Хакери и RPC, DCOM и ActiveX

RPC

В RPC се поддържат 2 типа handles: binding и context. Вторите са подобни на cookies , съхраняват се в клиента и се изпращат заедно с всяка заявка към сървъра.

Заплахите в RPC идват от:

1. **хакер изпраща “malformed” данни към RPC end-point, те не се отработват правилно, което води до някакъв срив;**
2. **незащитени данни се движат от/към сървър и хакер успява да прихване пакет и го разгледа;**
3. **хакер прихваща незащитени данни и ги модифицира.**

Защити:

* ползвайте /robust MIDL ключ на компилатора в Windows 2000 и нагоре. Добавя run-time checking към пристигащите в сървър пакети “Any malformed packet is rejected by the RPC marchaling engine”

* ползвайте атрибут [range] в IDL файл за управление размерността на уязвими параметри или полета.:

```
void Message([in, range(0,1023)] long lo,
                    [in, range(0,1023)] long hi,
                    [size_is(lo, hi)] char **ppData);
```

**** използва се съвместно с /robust

* изисквайте автентикация при свързване. Или поне при работа с важни данни. Това изисква добавен код и в кл. и в сървъра

настройки в клиента

```
status = RpcBindingSetAuthInfo( phone_Handle,
                                ...,
                                RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
                                RPC_C_AUTHN_GSS_NEGOTIATE,....);
```

Аргумент AuthnLevel (RPC Security Setting Levels):

RPC_C_AUTHN_LEVEL_DEFAULT	по подразбиране с настройки на сървъра
RPC_C_AUTHN_LEVEL_NONE	
RPC_C_AUTHN_LEVEL_CONNECT	автентикиране при първо свързване със сървър
RPC_C_AUTHN_LEVEL_CALL	при начало на всеки RPC call.
RPC_C_AUTHN_LEVEL_PKT	проверява дали всички данни са от “expected sender”
RPC_C_AUTHN_LEVEL_PKT_INTEGRITY	както по-горе, + проверка дали данните са модифицирани
RPC_C_AUTHN_LEVEL_PKT_PRIVACY	както по-горе + криптиране на данните

следователно е възможно автентикация + интегритет + защита
(инф. за идентитета се добавя към binding handle, който се предава I параметър при всяко RPC повикване.)

настройки в сървъра:

за да можете да анализирате client connection&security settings, следва да се инсталира authentication handler и в сървъра:

```
status = RpcServerRegisterAuthInfo( ..,
                                    RPC_C_AUTHN_GSS_NEGOTIATE,….);
```

параметърът определя как се автентикира клиента.

Сървърът извлича клиентската инф. за автентикиране, защита и т.н. от client binding handle с RpcBindingInqAuthClient() разположена в отдалечената процедура:

```
// RPC сървърна ф-ия с проверки
void Message( handle_t hPhone, unsigned char *szMsg)  {
        DWORD dwAuthn;
        RPC_AUTHN_HANDLE hPrivs;

        RPC_STATUS status = RpcBindingInqAuthClient( hPhone, &hPrivs,
                                                     &dwAuthn,…);
        if(status != RPC_S_OK) {
                printf("…  );
        RpcRaiseException(ERROR_ACCESS_DENIED);}
```

```
// проверка за ниво
if( dwAuthn < RPC_C_AUTHN_LEVEL_PKT) {…..  }
// за да се обработят данните по идентичността на
//  свързания клиент, за кратък период, идентичността му
//  се “възприема” от сървъра с функцията:
//  RPC_STATUS RPC_ENTRY RpcImpersonateClient(
//  RPC_BINDING_HANDLE BindingHandle              );

if(RpcImpersonateClient(.. ) != RPC_S_OK) {..}
GetUserName( szName…);
// RpcRevertToSelf() end impersonation and reestablish process
// security identity.

RpcRevertToSelf();
}
```

използвайте [strict_context_handle]атрибут в ACF файл

за случаите когато 1 context handle се ползва с > 1 интерфейси.
Той се предава при първото повикване на интерфейсен метод и не е допустимо да се приема при повиквания за методи от друг интерфейс.
Ето пример на опасен код:

В IDL файла

```
        interface PrinterOperations
{               typedef context_handle void *PRINTER_CONTEXT;
                void OpenPrinter([in, out] PRINTER_CONTEXT *ctx);
                void UsePrinter( [in] PRINTER_CONTEXT ctx);
                ..............}
        interface FileOperations{
                typedef context_handle void *FILE_CONTEXT;
                void OpenFile([in, out] FILE_CONTEXT *ctx);
                void UseFile( [in] FILE_CONTEXT ctx);
                ..............}
```

