки пакетов подписываются, но содержимое не шифруется
PktIntegrity	Целостность пакета	Проверка подлинности и целостности всех пакетов к компьютеру-цели команды, заголовки пакетов подписываются, но содержимое не шифруется
PktPrivacy	Секретность пакета	Проверка подлинности и целостности всех пакетов к компьютеру-цели команды, заголовки и содержимое пакетов подписываются и шифруются

Для обращения к объектам WMI используется специфический язык запросов WMI Query Language (WQL), который является одной из разновидностей SQL. Основное его отличие от ANSI SQL — это невозможность изменения данных, то есть с помощью WQL возможна лишь выборка данных с помощью команды SELECT. Помимо ограничений на работу с объектами, WQL не поддерживает такие операторы как DISTINCT, JOIN, ORDER, GROUP, математические функции. Конструкции IS и NOT IS применяются только в сочетании с константой NULL.

Запросы WQL обычно применяются в скриптах, но их также можно протестировать в программе Wbemtest и в консольной утилите Wmic (утилита wmic не требует написания ключевого слова SELECT и полей выборки)

Пример получения данных с удаленного компьютера

Чтобы получить данные WMI с удаленного компьютера, необходимо:

1. Инициализировать COM вызовом CoInitializeEx .
2. Инициализировать безопасность процесса COM, вызвав CoInitializeSecurity .
3. Получить исходный локатор в WMI, вызвав CoCreateInstance .
4. Получить указатель на IWbemServices для пространства имен на удаленном компьютере, вызвав IWbemLocator::ConnectServer . При подключении к удаленному компьютеру необходимо знать имя компьютера, домен, имя пользователя и пароль удаленного компьютера, к которому вы подключаетесь. Все эти атрибуты передаются в метод IWbemLocator::ConnectServer . Кроме того, убедитесь, что имя пользователя на компьютере, который пытается подключиться к удаленному компьютеру, имеет правильные права доступа на удаленном компьютере.
5. Создать структуру COAUTHIDENTITY, чтобы предоставить учетные данные для установки защиты прокси-сервера.
6. Установить защиту прокси-сервера IWbemServices.
7. Использовать указатель IWbemServices для запросов WMI.
   Следующий WQL-запрос является одним из аргументов метода.
   SELECT * FROM Win32_OperatingSystem
   Результат этого запроса хранится в указателе IEnumWbemClassObject.
   Установите защиту проксимера IEnumWbemClassObject.
8. Получить и отобразите данные из запроса WQL.

