New Features, New Holes: Issues with Inherited Rights in Novell Directory Services

By Steven H. Parker

The old saying. "If it ain't broke, don't fix it", embraces a well-known truth of systems management. Change, even when well managed, carries risk. This is especially true in the world of technology. In an essay on software complexity and security, Bruce Schneier wrote, "Digital technology has been an unending series of innovations, unintended consequences, and surprises…".[1] As systems evolve, the process of change invariably has unforeseen ramifications, and these can easily lead to security vulnerabilities.

On September 7, 2000, an advisory[2] was posted on the Bugtraq mailing list that described a potential problem with newer versions of Novell's Directory Services [NDS]. *[NOTE: In my role as a senior consultant with FogHorn Security, I was the original contributor of this advisory]*. The issue was, "A design weakness in NDS as shipped with Novell v5.0 and later can allow certain users to bypass IRF's, and gain escalation of privileges." Novell added functionality to NDS relative to inherited rights without fully considering the potential impact.

Context

To understand the threat, one must first have some knowledge of the design of NDS security. This paper will assume a basic familiarity with the concept of directories, and Novell's NDS in particular. I will explain the terminology that will be used.

I will refer to a single directory as a "tree". The [root] object will be considered to be the "top", and objects further from the root will be termed "lower" in the tree. For instance, .user.dept.org will be deemed lower in the tree than .admin. Objects that can hold other objects, Organizational Units, for instance, will be called "containers".

All objects in an NDS directory have properties associated with them. One very important property common to all objects is the "Object Trustees ACL" property [ACL]. The ACL contains a list of trustees, and specifies the access that each trustee has been granted to the object itself, and to the object's properties. It also contains a list of Inherited Rights Filters [IRFs], which will be explained later. There are three classes of rights which can be assigned to a trustee: "Object"

[1] http://www.counterpane.com/crypto-gram-0003.html#SoftwareComplexityandSecurity

[2] http://www.foghornsecurity.com/advisories/20000907

rights, which apply to the object itself, “All Properties” rights, which apply to all properties of the object, and “Selected Properties” rights, which apply to specific individual properties.

NDS allows for the inheritance of rights. This means if rights are granted to a container, they can also be made to apply, via inheritance, to all objects within the container, and its subcontainers. This is a very powerful feature which eliminates the need to set and maintain ACLs on every object, and makes the granting of rights to large sections of a tree simple and straightforward.

These inherited rights can be blocked in two ways. The most common method is the use of an Inherited Rights Filter [IRF]. An IRF is essentially a roadblock through which only specified rights are allowed to pass. For instance, if an IRF was set to allow Browse object rights only, no object rights other than Browse would be inherited at or below the level of the IRF regardless of the trustee to which they were assigned.

Another method for blocking inheritance of rights is to overide them with an explicit rights assigment. This method is more limited in that it blocks rights for specific trustees only. For instance, if .bob.it.acme is given BCDR [Browse, Create, Delete, Rename] rights to the .acme container, he would normally inherit those rights for the .hr.acme container as well. If he was then granted B [Browse] rights to the .hr.acme container, that explicit assignment would override the rights granted at .acme, and his effective rights to .hr.acme would be B only.

Calculating Effective Rights

One last concept that we need to address is that of effective rights. Let’s take a detailed look at how effective rights are calculated by NDS. The full algorithm used to calculate effective rights can be found in Novell’s documentation available on its website[3]. I’ll repeat it here with comments in italics.

1. Determine which objects the user is security equivalent to.

2. For the user and each object that the user is security equivalent to, determine its effective rights as follows.

    a) Starting at the root of the tree, check whether the user (or object) is assigned any inheritable rights. If so, use these rights as the initial effective rights.

    b) Descend a level in the tree along the branch containing the target object or property.

    c) Remove any rights that are blocked by inherited rights filters at this level.
    IMPORTANT! Note that a filter on All Property rights will NOT affect Selected Property rights.

