Global Information Assurance Certification Paper

Kerberos and Access Token Limitations

GIAC (GSEC) Gold Certification

Author: Josh Sprenger, josh.sprenger@yahoo.com
Advisor: Egan Hadsell

Accepted: July 22, 2011

Abstract

Kerberos has been the default authentication protocol for Windows since XP/2000. Although the protocol enjoys many benefits over its predecessors, it does have some weaknesses. One unintended weakness of Kerberos is the ability of the Kerberos token size to grow to the point where Denial of Service (DoS) issues arise. This is especially prevalent in large enterprises where during the 10 years that Kerberos has been the primary Windows protocol, some users have found their accounts to be members of several hundred groups. The result of this scenario includes inability to use important company resources such as Exchange Servers and the ability to authenticate to web sites. Additionally, this weakness can be used maliciously to cause widespread DoS throughout an enterprise.

1. Introduction

Kerberos replaced NTLM as the default authentication protocol for Microsoft Windows' client operating system with Windows XP and with its server operating system Windows Server 2000. Although the MIT-developed authentication protocol enjoys many benefits over its predecessors, it does have some drawbacks. One unintentional limitation of Kerberos is the ability of the Kerberos token size to grow to the point where Denial of Service (DoS) issues arise. This is can be a common occurrence in larger enterprises where during the 10 years that Kerberos has been the primary Windows protocol, some users over time have their accounts added to become members of several hundred groups. This type of situation can have several significant implications that range from the inability to log on to any network resource including one's own workstation to entire Exchange Servers becoming inoperative. Additionally, this weakness can be used maliciously to cause widespread DoS throughout an enterprise.

With the Windows operating systems dominating 90% of the desktop market and 73% of the Server market, it is important to investigate this topic as so many depend on Windows operating systems to run critical parts of their IT infrastructure (Foley, 2010). It is essential they be aware of how these situations can arise and how they can be monitored and controlled. This paper will first give an overview of Kerberos authentication and Windows groups. It will then explore the Kerberos access token limitation including symptoms of the issue, how it can be exploited, and then cover best practices for prevention, maintenance, and resolving token limitation issues.

2. Kerberos Authentication

2.1. Kerberos Authentication

All versions of Windows beginning with Windows 2000 use Kerberos as the default authentication method for Windows authentication. Although older forms of authentication such as NTLM can be configured to be accepted for backward compatibility with older operating systems and applications, Kerberos is the standard (De Clercq, 2004)

A Kerberos authentication scenario has three major parts: a client user, the Kerberos Key Distribution Center (KDC), and the resource the user wishes to access; in this example a server. Windows has the KDC configured as part of a domain controller. The KDC performs two major functions: the Ticket-Granting Service (TGS) and the Authentication Service (AS).

First, the client will negotiate access by logging on to the network by giving their username and password which is confirmed by the AS function of the KDC on the domain controller. The KDC accesses the user account information from the domain controller to authenticate the user. It also acquires the list of the user's SIDS from the local Active Directory domain. The global catalog server is also contacted by the KDC to obtain any universal groups the user is a member of or is a member of through membership in another group. The TGT's authorization data field is populated by the KDC with the user's SIDs and the SIDs from any valid universal groups. If the user authentication is successful, the AS replies with two items: an encrypted session key that has been hashed with the user's password and a TGT encrypted with a key only the TGS can decrypt. The TGT not only includes the authorization data of all the SIDs, but also parameters for time to live, and a session key for communicating with the client. The TGT is only usable for the local Active Directory forest and is valid for 10 hours by default, although this can be renewed during the user's session without necessitating the re-entry of the user password. The local machine caches the TGT to be used to request sessions to access other services throughout the Active Directory forest and other trusted domains.

When accessing a service, in this example on a server, the user gives the TGT to the TGS part of the KDC which then authenticates the TGT and generates a session key and service ticket for both the user and server to use. The KDC copies all of the SIDs that are part of the contents of the TGT's authorization data field to the service ticket's authorization data field. This service ticket contains the user's access token.

The client can read part of the session key with its own TGS session key received previously from the AS. The client will then give the server piece of the TGS reply to the target server. The server decrypts the data from the TGS using its own key from the

KDC. The client is then authenticated to the server with the service ticket to establish a session between the client and server. Once authentication has been established the session is good until the ticket's lifetime is exceeded, at which point the ticket must be renewed. (Garman, 2003).

3. Windows Groups

While the previous section explained how an access token is created within the Kerberos authentication process, to really understand access tokens the impacts of the different types of Windows groups, group nesting, and group and user domain location must be explored. This section will analyze the local groups, domain local groups, global groups, universal groups, distribution groups, and other Active Directory domain factors that affect the impact of an access token.

3.1. Local Groups

A local group is a group that is specific to an individual computer. A local Windows computer can contain user accounts that are entirely unconnected from accounts that are members of the domain that the computer is a part of. This is called a local user account, and it is only accessible from the local computer on which it resides. Additionally, a local user account only can exist on a Windows workstations or member servers. Windows domain controllers do not permit the creation of local user accounts. Similarly, a local group is simply a group that is specific to a single member server or workstation. A local group is primarily used to manage local user accounts. For example, there is a local Administrators group by default on every workstation and server which designates which users are administrators over the local machine. Even though local groups' primary use is to secure local resources on a computer, the group's membership doesn't need to be limited to local users. Although a local group normally contains local users, they can also contain domain users and groups. Besides local users, a local group can contain domain users, global groups, domain local groups, and universal groups.

