

An Audit of Active Directory Security, Part 5

Aaron Sullivan 2002-01-15

This is the fifth and final installment in a five-part series on auditing Active Directory security. The [first](#) article in the series offered a brief introductory overview of Active Directory. In the [second](#) installment we examined some of the security implications of the AD's default settings. The [third](#) article looked at LDAP, SASL and Kerberos in the context of AD security. The [fourth](#) part looked at some potential security concerns related to the Configuration Naming Context in AD. This article will examine some issues surrounding the multi-master replication scheme.

The basis for this article begins with the following question:

if two separate nodes on a directory-enabled network commit actions on the same object in the directory at (approximately) the same time, which node's actions will be considered final, and how will the details be replicated accurately?

The purpose of this question is to lay the groundwork for a discussion of the theoretical attack mentioned in the title of this article. Hence, we will first establish a hypothetical scenario that illustrates how and why such an attack may be possible in a generalized multi-master directory service, after which we will jump into a discussion of how Active Directory handles the conditions mentioned in the hypothetical scenario. Upon completion of that discussion, we will discuss how to circumvent Active Directory's controls and make the hypothetical scenario (and worse) happen anyway.

A Hypothetical Situation

Suppose my fellow system administrator, Bob, and I have both been given a long list of changes that we need to make to our organization's Active Directory. Suppose that neither Bob nor I are aware that the other has a list of changes to make. Bob and I have separately planned to make these changes at the same interval in time during the work-day. Finally, suppose that Bob's list and my list differ slightly on most changes to be made (say, Bob has version 1.0 of the list and I have version 1.1).

When Bob and I make our respective changes, what impacts could they have on the directory? Assuming that the directory has no controls in place to deal with activity of this sort, any number of things could happen. The directory could end up with two instances of the "same"

object, a corrupted object, and or an object with some of Bob's changes and some of my changes in effect. Any of these could result in user objects having an amalgamation of rights and/or multiple passwords (among other things). They could also result in critical portions of the directory (whether critical to the system or to the user base, it doesn't matter) getting changed and replicated in such a way to cause serious problems.

Replication could be an especially big problem if the changes were made on separate servers possessed of the same replica (as is commonly the case in a multi-master scheme). Ever play the rumor game in which one person whispers something in the ear of another person, who then whispers it to the next person down the line of people, so that the end message sounds nothing like the original? This is much like what could happen in replication. However, the replication cycle would probably be endless (as opposed to the rumor game) as there would never be a "finalized copy."

This could create a number of problems on a directory-enabled network. For instance, the directory could crash/shutdown, effectively causing "network outages" (especially if DNS is integrated into the directory). Furthermore, the various master-nodes (replica-carrying servers) could get into contention over which has the most recent object, forcing constant replication, chewing up CPU clock-cycles and bandwidth.

While one might say that the biggest problem underlying this scenario is simply the fact that Bob and I are not communicating adequately to properly coordinate our efforts. In this case, I am giving it serious consideration because it represents a significantly common situation and, therefore, should be addressed.

Managing Coincident Changes with Active Directory

Active Directory takes a very simple approach to tackling this problem. For the purpose of describing Active Directory's solution to this problem, we'll consider that the Active Directory network we are talking about here constitutes *one* site. From here, we'll note what Active Directory does at each step of the process of Bob and I implementing our respective changes.

When I implement my changes at Server A and Bob makes his changes at Server B, each object that we move, add, and/or modify carries a number called an Updated Sequence Number (USN), which is incremented on each server. The same object, on two different servers, carries a unique USN. In addition to that, each server's replica of the directory carries a unique USN

that increases in increments every time a change is made to that replica's contents. The servers keep track of their partner's replica's USN and the USN(s) of each object/attribute on that partner in order to keep track of change.

The general effect that this has is to improve the stability of the overall directory state in regards to data integrity in the replica. It also means that consistency between replicas is maintained more loosely and that automatic (unforced) synchronization is generally much less timely than one might expect. The exception to this consistency rule is if the object in the directory is labeled with an attribute called `isCriticalSystemObject`, in which case, synchronization is forced.

So when Bob and I make our changes to the same data on separate servers at roughly the same time on the same objects and/or attributes, the server with the highest change in USN for each object is treated as the most recent copy for each object and attribute. In other words, the USNs are compared before replication and the highest USN wins.