Следующий пример демонстрирует приведенные выше действия:

```
#include <iostream>
#include <comdef.h>
#include <Wbemidl.h>
#include <wincred.h>
#include <strsafe.h>

#pragma comment(lib, "wbemuuid.lib")
#pragma comment(lib, "credui.lib")
#pragma comment(lib, "comsuppw.lib")

#define _WIN32_DCOM
#define UNICODE


using namespace std;


int __cdecl main(int argc, char **argv)
{
    HRESULT hres;

    // Шаг 1: --------------------------------------------------
    // Инициализация COM. ------------------------------------------

    hres =  CoInitializeEx(0, COINIT_MULTITHREADED);
    if (FAILED(hres))
    {
        cout << "Failed to initialize COM library. Error code = 0x"
             << hex << hres << endl;
        return 1;
    }

    // Шаг 2: --------------------------------------------------
    // Установка уровней безопасности COM ------------------------
```

```
hres =  CoInitializeSecurity(
    NULL,
    -1,
    NULL,
    NULL,
    RPC_C_AUTHN_LEVEL_DEFAULT,
    RPC_C_IMP_LEVEL_IDENTIFY,
    NULL,
    EOAC_NONE,
    NULL
    );


if (FAILED(hres))
{
    cout << "Failed to initialize security. Error code = 0x"
        << hex << hres << endl;
    CoUninitialize();
    return 1;
}

// Шаг 3: ---------------------------------------------------
// Создание локатора WMI -------------------------

IWbemLocator *pLoc = NULL;

hres = CoCreateInstance(
    CLSID_WbemLocator,
    0,
    CLSCTX_INPROC_SERVER,
    IID_IWbemLocator, (LPVOID *) &pLoc);

if (FAILED(hres))
{
    cout << "Failed to create IWbemLocator object."
        << " Err code = 0x"
        << hex << hres << endl;
    CoUninitialize();
    return 1;
}

// Шаг 4: ---------------------------------------------------
// Подключение к WMI через IWbemLocator::ConnectServer

IWbemServices *pSvc = NULL;

// Получение реквизитов доступа к удаленному компьютеру
CREDUI_INFO cui;
bool useToken = false;
bool useNTLM = true;
wchar_t pszName[CREDUI_MAX_USERNAME_LENGTH+1] = {0};
wchar_t pszPwd[CREDUI_MAX_PASSWORD_LENGTH+1] = {0};
wchar_t pszDomain[CREDUI_MAX_USERNAME_LENGTH+1];
wchar_t pszUserName[CREDUI_MAX_USERNAME_LENGTH+1];
wchar_t pszAuthority[CREDUI_MAX_USERNAME_LENGTH+1];
BOOL fSave;
DWORD dwErr;

memset(&cui,0,sizeof(CREDUI_INFO));
cui.cbSize = sizeof(CREDUI_INFO);
cui.hwndParent = NULL;
cui.pszMessageText = TEXT("Press cancel to use process token");
cui.pszCaptionText = TEXT("Enter Account Information");
```

```
    cui.hbmBanner = NULL;
    fSave = FALSE;

    dwErr = CredUIPromptForCredentials(
        &cui,
        TEXT(""),
        NULL,
        0,
        pszName,
        CREDUI_MAX_USERNAME_LENGTH+1,
        pszPwd,
        CREDUI_MAX_PASSWORD_LENGTH+1,
        &fSave,
        CREDUI_FLAGS_GENERIC_CREDENTIALS |
        CREDUI_FLAGS_ALWAYS_SHOW_UI |
        CREDUI_FLAGS_DO_NOT_PERSIST);

    if(dwErr == ERROR_CANCELLED)
    {
        useToken = true;
    }
    else if (dwErr)
    {
        cout << "Did not get credentials " << dwErr << endl;
        pLoc->Release();
        CoUninitialize();
        return 1;
    }

    // change the computerName strings below to the full computer name
    // of the remote computer
    if(!useNTLM)
    {
        StringCchPrintf(pszAuthority, CREDUI_MAX_USERNAME_LENGTH+1,
L"kERBEROS:%s", L"COMPUTERNAME");
    }

    // Подключение к пространству имен root\cimv2
    //---------------------------------------------------------

    hres = pLoc->ConnectServer(
        _bstr_t(L"\\\\COMPUTERNAME\\root\\cimv2"),
        _bstr_t(useToken?NULL:pszName),
        _bstr_t(useToken?NULL:pszPwd),
        NULL,
        NULL,
        _bstr_t(useNTLM?NULL:pszAuthority),
        NULL,
        &pSvc
        );

    if (FAILED(hres))
    {
        cout << "Could not connect. Error code = 0x"
             << hex << hres << endl;
        pLoc->Release();
        CoUninitialize();
        return 1;
    }

    cout << "Connected to ROOT\\CIMV2 WMI namespace" << endl;
```

```
    // Шаг 5: ------------------------------------------------
    // Создание структуры COAUTHIDENTITY