As far as local groups and token size, local groups are only added to a user's access token when the user authenticates to the local computer. The local group SIDs will not be part of the user's access token when accessing any other domain resources outside of the local computer. Therefore, it is unlikely for local groups to have a significant impact on access token issues (Johansson, 2008).

3.2. Domain Local Groups

Domain controllers do not have a local account database and therefore no local groups. This is why domain local groups exist. Even though there are no local users or local groups on domain controllers, they still need to manage local resources. This is what domain local groups are for.

When a Windows server operation system is installed onto a computer, the system usually begins as either a standalone server or as a member server. In both cases, local user accounts and local groups are produced during the installation. If the computer is promoted to a domain controller by running the DCPROMO process, the local groups and local user accounts are transformed into domain local groups and domain user accounts. Once a computer is made a domain controller it shares a common user account and group database with all other domain controllers in the domain. If a user is added to a domain group on one domain controller, it will be a member of that group on all domain controllers in the domain.

The main restriction with domain local groups is that they are restricted to manage access to resources only in the domain in which they reside. This can be a disadvantage compared to a global group or a universal group as they can be used in every domain within a forest. Domain local groups can, however, have users, global groups, and universal groups from other domains as members (Johansson, 2008).

The advantage of domain local groups from an access token standpoint is that since their scope is limited to the domain in which they reside they are only added to a user's token when a user authenticates to a resource within the same domain as the domain local group. For example, assume that John belongs to the domain "North_America_Domain" and is a member of a domain local group North_America_Domain\Canada Users. John is also a member of a domain local group

Europe_Domain\Canada Users. When John logs on to a computer that belongs to North_America_Domain (for example, North_America_Domain\WorkstationA), a token is generated for John on the computer, and the token contains, in addition to all the universal and global group memberships, the SID for North_America_Domain\Canada Users. It will not contain the SID for Europe_Domain\Canada Users because the computer where John logged on (North_America_Domain\Workstation1) belongs to North_America_Domain. Likewise, when John logs on to a computer that belongs to Europe_Domain (for example, Europe_Domain\WorkstationA), a token is created for John on the computer, and the token contains, in addition to all the universal and global group memberships, the SID for Europe_Domain\Canada Users; it will not contain the SID for North_America_Domain\Canada Users because the computer where John logged on (Europe_Domain\WorkstationA) belongs to Europe_Domain. However, when John logs on to a computer that belongs to South_America_Domain (for example, South_America_Domain\Workstation1), a token is created for John on the logon computer that contains all universal and global group memberships for John's user account. Neither the SID for North_America_Domain\Canada Users nor the SID for Europe_Domain\Canada Users is contained in the token because the domain local groups that John is a member of are in a different domain than the computer where John logged on (South_America_Domain\Workstation1). Conversely, if John were a member of some domain local group that belongs to South_America_Domain (for example, South_America_Domain\Canada Users), the token that is generated for John on the computer would include, in addition to all the universal and global group memberships, the SID for South_America_Domain\Canada Users.

The use of domain local groups can have a dramatic effect on the size of access tokens within an Active Directory forest. A good way to reduce token size is to put resources across different domains and use domain local groups to grant permissions to those resources. This is significant from an access token perspective as one option for reducing access token size is to place users in one domain and server resources in a separate domain. For example, if Exchange page pool memory (which will be explained in detail in a later section) is an issue for a company, placing the Exchange servers in a separate domain would remove all the SIDs of all domain local groups from the user's

own domain from all user access tokens. If the company heavily utilized domain local groups in the user's domain this could be a substantial reduction in token size and the amount of page pool memory taken because of token size issues.

3.3. Global Groups

Global groups are the default group created when creating a group within Active Directory and is thereby the most used group type. Microsoft recommends using global groups to nest Active Directory user accounts. Global groups can be placed inside of each other, but can only have Active Directory resource as members. One global group can be made a member of another global group, a domain local group, or a universal group. A limitation of global groups is they can only have users and groups that exist within the same domain as members. Unlike domain local groups, domain users and domain groups from other domains cannot be nested in a global group in a different domain. Local user accounts or local groups cannot be placed into a global group. However, a global group can be added to a local group. This nesting is a common way of giving domain users permissions to local computer resources. Similarly, placing global groups within domain local groups is the Microsoft best practice for giving domain users permissions to directory resources (Johansson, 2008).

To continue the example above with Global Groups, assume that instead of North_America_Domain\Canada Users and Europe_Domain\Canada Users being domain local groups they are instead global groups. Since John belongs to the domain "North_America_Domain" he would only be able to be a member of the global group North_America_Domain\Canada Users because Global Groups can only contain members from their own domain. However, the North_America_Domain\Canada Users group could be used to secure resources within the Europe_Domain because unlike domain local groups, global groups can manage access to resources outside their domain. Global groups have a major impact on access token size. No matter where a user authenticates, all of the user's global groups will be included in the user's access token. This is different from local groups which are only included in a user's token on a local computer and domain local groups which are only included in a user's token within the same domain as the domain local group.

3.4. Universal Groups

Universal groups are the most flexible Windows group in that they can contain users and groups from any domain in the forest and can be used to secure resources in any domain in the forest. With so many advantages over the other group types one might wonder why not use only universal groups.