AD replication geeks can poke holes in this example for incompleteness, but, as stated previously, it serves as a basic example for arguing the theoretical attack. For those who would like a more comprehensive explanation, although by no means complete, check out the following URLs:

http://support.microsoft.com/servicedesks/webcasts/so_projects/so12/soblurb12.asp

http://support.microsoft.com/servicedesks/webcasts/so_projects/so13/soblurb13.asp

http://support.microsoft.com/servicedesks/webcasts/so_projects/so14/soblurb14.asp

http://support.microsoft.com/servicedesks/webcasts/so_projects/so15/soblurb15.asp

The Theory Behind the Potential Attack

But how does Active Directory behave when the conditions created by Bob and I become exceptional (i.e., are widely expanded and implemented at much more drastic speeds than usual)? This is where the theory comes into play, and the simplest way to apply the theory is to start changing USNs at random, to random objects, and at rapid speeds. In other words, to accelerate the application of the theory, start changing USNs, objects, and attributes at random, while always adding the option "TRUE" to the attribute `isCriticalSystemObject`, to all objects in the directory.

Now, before we start talking about the effects of this on Active Directory's behavior, and the global effect that such behavior will have, let's talk about what is required to commit such behavior. First off, rights that are higher than that of Administrator are required. In order to change USNs at will, Local System authority is required. Otherwise, in order to put the theory in effect, one would have to come up with an engine that would still require at least administrative rights to operate, and would randomly and rapidly request random changes to the objects in the directory, thus effecting changes to the USNs of these objects. With administrator rights, one can still utilize the accelerating effect of "isCriticalSystemObject: TRUE." So, having completed all the aforementioned items, stop and take a breath for a moment...

Now, toss in a replication cycle or two (if the system even makes it past one cycle)...

By now, I would expect that you realize what this could do to a large, massively AD-enabled organization's network: highly distributed, effective, denial of service to both the network itself and the respective Multi-Master nodes (and all components of desktop operation that require the directory in order to function). By my best estimates, such a denial of service would also be extremely difficult to stop, and even more difficult to repair the damages.

Prevention of the Potential Attack

Readers who are inclined to consider the effects of this will quickly see that there are a number of ways to spin and or enhance this theory's intent. That much is easy, but how does one attempt to prevent such a thing from happening?

I assume that in order to have the time, permissions, and malice to do this, one would have to be a system administrator employee/ex-employee of an organization that owns a sizeable AD enabled network. So the first step in preventing this could simply be considered good, secure, careful internal security administration. However, if all people and all organizations were completely effective at such a thing, distributed denial of service would not be the topic of discussion that it is today.

The use of ingress and egress filters to monitor rates of change and change requests in the directory might also be effective at preventing bandwidth blackouts, but I don't believe that such a system (the way I imagine most think of ingress/egress monitors) would prevent a more slow and generalized data corruption that would occur.

Having reading this, some will ask: "Why waste your time with such a theory when you could just as effectively cause a mass denial of service through other non-electronically-based means (such as bombs, wire cutters, fires, etc)?" The explanation is simple and threefold. First of all, I would not discount the possibility of something like this happening, as there is clear record of similar things that have happened in the past (like the chaos that the RemoteExplorer virus/worm wrought, or the script that some disgruntled sysadmin wrote at Company XYZ that went through and randomly deleted files over time). Secondly, discussions of hypothetical situations like this raise awareness of issues that need to be addressed, like the general state of security with Active Directory, small parts of which have already been brought to light both within and without this series of articles. Thirdly, I'm speculating here, but I'm pretty sure I'm not the only person who has considered such vulnerabilities.

In Conclusion

To close this series, I hope that this series of articles have raised your level of Active Directory enlightenment and awareness. I sincerely enjoyed the time that I have had researching and writing it all. Despite the criticisms leveled here and elsewhere against AD, and despite its inherent problems, I still hold the opinion that it is a fantastic technology because of its relative ease of use, scalability, and integration. However, as has been mentioned before by others, these things all come with a price greater than the one you pay out of your wallet.

Relevant Links

[An Audit of Active Directory Security, Part One: An Overview of Active Directory and Security](#)

Aaron Sullivan , SecurityFocus

[An Audit of Active Directory Security, Part Two: Understanding the Security Implications of Active Directory Default Settings](#)

Aaron Sullivan , SecurityFocus

[An Audit of Active Directory Security, Part Three: Understanding LDAP, SASL, and Kerberos in the Context of AD Security](#)

Aaron Sullivan , SecurityFocus

[An Audit of Active Directory Security, Part Four: Keys to the Kingdom - the Configuration Naming Context](#)

Aaron Sullivan , SecurityFocus

[Privacy Statement](#)

Copyright 2006, SecurityFocus