Ако хакер използва printer context във файлов интерфейсен метод, той вероятно ще срине RPC сървъра поради грашка в преобразуването:

```
void UseFile(FILE_CONTEXT ctx) {
        CFileManipulator cFile = (CFileManipulator*) ctx;
        ..... }
```

Това хакерът ще постигне така:

```
void *ctxAttacker;
OpenPrinter(&ctxAttacker);
UseFile(ctxAttacker);      // всъщност се изпраща PRINTER_CONTEXT
```

За да се избегне това, в ACF файла:

```
[ explicit_handle, strict_context_handle]
interface PrinterOperations{}
interface FileOperations{}
```

Тогава RPC runtime ще проверява дали всеки context handle подаван на PrinterOperations е създаден от PrinterOperations и същото за FileOperations

•не разчитайте на информацията от context handles, който може по някакъв начин да е прихванат и дори да е криптиран (т.е. неразбираем за хакера) да се използва за вредни действия.

* някои продукти тестват правата при пръв достъп с context handle и приемат, че всички следващи повиквания от същия handle имат същата идентичност. Тествайте правата непосредствено преди всяка важна операция и то по всякакъв друг , допълващ начин (освен context handle info).

* RPC се старае да гарантира context handle в рамките на 1 мрежова сесия (това зависи доколко мрежовия транспорт може да гарантира идентитета в рамките на сесия), но RPC не гарантира context handle идва от 1 security session. Там е уязвимостта за атаки.

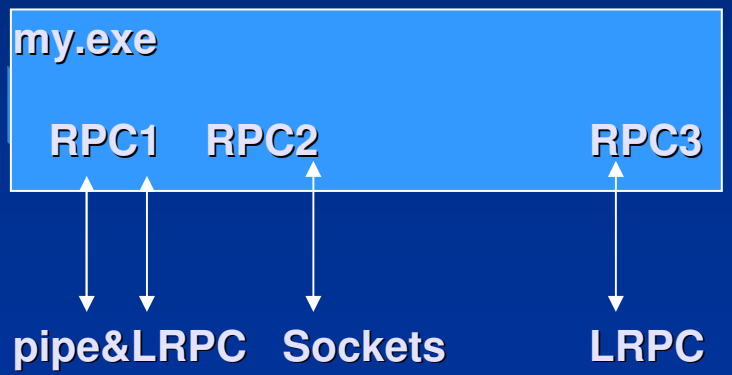
използвайте security callback RPC функции RpcServerRegisterIf2() или RpcServerRegisterIfEx() вместо RpcServerRegisterIf() за повишаване защитата при RPC повиквания. С тях регистрирате ф-ия в RPC сървъра, като указвате име на ф-ия викана от RPC runtime, която проверява дали клиентът има права да повика RPC ф-ията от този интерфейс.
Кодът позволява клиент да се свърже само при RPC_C_AUTHN_LEVEL_PKT или по-голям. Пример на код, позволяващ свързване на клиент, само с определено ниво на сигурност:

```
……….
// security callback ф-ията се вика когато RPC ф-ия е повикана
RPC_STATUS SecurityCallBack( RPC_IF_HANDLE idIF, void *ctx) {
        RPC_AUTHZ_HANDLE hPrevs;
        DWORD dwAuthn;
        RPC_STATUS status = RpcBindingInqAuthClient( ctx, &hPrivs,  &dwAuthn,…);
        if(status != RPC_S_OK)
                { /* error */ ……}
// проверява нивото на достъп
        if( dwAuthn < RPC_C_AUTHN_LEVEL_PKT) {
                /* error */ ….}
        return RPC_S_OK;
        }
……….
void main() {
……….
status = RpcServerRegisterIfEx(…, SecurityCallBack);
…}
```

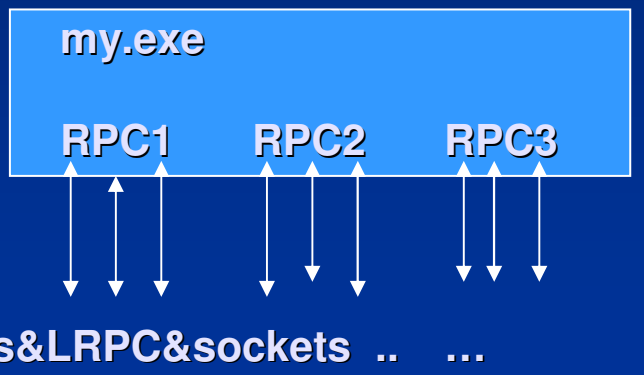
Влияние на множество RPC Servers в общ процес

RPC е отделена (и неинформирана) от мрежовия протокол чрез който се вика.
Това има страничен ефект:
ако RPC сървър е в общ процес с други RPC сървъри, всички те слушат по отделни протоколи:

концепция



реалност



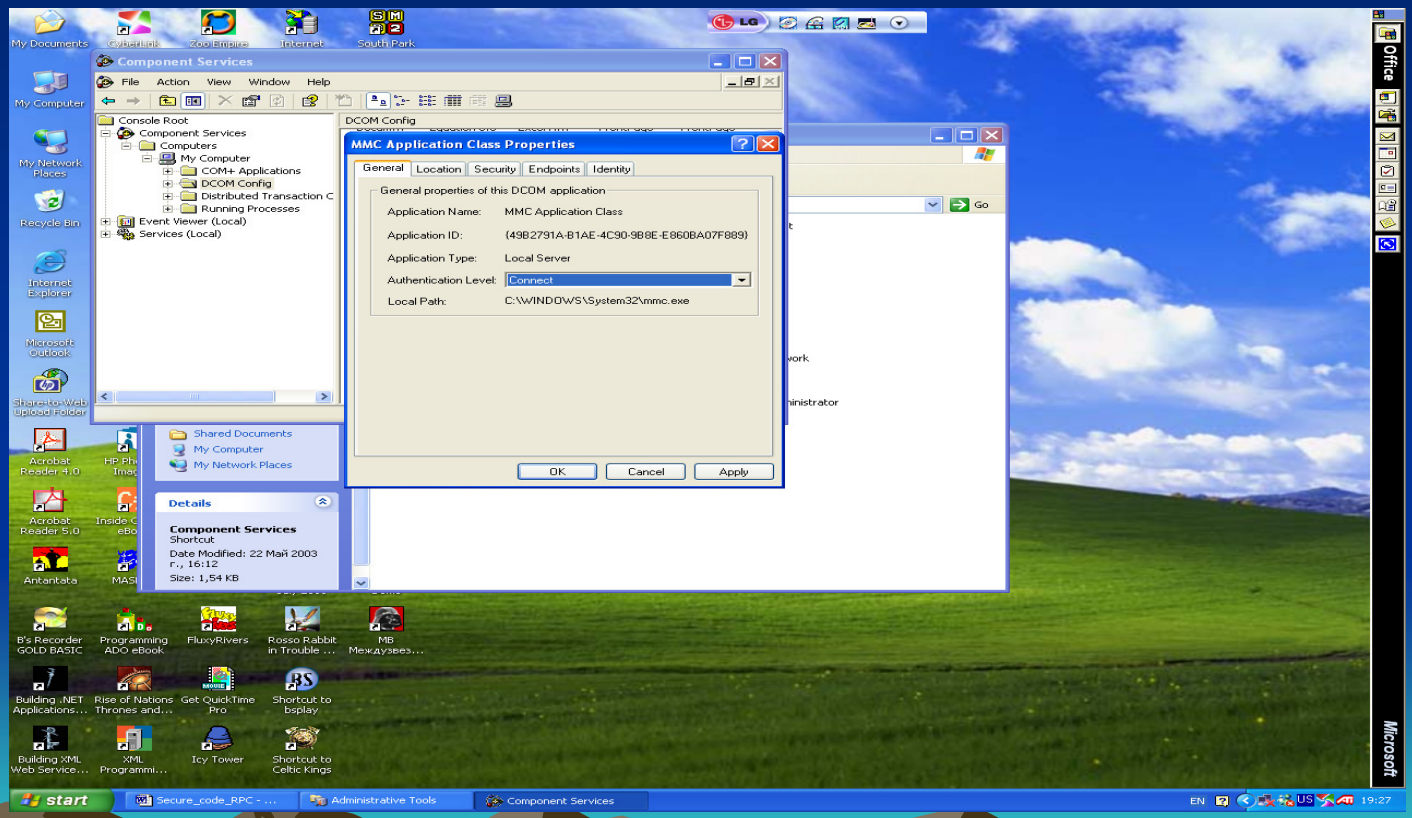
ако сте защитени при LRPC, това не значи че сте защитени и при канали или sockets.Следователно незащитен е и целия процес.
Проверявайте по кой протокол е дошла заявката:

RpcBindingToStringBinding() и след това RpcStringBindingParse(), който отделя частите на binding низа и от тях можете да отделите името на протокола

Сигурност с DCOM

Той е обвивка на RPC, която цели да позволи COM повиквания да се изпълняват в мрежа. Следователно всичко за RPC е валидно и тук.

1. Настройки на COM и DCOM на системно ниво
(Distributed COM Configuration Properties) диалог или
Microsoft Management Console.Позволяват настройки по достъпа на COM и DCOM обекти:



-on/off за всички компоненти

- разрешение за COM Internet Service за този компютър (позволяват RPC в/ху HTTP)

- Authentication level опция: същите като RPC security level

- Provide additional security for reference tracking опция: въвежда следене и забрана друг процес (хакер) да вика IUnknown::Release() за COM обекта.