    COAUTHIDENTITY *userAcct =  NULL ;
    COAUTHIDENTITY authIdent;

    if( !useToken )
    {
        memset(&authIdent, 0, sizeof(COAUTHIDENTITY));
        authIdent.PasswordLength = wcslen (pszPwd);
        authIdent.Password = (USHORT*)pszPwd;

        LPWSTR slash = wcschr (pszName, L'\\');
        if( slash == NULL )
        {
            cout << "Could not create Auth identity. No domain specified\n" ;
            pSvc->Release();
            pLoc->Release();
            CoUninitialize();
            return 1;
        }

        StringCchCopy(pszUserName, CREDUI_MAX_USERNAME_LENGTH+1, slash+1);
        authIdent.User = (USHORT*)pszUserName;
        authIdent.UserLength = wcslen(pszUserName);

        StringCchCopyN(pszDomain, CREDUI_MAX_USERNAME_LENGTH+1, pszName,
slash - pszName);
        authIdent.Domain = (USHORT*)pszDomain;
        authIdent.DomainLength = slash - pszName;
        authIdent.Flags = SEC_WINNT_AUTH_IDENTITY_UNICODE;

        userAcct = &authIdent;

    }

    // Шаг 6: ------------------------------------------------
    // Установка защиты прокси сервера ------------------

    hres = CoSetProxyBlanket(
       pSvc,
       RPC_C_AUTHN_DEFAULT,
       RPC_C_AUTHZ_DEFAULT,
       COLE_DEFAULT_PRINCIPAL,
       RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
       RPC_C_IMP_LEVEL_IMPERSONATE,
       userAcct,
       EOAC_NONE
    );

    if (FAILED(hres))
    {
        cout << "Could not set proxy blanket. Error code = 0x"
            << hex << hres << endl;
        pSvc->Release();
        pLoc->Release();
        CoUninitialize();
        return 1;
    }

    // Шаг 7: ------------------------------------------------
    // Получение данных через WMI ----

    // Например, получим имя ОС
```

```
IEnumWbemClassObject* pEnumerator = NULL;
hres = pSvc->ExecQuery(
    bstr_t("WQL"),
    bstr_t("Select * from Win32_OperatingSystem"),
    WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
    NULL,
    &pEnumerator);

if (FAILED(hres))
{
    cout << "Query for operating system name failed."
        << " Error code = 0x"
        << hex << hres << endl;
    pSvc->Release();
    pLoc->Release();
    CoUninitialize();
    return 1;
}

hres = CoSetProxyBlanket(
    pEnumerator,
    RPC_C_AUTHN_DEFAULT,
    RPC_C_AUTHZ_DEFAULT,
    COLE_DEFAULT_PRINCIPAL,
    RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
    RPC_C_IMP_LEVEL_IMPERSONATE,
    userAcct,
    EOAC_NONE
    );

if (FAILED(hres))
{
    cout << "Could not set proxy blanket on enumerator. Error code = 0x"
         << hex << hres << endl;
    pEnumerator->Release();
    pSvc->Release();
    pLoc->Release();
    CoUninitialize();
    return 1;
}


SecureZeroMemory(pszName, sizeof(pszName));
SecureZeroMemory(pszPwd, sizeof(pszPwd));
SecureZeroMemory(pszUserName, sizeof(pszUserName));
SecureZeroMemory(pszDomain, sizeof(pszDomain));


// Шаг 9: -------------------------------------------------
// Получение данных из запроса в шаге 7 ------------------

IWbemClassObject *pclsObj = NULL;
ULONG uReturn = 0;

while (pEnumerator)
{
    HRESULT hr = pEnumerator->Next(WBEM_INFINITE, 1,
        &pclsObj, &uReturn);

    if(0 == uReturn)
    {
        break;
    }
```

```
        VARIANT vtProp;

        // Выбираем поле Name
        hr = pclsObj->Get(L"Name", 0, &vtProp, 0, 0);
        wcout << " OS Name : " << vtProp.bstrVal << endl;

        // Выбираем поле свободной памяти
        hr = pclsObj->Get(L"FreePhysicalMemory",
            0, &vtProp, 0, 0);
        wcout << " Free physical memory (in kilobytes): "
            << vtProp.uintVal << endl;
        VariantClear(&vtProp);

        pclsObj->Release();
        pclsObj = NULL;
    }

    // Очистка
    // ========

    pSvc->Release();
    pLoc->Release();
    pEnumerator->Release();
    if( pclsObj )
    {
        pclsObj->Release();
    }

    CoUninitialize();

    return 0;

}
```

Контрольные вопросы:

1) Что такое WMI и зачем данный инструмент нужен?

2) Какие основные возможности у WMI?

3) Опишите архитектуру и принципы работы WMI.