[3] http://www.novell.com/documentation/lg/nds8/docui/index.html#../usnds/objs_enu/data/hy8ds371.html

d) Check whether the user (or object) is assigned any inheritable rights at this level. If so, apply the assigned rights by adding any that correspond to types not yet in the effective rights and by replacing any that correspond to types already in the effective rights.
In other words, whatever rights are assigned at this level become the effective rights. Note that explicit rights which are not set as inheritable, do not affect the rights calculations unless they apply to the target object itself.

e) Check whether the current level is the target object (or the object containing the target property). If so, continue with the next substep. If not, return to Substep b.

f) Check whether the user (or object) is assigned any noninheritable rights at this level. If so, apply the assigned rights by adding any that correspond to types not yet in the effective rights and by replacing any that correspond to types already in the effective rights.
By "types", Novell is referring to Object, All Property, and Selected Property rights.

3. Add up the effective rights of the user and the objects that the user is security equivalent to.

4. Add any rights that are implied by the other rights.
It is not until this final step that the three types of rights are combined. For example, supervisor object rights implies S rights to all properties as well, regardless of any explicit assignments. Also, Selected Property rights override All Property rights for the specified property, even if the Selected Property right is more restrictive. The only exception is that S rights cannot be overridden in this way.

The Change

In earlier versions of NDS, only Object rights and All Property rights could be inherited. With the release of Netware 5.0, Novell added the ability to designate Selected Property rights as inheritable. This is a welcome and beneficial change, however, it does present new considerations, and can actually open holes in NDS security during an upgrade from Netware 4.x to 5.x.

The Security Concern

Allowing Selected Property rights to be inherited created a second path for object property rights to flow down the tree. This was destined to create problems. A good illustration is a bug found in IIS, and posted to BUGTRAQ on August 16, 1999[4]. Apparently, IIS could be configured to block access to certain files or directories based on their name. If those files could be referenced using a different name, an 8.3 short filename for instance, the access controls could be bypassed. Similarly, NDS now allows rights to object properties to be inherited in two ways. Unfortunately, they do not provide a convenient method of blocking both, thus inviting problems.

To better understand the potential problems, let's consider some of the situations where the blocking of inherited rights may be necessary. A good example can

[4] http://www.securityfocus.com/bid/582

be drawn from Novell's own website. In early 2000, Novell and Microsoft had a very public disagreement[5] over an alleged bug in Microsoft's Active Directory. At issue, was the inability to restrict a domain administrator's access to designated Organizational Units within the domain. Novell was very proud of the ability of NDS to completely block administrative rights to portions of their directory:

“*In Novell's NDS there is a feature known as an "Inherited Rights Filter (IRF)". The IRF can block administrative rights from affecting sensitive departments, such as payroll or human resources.* “

The procedure for doing so is described in Novell TID# 10011973[6], in answer to the question, “*How do I create an admin for a container that cuts off the main admin?*”. The process includes the setting of an IRF on the container to which the high level admin should not have access. Novell states, “Set the IRF … to [B] object rights and [R] property rights.” They make no mention of setting IRFs on Selected Property rights, even though the document was updated as recently as November 20, 2000.

Apparently, even Novell has missed this issue. The instructions they give do NOT properly protect the sub-container, since the IRF as specified allows Selected Property rights to be inherited. An admin with rights to a higher level container could grant themselves an inheritable Write [W] right to the ACL property in the higher level container. That right would be inherited by the “protected” container, allowing the admin to modify its ACL property, granting himself full control of that container.

This particular situation should be rare. In most situations, high level administrators will be trusted to administer all areas of the tree, and no attempt to block their rights via IRFs will be made. However, there are other situations that must be considered.

The following is from Novell TID 10015319, “Implementing Password Management in NW5”[7].

“Large corporations often employ a Helpdesk support team to handle user questions and issues. The network administrator would like to be able to grant the minimum set of NDS rights to the NDS User/Group/Container object for the

[5] http://www.novell.com/competitive/nds/security.html

[6] http://support.novell.com/cgi-bin/search/search.pl?database_name=kb&type=HTML&docid=%03%25F184921%3a978487357%3a%20%28%20TID%2010011973%20%29%20%20%07%01%00&byte_count=29891

[7] http://support.novell.com/cgi-bin/search/search.pl?database_name=kb&type=HTML&docid=%03%2cF185321%3a978487859%3a%20%28%20password%20management%20%29%20%20%07%01%00&byte_count=11157

Helpdesk users so that they can handle the common user issues of enabling an account or changing a password."