There are disadvantages of universal groups. Like global groups, universal groups also add a SID to a user's access token no matter what domain the group or user reside. Those companies that have token size issues may want to consider other group types. There is another potentially harmful disadvantage that comes with using universal groups. Every domain has a global catalog which is a distributed data repository that contains a searchable, partial representation of every object in every domain in the forest. Universal groups as well as every member listed within the group are written to the global catalog server. As more and more universal groups are created the global catalog, which is replicated across the network to all other global catalog servers, can become large. Large global catalogs have been known to lead to network performance issues. Other types of groups do not place this type of load on the global catalog. Only global groups are listed in every global catalog, but unlike universal groups their membership is not included (Johansson, 2008).

Table 1 below shows the nesting limitations of the different types of security groups.

Group Type	Can Be Nested into Local	Can Be Nested into Domain Local	Can Be Nested into Global	Can Be Nested into Universal
Local	No	No	No	No
Domain Local	Yes	Yes, if in the same domain	No	No
Global	Yes	Yes	Yes, if in the same domain	Yes
Universal	Yes	Yes	Yes	Yes

Table 1

3.5. Distribution Groups

Local, domain local, global, and universal groups are all classified by Windows as Security groups. They are all used to manage access to resources. Windows does have an additional type of group. Distribution groups are for use with e-mail applications like Microsoft Exchange Server. They are used to send email to a group of users. Distribution groups are not security-enabled, so they cannot be used to give or deny access to resources. Distribution group memberships aren't distributed with access tokens. Because of this, distribution group memberships don't normally affect the token size. However, when distribution groups are used to secure a resource, they convert to security groups when the resource is accessed for the first time and then do affect token size. This can have a significant impact on access token size. A group that an IT organization expects not to cause token issues can surprise them when they find out areas within the organization are using them inappropriately causing them to transform to security groups (Johansson, 2008).

Table 2 below details when a SID is added to a user's access token.

Group Type	SID added to token on local computer	SID added to token within any computer in same domain	SID added to token within any computer in different domain
Local	Yes	No	No
Domain Local	Yes	Yes	No
Global	Yes	Yes	Yes
Universal	Yes	Yes	Yes
Distribution	No	No	No

Table 2

3.6. Active Directory Functional Level

The effect a group will have on the size of an access token will also depend on the functional level of the Active Directory domain. When all domain controllers within a the domain are upgraded to the newest operating system version the functional level of the domain can be raised to the most current level which activates the newest

functionality within the domain. Mixed mode domains contain domain controllers running a mix of newer and older operating systems. Native mode domains have all domain controllers running the latest operating system version (Microsoft, 2010).

If the user account domain is in mixed mode, then you can't nest global groups in other global groups. But in native mode you can nest, and each nested group and its sub-nested sub-groups will be added to the token. Nesting does not help you reduce token size, but it can make it harder for you to figure out how many groups each user really belongs to.

Native mode impacts the way universal groups are treated within an access token. Mixed mode does not include universal groups within a user's access token. Native mode will add a universal group SID to the access token. It will also add any groups nested within the universal group to the user's access token (Microsoft, 2007).

3.7. Other Group and Token Considerations

Different groups take different amounts of space within an access token. Domain Local, Global, and Universal groups usually require 44 bytes of space each within a token. This estimate considers the SID length to be 28 bytes for groups, a length that has been the same with older versions of Windows and has remained consistent through current versions of the OS. Additional memory is allocated to attribute data linked to each SID which takes up another 16 bytes of space. Furthermore, in addition to the space for group membership SIDs, there is approximately 500 bytes of overhead in each access token.

IT administrators and other areas of management tend to belong to the most groups and therefore have the largest access tokens. These individuals normally have access to more resources within a company. As a best practice access to resources within a Windows environment is granted through group membership so it makes sense that these users with special privileges would be populated in more groups than the average user.

The existence of SID History can have a large influence on the size of access tokens within a domain. When accounts are migrated from one domain to another, the groups those accounts belong to must also be migrated to retain the same access for the

users. When the groups are migrated new SIDs are created for them for the new domain, but the old SIDs are also migrated. The old SIDs are referred to as SID History. When the user authenticates both the SID History and the new SID will be loaded in their access token. When that user attempts to access a resource, both the SID History and the new SIDs will be evaluated. If the resource is in the new domain, having the new SID in the access token will result in access for the user. If the resource is in the old domain, the inclusion of SID History within the token will permit access. Preserving SID History after a user migration will increase the size of an access token and in some cases double it (Microsoft, 2007).

4. Kerberos Access Token Limitation

4.1. The Kerberos Access Token Issue

As described above, Windows creates an access token for each user logon. The access token specifies the user account ID and the security identifiers (SIDs) of all the security groups to which the user belongs. The token's size grows as the user is added to additional security groups, which are represented by an additional SID within the token. If a user is a member of a group either directly or by membership in another group, the SID for that group is added to the user's token. There is a built-in limit in that Microsoft states that the number of SIDs that can be contained in an access token is restricted to 1024.