- настройки на access permissions по отделни потребители – кои могат да пипат текущия COM обект, кои могат да стартират обект (launch permissions), кои могат да редактират configuration информация (configuration permission)

- кои протоколи се разрешават за DCOM да ги използва (както и кои портове).Определяне на портове облекчава настройките на firewall (ако COM ще преминава през защитна стена) administrator. Възможна е и настройка през firewall в която connection може да се създава в 1 посока, но не и в обратната.

2.Настройки на сигурността на ниво приложение

В горните настройки на Windows в Application Tab е допустимо извършване на настройки на ниво приложение.
Всички обекти в едно приложение (или един DLL) трябва да имат еднакви позволения. В противен случай – разделяне в отделни dll и тогава DCOM приложението ще се изпълнява при различни user contexts.

3. DCOM Security настройки (в клиент и в сървър) в кода

CoInitializesecurity() // в кл и в сървър.
Настройки на :
- кой има достъп до обект в сървъра. (Веднаж настроен, не може да се променя);
- ниво на защита (RPC security settings levels);

IClientSecurity::SetBlanket() // в клиент.
(Позволява настройки на опции по security и то за отделен интерфейс.)

Сигурност при ActiveX контролите

Могат да принесат големи вреди тъй като се вграждат в HTML и използват от скриптови езици. Тъй като e-mail клиенти позволяват изобразяване на HTML форматиран текст, то ActiveX могат да се съдържат и в тях (при определени настройки на на mail application). Outlook 2002 не допуска по подразбиране ActiveX като част от mail.

Производителите на контроли ги маркират като:
- **safe for initialization (SFI):** При инстанциириране, контролът зарежда данни (локални или отдалечени). Ползва IPersist подобни интерфейси.Източниците на данни могат да са “несигурни”. Ако е гарантирано , че не може да има проблеми с използваните данни , говорим за SFI контрол
- **safe for Scripting(SFS):** Гарантирано е че вграждане в script код не може да генерира проблеми, дължащи се на контрола.

Пример за контрол който не е SFS:

Контрол от достоверен източник е предназначен за download от потребителите на сайта. Контролът има метод Print() за отпечатване на произволен файл на произволен принтер.

Следователно той има достъп до ресурси на потребител и с него могат да се отпечатат (на чужд компютър) и разгледат файлове на user, без негово знание

Контрол, макар и safe (когато просто се използва от user), може да стане unsafe, когато е вкаран в автоматизация от untrusted script or Web page. (Напр. Excel е trusted tool, но чрез хакерски скрипт, могат да се използват негови automation ф-ии за изтриване на файл, например)

Как да маркирате контрола си като SFI или SFS :

msdn.microsoft.com --> safe for scripting

Кога контрол е SFS или SFI:

Листвате всички events, methods, properties на контрола. Ако никое от тях не:
- прави достъп до информация от локалния компютър или мрежа (особено в регистъра);
- извлича и не работи с лична информация като keys, passwords, documents;
- модифицира или трие информация от локалния компютър или мрежа;
- може да срине хоствалото я приложение;
- консумира значително време, ресурс, памет, дисково пространство;
- изпълнява потенциално опасни системни повиквания, включително стартиране на файлове

контролът може да се маркира като SAFE.

Добре е в контрола да реализирате IObjectSafety. Той позволява контейнерът (напр. IE) да разпита вашия обект за safety

Добре е да ограничите възможността вашия контрол да може да се вгражда в script само от определен (доверен) домейн.
(Напр. да се ползва само от web страници на myshopping.com домейн.)

това може да се направи:
- с реализация на IObjectWithSite , който има метод SetSite(), викан от контейнера (напр. IE го вика) и връща указател към IUnknown на контейнера.
- Чрез него можете да направите нужния анализ.

```
pUnk-->QueryInterface(IID_IServiceProvider, &pSP);
pSP-->QueryService(IID_IWebBrouser2, &pWB);
pWB-->getLocationURL(bstrURL);
```

- определяте в кода си дали bstrURL е името на доверения URL. Тук има следната хитрост: хакерът може да е маскирал името на сървъра си : www.myshopping.com.foo.com!!!
В Wininet.dll има експортирана функция InternetCrackUrl(), която взима правилния host name от URL.

Достъпна е библиотека SiteLock

http://msdn.microsoft.com/downloads/samples/internet/components/sitelock/default.asp

разработена през 2003, която позволява създаване на сигурни ActiveX контроли за Web sites.
Там има функции за ограничаване достъпа до контрола само до списък от домейни, което затруднява хакери да ползват контрола за вредни цели.
Има функции, позволяващи контролът да изглежда и се държи по различен начин в различни домейни и др.