Now that Selected Property rights can be made inheritable, it is easy to grant rights to a subset of an object's properties. The Novell document details which properties to which the support staff must have rights in order to unlock accounts and reset passwords. It is both tempting and convenient to grant those rights very high in the tree and let them flow down to all user objects via inheritance. However, if any containers are protected by IRFs, and those IRFs don't explicitly block Selected Property rights, support staff may have the ability to reset passwords in areas they were never intended to.

Another option for blocking these types of limited rights from applying to a container would be to create an explicit rights assignment. For instance, if a group, .passwd_reset, is granted limited property rights at the root of the tree, and you wish to block those rights on the container .vips.exec, you could create an explicit rights assignment for the .passwd_reset group on that container, specifying more limited rights.

Be careful! Just as with IRFs, if an explicit assignment is not done properly, the desired effect may not be achieved. In the password management example, the explicit assignment would need to specify Selected Property rights for each property. Do not make the mistake of thinking that an explicit assignment of Read [R] only for All Property rights will limit Selected Property rights. It will NOT!

There are undoubtedly many situations where inheritance can get an administrator in trouble if not fully understood and properly implemented. Consider what would happen if the schema was extended. The new properties will not be protected by any previously set IRFs. Doing so would be impossible, since you can't set an IRF for a property until it actually exists in the schema. This is a design weakness within NDS. Clearly, the ability to block ALL Selected Property rights with a single IRF is needed.

Another important scenario to consider is an upgrade from Netware 4.X to 5.X. During the upgrade process, any existing IRFs put in place to block supervisor rights WILL be rendered ineffective by the functional changes relative to inheritance. Why? If someone has supervisor rights, they WILL have the ability to modify ACLs on containers above the IRF, and thus, will be able to use Selected Property rights to bypass the existing IRF. This is a bigger issue than it seems. Remember, Novell's own documentation and training materials outline situations where blocking the S right is necessary. Nevertheless, Novell has failed to alert anyone to the danger posed by the new functionality in NDS, nor have they provided tools to update existing IRFs.

Although Novell has clearly missed this issue, it is not a complete disaster. The most serious exploits require an attacker to have the ability to change ACLs on containers. In a properly configured NDS environment, few people will have those rights, and those that do will be trusted anyway. The greater danger is the potential for misconfiguration. As was asserted in the FogHorn Security advisory, this is not a bug in NDS. However, it is a design issue that must be carefully considered by anyone responsible for security in an NDS environment.

Bibliography

Schneier, Bruce. “Cryptogram.” March 15, 2000. URL: http://www.counterpane.com/crypto-gram-0003.html#SoftwareComplexityandSecurity (19 Feb. 2001)

Parker, Steven. “Bypassing Inherited Rights Filters in Novell Directory Services.” September 7, 2000. URL: http://www.foghornsecurity.com/advisories/20000907 (19 Feb. 2001)

“How NDS Calculates Effective Rights.” NDS 8 Online Documentation. 21 April. 1999 URL: http://www.novell.com/documentation/lg/nds8/docui/index.html#../usnds/objs_enu/data/hy8ds371.html (19 Feb. 2001)

“Microsoft IIS And PWS 8.3 Directory Name Vulnerability.” 10 Nov. 2000 URL: http://www.securityfocus.com/bid/582 (19 Feb. 2001)

“The NDS Advantage: AD Security.” 18 Feb. 2000 URL: http://www.novell.com/competitive/nds/security.html (19 Feb. 2001)

“10011973: File System Directory and File Rights.” 20 Nov. 2000 URL: http://support.novell.com/cgi-bin/search/search.pl?database_name=kb&type=HTML&docid=%03%25F184921%3a978487357%3a%20%28%20TID%2010011973%20%29%20%20%07%01%00&byte_count=29891 (19 Feb 2001)

“10015319: Implementing Password Management NW5.” 1 Dec. 2000 URL: http://support.novell.com/cgi-bin/search/search.pl?database_name=kb&type=HTML&docid=%03%2cF185321%3a978487859%3a%20%28%20password%20management%20%29%20%20%07%01%00&byte_count=11157 (19 Feb. 2001)