The way Windows allocates memory for the token can further complicate access token issues. The default size of a user's access token is 4K. The exact amount of memory an additional group membership (and thus additional SID) contributes to the amount of memory within this initial 4K default access token varies slightly depending on the type of group – Global, Domain Local, Built-in, Local, or Universal. However, on average approximate 80 groups will completely fill up this 4K of access token memory. Once a user goes over this amount Windows does not increment the size of the token by the amount needed for each additional SID added. Instead Windows allocates another 4K of memory, thus doubling the size of the access token. Similarly, at approximately 160 total group memberships, a user's access token will again increase by 4K from 8K to 12K. This small amount of memory may not seem like a problem, but the

way different applications store these tokens and the limits that are defined around their storage, it can become a significant issue. This is especially true in large organizations with many thousands of users with the potential to be in several hundred groups (Lee, 2006)

4.2. Symptoms of Access Token Size Issues

The issues that arise from Kerberos access token limitations can be divided into two categories:

- Either TGT when the user initially logons on is too large
- The session tickets created (TGS traffic) when a user requests access to a network resource are too large

Either of these instances causes the Kerberos ticket size to be an issue. The first set of symptoms is when the number of SIDs in the TGT is too large. There is a built-in limit of the number of SIDs that can be contained in an access token. That limit is restricted to 1024. Although it allows this number to be surpassed within Active Directory, the consequences is the inability of the user to log in. Figure 1.1 shows an example of the error an end user will receive once they meet this criterion (Stevenson, 2005)

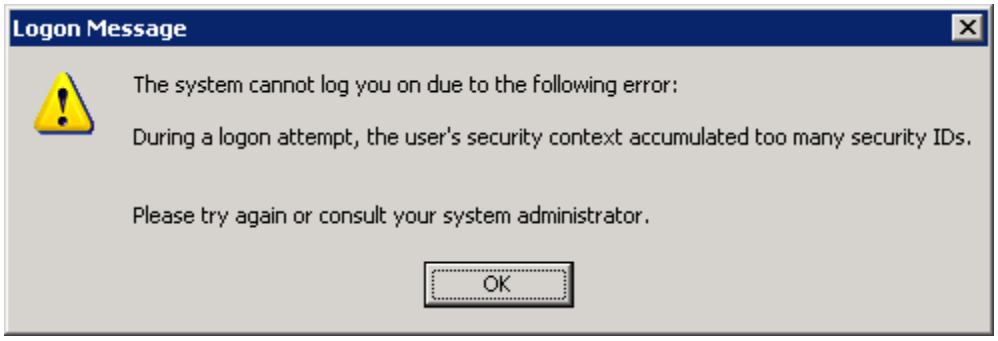

Figure 1

But token limitation can impact other applications as well. The session tickets created are the root of the problem for Exchange Server 2003. Exchange 2003 Servers use paged pool memory, the portion of shared system memory that can be paged to the

disk paging file, to cache security information for every user session opened against the server. A token is created for every user session generated by an e-mail client such as Microsoft Office Outlook. Depending on the Exchange-dependent applications the end-user has open; multiple sessions may be opened for each Exchange-dependent client application, causing multiple tokens to be cached on the server for each client. The paged pool memory footprint of each token is generally based on the number of security groups to which the user belongs. The more security groups that a user is a part of, the more paged pool memory will be consumed by the tokens associated with that user's sessions (Lee, 2006)

For example, suppose a large corporation assigns 10,000 users to each Exchange server. The corporation has not been proactive in managing group membership and the average total groups each user is a member of averages 350. Thus, their average token size is 20K. As a token is created for every user session generated by an Exchange-dependent client, each Exchange server will have four tokens for every user generated by the following client applications they use: Microsoft Office Outlook, the Global Address List, Public Folders, and Office Communicator. That 80K per user has now quadrupled to 64K per user. The amount of paged pool memory required for those user sessions amounts to an average of 800 MB. While this may not be a problem for newer 64-bit Windows Operating systems, those companies still running Windows Server 2003 32-bit Operating Systems on their Exchange servers are limited to 4 GB of memory. When taking into account the amount of memory the OS, Antivirus, and all other server resources require, a server can quickly run out of all available memory. This results in a denial of service for users attempting to connect when no free memory is available.

Access token issues can also cause denial of service issues with Internet Information Server 6.0 (IIS 6.0), Microsoft's web server. IIS 6.0 has a default http host header request limit of 16K which is exceeded by users with a Kerberos token size of anything over this amount. At approximately 4K of memory for every 80 groups, this limit is exceeded by all users that have more than 320 SIDs. Users with tokens of over 16K are denied when authenticating to the IIS 6.0 server.

Large token size has several additional consequences as well:

- The access token must be evaluated during the user logon process before logon can be completed. Because of this, as the number of security group memberships grows, the logon process takes longer.

- To determine access permissions, the Kerberos session ticket which contains a user's access token is sent to every computer that the user accesses. Because of this, the size of the access token has a direct impact on the network traffic load.

- Each machine receiving the Kerberos ticket has a predetermined size within the LSA\Kerberos\Parameters registry. It is called MaxTokenSize. This setting limits the creation of an Access Token based on Kerberos (12,000 bytes by default XP/Windows 2003). When this buffer limit is reached, it results in Kerberos-based access token creation issues.

- HTTP header size due to a large Kerberos session ticket being transmitted can also have a direct impact on the network traffic load. If a larger HTTP buffer size is required, network bandwidth will be impacted (Thomas, 2009)

4.3. Denial of Service

Access token issues within the Windows operating system open the potential for both inadvertent and intentional Denial of Service (DoS) scenarios. Through complex group nesting circumstances a company can unintentionally cause a DoS on a group of users or potentially to enterprise services in which the entire company depends upon. Similarly, the ease of creating such access token issues can allow for a discontented worker to use the common task of having a group created and populated with users to cripple the ability of certain users to access services or bring entire services offline.
An example of an actual inadvertent occurrence of a DoS within a large corporation took place when a large retail company created groups for each of their retail stores across the world. The purpose of the groups was to grant access to a specific file share created for each of the locations. They created two groups for each location by store number. They created a global group called "Store XXXX File Share GG" and a domain local group called "Store XXXX File Share DLG" for every location. "XXXX" was the actual store number, GG stood for Global Group and DLG stood for Domain Local Group. The

company followed the Microsoft recommended practice of populating Global Groups with users, nesting Global Groups within the Domain Local Groups, and granting access to resources to the Domain Local Groups. Management within each location was populated within each global group and each domain local group was granted read and write access to the specific file share for each store. The global group for each store number was then nested within the domain local group for that same store number, thus giving the store managers access to read and write files to the file share for their specific store number. This portion of the design did not contribute to access token issues as it only added two group SIDs, one for the store global group and one for the tore domain local group, to each user's access token.

However, it was determined that certain regional managers required access to their region's file shares and corporate managers required access to all file shares. The result of this decision was to create five regional groups: US East File Share GG, US Midwest File Share GG, US West File Share GG, Canada File Share GG, and Europe File Share GG. These Global groups were each nested within 60-80 of the "Store XXXX File Share GG" global groups. Additionally, a "Corporate File Share GG" global group was created for management within the corporate office to have access to all store file shares throughout the company. This group was nested within each of the five regional global groups.

This last set of changes had a damaging impact due to access token issues. The members of the regional global groups had an average of 140 group SIDs added to their tokens. This was due to being added to an average of 70 global groups which were each a member of 70 domain local groups. While not all uses within these groups had adverse experiences due to access token issues, many were already members of several hundred groups. Those that exceeded 320 groups now had access token sizes of 16K, the same number that is also the default limit of the IIS 6.0 host header size. Users above this threshold experienced the inability to authenticate too many web resources.

The corporate managers experienced the most deleterious impact from being added to the corporate file share group. This group was nested within the five regional file share global groups which each in turn were nested within an average of 75 individual store global groups. Due to the group nesting the members of this group had

406 SIDs added to their token for the global groups they were now a member of and an additional 400 SIDs added for the domain local groups they were now a member of through the group nesting. Adding 806 SIDs to their token brought the majority of these users over the 1024 SID limit. They lost the ability to logon to any domain resource. Alternatively, here is an example of a company employee taking advantage of the ease of creating an access token issue to cause a DoS. Creating groups at many large companies is a fairly common task that isn't normally scrutinized. Usually creating a group is a matter of filling out a form. If approval is required, usually it is around making sure the group name meets naming standards, not whether it will contribute to access token issues. The lack of scrutiny over group creation can allow for a discontented worker to use this common task to have a group created and populated with users to cripple the ability of certain users to access services or bring entire services offline.

At one particular company all users within their "Technology Services" group, a group of several thousand workers that managed all workstations, servers, networks, and applications throughout the enterprise had access to an online tool to create groups. The reason for allowing such broad access for group creation was that the act of creating the group didn't grant any access to the group. A resource owner would have to grant the group access to a resource before the membership within the group would allow any access. So, if for example an application owner created a new group using the tool, they would only be able to grant access to the application they owned unless they received approval from another resource owner for the group to have access to a resource the application owner didn't own.

One particular disgruntled worker used the ease of creating groups as a method for causing a DoS within his company. The worker began creating a few groups each day and nesting the Domain Users group within the new groups he created. The Domain Users group contains every user within a domain. At this particular company, that was over 100,000 users. Each time the "Domain Users" group was nested within a new group, it added one SID to every user in the organization.

The average number of SIDs of all users slowly increased within the entire company as more groups were created containing "Domain Users" as a member. The impact was first felt when users were unable to connect to the company's Exchange

Server to access their email. The Exchange Administrators began searching for answers as to why paged pool memory suddenly started to quickly grow over a matter of weeks at a steady rate. After researching the issue they found that every access token needs a specific quantity of Windows kernel memory and while the extent depends on a few different elements, group membership was the most significant as the magnitude of the token size increases in proportion to the total group memberships of an account.

The overall size of the access token when it is under 4K will take the exact same amount of kernel memory as the size of the access token. For example, if the access token is 3428 bytes in total size, it only takes that 3428 bytes of memory. However if a token goes any amount over 4K (4096 bytes), then it doubles the amount of memory it requires to 8K (8192 bytes). So if the access token actually only requires 5047 bytes of memory for all of the SIDs it contains, it will still take a full 8K of kernel memory. Similarly, if the access token were to require 9322 bytes of memory (or any amount over 8K), it would require a memory allocation of 12K (12288 bytes). As mentioned previously, users in less than 80 security groups typically will have access tokens under the 4K limit, but any number above this will, on average, double their token size to 8K. So in a situation where many users are members of between 70 and 80 security groups, but over a couple of weeks are added to 10 more security groups, they all are driven above the 4K threshold. Unexpectedly the total amount of kernel memory needed for each connection to Exchange doubles from 4K to 8K.

An Outlook user will normally have 6 to 8 connections to an Exchange server due to different connections. Separate connections are not only required for mailbox access, but also for services like offline address list, calendar, desktop search, and public folders. A user with 8 connections and a 8K token would need 64K of paged pool memory on the server just for their access tokens. A typical mailbox server at a large organization such as this one will allocate around 200 MB of memory for access token connections. At this figure it would be feasible to maintain approximately 6000 users if each averages 4K for access token size. However if all users average 8K tokens the 200MB apportioned for page pool memory will only support 3000 users. This restriction is independent of other resources such as hard drive space, processor speed, or network bandwidth. Other server resources may be able to handle a larger load, but page pool memory pushing the overall

server memory to its limit is a bottleneck that will prevent Exchange services from being accessed by users once the memory limit is reached.

This is exactly what was happening in this case. The servers were scaled to handle 5,000 users each, but as everyone in the company had their access token SID grow by a few each day; large numbers of users were having their 4K access token jump to 8K and their 8K token rise to 12K. Multiply these numbers by the thousands of users and the 200 MB allocated for paged pool memory fills up quickly.

The Exchange team used a set of scripts that will be covered in a later section that were made available by the Microsoft Exchange team to track the group membership trends and confirm the growth movement of all users. The Security team eventually traced the activity back to the disgruntled employee and deleted the unnecessary groups (Lee, 2006).

5. Preventing, Monitoring, Resolving Token Issues

5.1. Prevention

Reducing access token issues begins with analyzing the current group creation process within an organization. Questions that should be asked include:

- Who can create a group?
- Should all those that have the ability to create groups really have it?
- Is there a process in place to regularly review group membership and remove members that no longer require membership??
- Is there a process in place to delete old groups that are no longer used?

Answering these questions and making the necessary changes to limit who can create groups and establishing the necessary processes to review groups and group membership are essential first steps in preventing group membership and token issues.

Preventing the common nesting scenarios that can cause a large organization to quickly have large token size issues is another important step. There are a few common group nesting scenarios that may seem like logical groupings when the groups are initially created, but at the in time contribute significantly to access token issues. As discussed above, the Microsoft Best Practice for group structure is to add users to Global

Groups, nest the Global Groups within Domain Local Groups, and assign permissions to the Domain Local Group. However, this common group nesting scenario can cause major access token size issues when creating a single Global Group and nesting it in several Domain Local Groups. For example, say a company has a group of travelling salesmen and place them in the "Travelling Sales" Global Group. There is a Domain Local Group (named "Canada Printers", "New York Printers", "Philadelphia Printers", etc.) for every local office to grant access to the printers within that office. The company has 100 offices. Normally only groups that contain users from the local office are added to the printer domain local groups. However, since the "Travelling Sales" users often visit many different offices throughout the year, the IT department decides it makes sense to add them to the printers' domain local group in all 100 offices. All of the users within the "Travelling Sales" group now have 101 SIDs added to their access token due to this configuration. While it is easy to see from an access token size viewpoint this is a poor configuration, from strictly an access standpoint it seems logical.

Another example of a common group nesting scenario that can cause major access is creating multiple Global Groups or Domain Local Groups with the exact same users. For example, the Accounting team initially just has a standard "Accounting Users" group that contains the entire 40 members of the department. Over time the following additional groups are created: "Tax Tool Users," "Accounting Printer Users," "Finance Software Users," "Income Statement Users," and "Payroll Users." All the groups contain the exact same 40 Accounting team members when only one group was probably necessary.

Another preventative step is to document groups that are created. When people don't have the resources available to determine whether a group they need already exists, they most likely are going to create a new group themselves. Duplicate groups can be especially damaging when they are particularly large groups. For example, someone needs to create a group for company benefits that includes only internal company employees that excludes contractors, business partners, and vendors. Without proper documentation, they have Benefits group created. They didn't realize they had duplicated a group that was already created three times already for travel, annual reviews,

and company discounts. Instead of having a single SID for an internal users group, all the users have four SIDs.

Documenting groups can be equally important for deleting old groups that are no longer needed and for consolidating groups that are redundant to minimize access token size. When there is an old group sitting in a production environment and nobody seems to know what exactly it is for, people will tend to be cautious and leave the group alone. So groups that are no longer used will remain adding to users' access tokens. The same situation arises when trying to reduce group memberships for users with access token problems. If there are groups that have the same users in them and could be consolidated to a single group, but there is uncertainty as to what exactly many of those excessive groups are securing, it will be difficult to complete the reorganization to reduce token size (Stevenson, 2005).

5.2. Monitoring

There are a few options to consider when determining how to examine group membership within an organization. The Microsoft Exchange team created two extremely useful scripts that each merit regular use.

The first is group_statistics.vbs that gives a text-based histogram view that displays how many users belong to 70 groups, 80 groups, 90 groups, etc. This can assist in determining the average token size for users. This is an important script to run regularly. When it comes to monitoring group membership levels within an organization it helps to run group_statistics.vbs to create an initial report to baseline where an organization is currently and then update the report regularly to determine group membership trends. The trends might show that the problem is isolated to a few users or may show that the average groups a user is a member of is growing so quickly across the entire organization that action needs to be taken immediately to address potential issues. A copy of this script can be found in Appendix A.

The second script the Exchange team created is the groups.vbs script that simply lists every email enabled user in the forest and lists all of the groups in which they are a member. This can be ran against all users or limited to specific users. This script is useful to run against the users identified as having a significant number of group

memberships to the point where DoS may occur. A copy of this script can be found in Appendix A.

Another tool that is useful in determining group membership issues is the Group Membership Evaluation tool within Ntdsutil.exe. This tool will generate a report that can assist in identifying groups that may cause access tokens to exceed the group membership limit. The report can help by decreasing the number of groups that need to be examined by isolating the groups a user, group, or computer is a member of. This is achieved by providing a list of security identifiers (SIDs) that will be present in the access token for an account or group. This information can then further assist in narrowing down what group memberships are having the largest impact on token size (Lee, 2006).

5.3. Resolving

Unfortunately there is no "one size fits all" solution for remediating token size issues. It often requires one to dissect a user's group membership and scrutinize for common patterns that often are the cause of large increases in group membership. Fixing two of the more common group nesting scenarios that often cause token size issues are examined below (Stevenson, 2005).

A common problem group nesting scenario that was discussed in the Prevention section above had a company with a group of travelling salesmen who were members of the "Travelling Sales" Global Group and were then nested in a series of 100 Domain Local Groups (named "Canada Printers", "New York Printers", "Philadelphia Printers", etc.) for every local office they visit.

Fortunately, this is one of the easier configurations to remedy. Creating a single domain local group, nesting the global group within the new domain local group, removing the global group from the numerous domain local groups it is currently a member of, and granting permissions to the new domain local group will drastically reduce the size of the access token. For the above example, following these steps will fix the problem:

- Creating a "Travelling Sales Printer" domain local group

- Remove the "Travelling Sales" global group from the 100 office printer domain local groups
- Add the "Traveling Sales" global group to the "Travelling Sales Printer" domain local group
- Grant the "Travelling Sales Printer" group permissions to all printers at all offices

These simple steps will reduce the SIDs within the travelling sales users' access token from 101 to just 2. While straightforward, this still involves assigning a new group permissions to various resources which can be difficult if the exact resources the old groups have access to are not well documented. If the domain local groups are just used to give access to different file shares, it should be fairly clear-cut, but when companies don't do a proficient job documenting their groups, there tend to be situations where there is uncertainty as to what access certain groups give users.

The other common group nesting scenario that can cause major access token issues is creating multiple Global Groups or Domain Local Groups with the exact same users or groups. The example used above in the Prevention section had the Accounting team as members of an "Accounting Users" group that contained all members of the department. Over time the following redundant groups were created: "Tax Tool Users," "Accounting Printer Users," "Finance Software Users," "Income Statement Users," and "Payroll Users." All the groups contained the exact same Accounting team members. Instead of using the one "Accounting Users" groups to grant access to accounting resources, unnecessary additional groups were created with the same members in each one.

Another option outside of group reconfiguration that can resolve some token size issues is upgrading to 64 bit operating systems which do not have the memory limitations of 32 bit operating systems. This change will certainly resolve the Exchange Page Pool Memory issue if an Exchange Server is upgraded to a 64 bit operating system and additional memory is added to each server.

6. Conclusion

Access token and group membership issues are not likely to become detected until they are causing significant problems within an organization. Once they begin causing complications, however, they can be quite costly and time-consuming to deal with. Large organizations that have been using Active Directory since its inception are likely to face these difficulties. Poorly regulated group creation can cause both unintended and malicious DoS of anything from a single user's desktop to several critical servers across an enterprise. It is in these organizations' best interest to take preventative measures that include analyzing the group creation process within their organization and regularly monitoring group membership levels across all users. Only then can they deal with these issues before they cause outages.

7. References

De Clercq, Jan. *Windows Server 2003 Security Infrastructures: Core Security Features* . 1st ed. New York, NY: Digital Press, 2004. 137-163. Print.

Foley, Mary Jo. (2010, February 26). *Behind the IDC data: Windows still No. 1 in server operating systems*. Retrieved from http://www.zdnet.com/blog/microsoft/behind-the-idc-data-windows-still-no-1-in-server-operating-systems/5408

Garman, Jason *Kerberos: The Definitive Guide* Sebastopol, CA: O'Reilly & Associates, Inc (2003). 28-40

Johansson, Jesper. *Windows Server 2008 Security Resource Kit* Bellevue, WA: Microsoft Press (2008). 7-10. Print

Lee, Mike. (2006, January 3). *Scripts to count and generate statistics about the number of security groups to which an Exchange user belongs*. Retrieved from http://blogs.technet.com/b/exchange/archive/2006/01/03/416840.aspx

Microsoft. (2010, December 10). *How to raise active directory domain and forest functional levels*. Retrieved from http://support.microsoft.com/kb/322692

Microsoft. (2007, November 16). *How to monitor and troubleshoot the use of paged pool memory in Exchange Server 2003 or in Exchange 2000 Server*. Retrieved from http://support.microsoft.com/kb/912376

Stevenson, Susan (2005). *Addressing problems due to access token limitation*. Redmond,

WA: Microsoft Corporation.

Thomas, D. (2009, October 23). *Kerberos Authentication Issues* Retrieved from http://blogs.iis.net/thomad/archive/2009/10/23/kerberos-authentication-issues.aspx

8. Appendix

Here is the groups.vbs script (Lee, 2006):

```
'=====================================================================================
' NAME: Groups.vbs
' AUTHOR: Kyryl Perederiy, Microsoft IT, MACS Engineering
' DATE  : 12/15/2005
' COMMENT: The script runs through all mailbox enabled user objects in the
' forest and calculates the number of security groups and groups in SID
' history for each object. User objects can be filtered by Exchange home server.
' PARAMETERS: <output file> <GC Domain Controller> <Domain Naming Context> [<Exchange Server(s)>]
' EXAMPLE: CSCRIPT groups.vbs groups.tsv EXCH-DC-01 dc=root,dc=company,dc=com EXCH-MBX-*
' Version 1.0
'=====================================================================================
On Error Resume Next
Set strArgs = WScript.Arguments
Set fso = CreateObject("Scripting.FileSystemObject")
Set fileStream = fso.OpenTextFile(strArgs(0), 2, True, TristateTrue)
fileStream.WriteLine "DN      Mail        Domain   Login       Server      GRP        SIDHISTORY"

Count=0
DCS = strArgs(1) ' Domain Controller
strDomainNC = strArgs(2) ' Domain Naming Context for the forest
strFilter = "(&(mail=*)(objectCategory=person)(objectClass=user)" &_
                                "(msExchHomeServerName=*" & strArgs(3) & "))" 'Mail users search filter

Set oConnection = CreateObject("ADODB.Connection") ' Setup the ADO connection
Set Com = CreateObject("ADODB.Command")
oConnection.Provider = "ADsDSOObject"
oConnection.Open "ADs Provider"
Set Com.ActiveConnection = oConnection ' Create a command object on this connection
Com.CommandText = "<LDAP://" & DCS & ":3268/" & strDomainNC & ">;" &_
                                                strFilter & ";distinguishedName,mail,sAMAccountName," &_
                                                "msExchHomeServerName,sIDHistory,homeMDB;subtree"

' Set search preferences
Com.Properties("Page Size") = 1000
Com.Properties("Asynchronous") = True
Com.Properties("Timeout") = 120 ' seconds
set oRecordSet = Com.Execute

oRecordSet.MoveFirst

While Not oRecordset.Eof

        Count=Count+1
        DN = oRecordset.Fields("distinguishedName").Value
```

```
        Mail = oRecordset.Fields("mail").Value
        Server = oRecordset.Fields("msExchHomeServerName").Value
        Server = Mid(Server,InStrRev(Server,"=")+1)
        Domain = Split(DN,",DC=")
        Login = UCase(Domain(1)) & "\" & oRecordset.Fields("sAMAccountName").Value

        set oDirObject = GetObject("LDAP://" & DCS & "/" & replace(DN,"/","\/"))

        ' tokenGroups is a computed attribute that contains the list of SIDs
        ' due to a transitive group membership expansion operation on a given user
        oDirObject.GetInfoEx ARRAY("tokengroups"),0

        ' Size of the array correspond to the number of groups
        GROUPS = ubound(oDirObject.GetEx("tokengroups"))+1

        If IsNull(oRecordSet.Fields("sIDHistory").Value ) Then
                SIDHIST = "0"
        Else
                SIDHIST = ubound(oDirObject.GetEx("sidhistory"))
        End If

        WScript.Echo Count & CHR(9) & DN & CHR(9) & GROUPS
        fileStream.WriteLine _
                DN & CHR(9) &_
                Mail & CHR(9) &_
                UCase(Domain(1)) & CHR(9) &_
                Login & CHR(9) &_
                Server & CHR(9) &_
                GROUPS & CHR(9) &_
                SIDHIST & CHR(9)

        oRecordset.MoveNext

Wend

WScript.Echo "Total: " & Count & " users found on the server(s): " & strArgs(3)
```

Here is the groups_statistics.vbs script (Lee, 2006):

```
'=================================================================================

' NAME: groups_statistics.vbs

' AUTHOR: Kyryl Perederiy, Microsoft IT, MACS Engineering

' DATE  : 12/15/2005

' COMMENT: Runs through all mailbox enabled user objects in the forest and

' calculates statistical distribution for group membership

' PARAMETERS: <output file> <GC Domain Controller> <Domain Naming Context> [<ExchHomeServerName>]

' EXAMPLE: CSCRIPT groups_statistics.vbs groups_statistics.tsv EXCH-DC-01 dc=root,dc=company,dc=com
EXCH-MBX-0*

' Version 1.0
```

```
'====================================================================

On Error Resume Next

Dim GROUPS(100)

Set strArgs = WScript.Arguments

Set fso = CreateObject("Scripting.FileSystemObject")

Set fileStream = fso.OpenTextFile(strArgs(0), 2, True, TristateTrue)

fileStream.WriteLine "Groups" & CHR(9) & "Users"


Count=0

DCS = strArgs(1) ' Domain Controller

strDomainNC = strArgs(2) ' Domain Naming Context for the forest

strFilter = "(&(mail=*)(objectCategory=person)(objectClass=user)" &_

                    "(msExchHomeServerName=*" & strArgs(3) & "))" 'Mail users search filter


Set oConnection = CreateObject("ADODB.Connection") ' Setup the ADO connection

Set Com = CreateObject("ADODB.Command")

oConnection.Provider = "ADsDSOObject"

oConnection.Open "ADs Provider"

Set Com.ActiveConnection = oConnection ' Create a command object on this connection

Com.CommandText = "<LDAP://" & DCS & ":3268/" & strDomainNC & ">;" &_

                                    strFilter & ";distinguishedName,sAMAccountName;subtree"


' Set search preferences.

Com.Properties("Page Size") = 1000

Com.Properties("Asynchronous") = True

Com.Properties("'Timeout") = 120 'seconds

set oRecordSet = Com.Execute
```

```
oRecordSet.MoveFirst


While Not oRecordset.Eof


        Count=Count+1
        set oDirObject = GetObject("LDAP://" & strArgs(1) & "/" &_
                replace(oRecordset.Fields("distinguishedName").Value,"/","\"))
        oDirObject.GetInfoEx ARRAY("tokengroups"),0
        GRP = ubound(oDirObject.GetEx("tokengroups"))+1
        GROUPS(Int(GRP/10)) = GROUPS(Int(GRP/10)) + 1
        WScript.Echo Count & CHR(9) & oRecordset.Fields("sAMAccountName").Value & CHR(9) & GRP
        oRecordset.MoveNext
Wend
WScript.Echo "Total: " & Count & " users found"
WScript.Echo "See " & strArgs(0) & " for details..."
For i=0 to 100
        fileStream.WriteLine i*10 & CHR(9) & GROUPS(i)
Next
```