Windows 2000 Security - An Overview and Analysis Part 1

Dr. Jesper M. Johansson

Introduction and History

On February 17, 2000 Microsoft released the newest version of its "industrial strength" operating system, Windows NT. This version, originally named Windows NT 5.0, was renamed Windows 2000 about half-way through the beta cycle. Although the name change appears to have been largely a marketing move, it has caused no small amount of confusion for users. Many still believe that Windows 2000 is essentially a "better Windows 98." Nothing could be further from the truth. Contrary to Windows 95 and 98, both of which have virtually no meaningful security mechanisms, Windows 2000 is an upgrade to Windows NT 4.0. Windows NT 4.0 was designed with a security model in mind; Windows 2000 modifies and extends this model. Hence, a discussion of the security features of Windows 2000 is highly meaningful and relevant. In this first part of a paper on this subject I present the security model of Windows 2000 as it relates to three basic components of security: user identification, user accountability, and object security.

Windows 2000 is quite possibly the most complex operating system ever built. It follows that the security related features are exceedingly complex as well. In this paper I will not try to cover all security related features of Windows 2000. To do so would require a few books. Nor am I trying to expose all security related vulnerabilities in Windows 2000. No one could do that at this early stage. I am simply setting out to evaluate how the most fundamental tenets of InfoSec, user identification, accountability, and object security, are designed and implemented in Windows 2000. Each section covers a specific topic. Section 2 contains a review of the Windows 2000 security model, and Section 3 discusses how user identification information is managed in Windows 2000. The next and final paper in the series looks at how it is possible to restrict and audit access once users are identified. The second paper also presents some observations and conclusions.

The Security Model in Windows 2000

Windows 2000 is based on the same security model as its predecessor, Windows NT 4.0. That model, originating with the first release of Windows NT, is designed from the ground up to match the security requirements of the United States Department of Defense's Trusted Computer Systems Evaluation Criteria (TCSEC) C2 classification [1]. These requirements dictate that the trusted computing base must provide three security measures:

- User identification
- User accountability
- Object security

These same features are part of the new Common Criteria that is replacing the TCSEC as the preferred security evaluation criteria. Windows 2000 has elements of at least the following security functional requirements [2]:

FAU_GEN	Security audit data generation
FAU_SAR	Security audit review
FDP_ACC	Access control policy
FDP_ACF	Access control functions
FDP_RIP	Residual information protection
FIA_AFL	Authentication failures
FIA_ATD	User attribute definition
FIA_UAU	User authentication
FIA_UID	User identification
FIA_USB	User-subject binding

Note that *no formal evaluation of Windows 2000 with respect to any formal security criteria has been performed*. By listing formal requirements I thus do not mean to imply that Windows 2000 is entirely compliant with them. However, Windows 2000 does at least meet elements of these functional requirements. Whereas in Windows NT 4.0 both accountability and object security are optional during setup, user identification in addition to these two requirements is optional in Windows 2000. A given system does not have to enforce any of them, although for the remainder of this paper we will assume that a system is configured to do so.

Security Requirements and Windows 2000

The most basic requirement for any secure system is user identification. In Windows 2000, user identification is optional. During setup, the person installing the system is asked whether the system should automatically log on a user at startup. This could also be done in Windows NT 4.0, but is not configured during setup. If automatic logon is selected, all user identification is effectively disabled and further security measures based on user identification, such as object security and user accountability, are meaningless. However, if the system is configured to require a logon, it can track user accounts. These accounts can be managed either locally on the workstation, or by a network server known as a *domain controller* (DC). Local accounts are usable only to connect to the workstation or server where they are defined. They are stored in a portion of the Registry database on the local system. This portion of the Registry is readable only by registered operating system components, and contains all information about each user's account, including the password representations. If network accounts are used, the computer must be a member of a domain. A domain is a grouping of systems that share a common user account database. Accounts are managed by the *Security Accounts Manager* (SAM), which manages the SAM Database. Under Windows NT 4.0, the domain accounts database was stored in the Registry of the DCs, just like a local database. However, in Windows 2000 it is stored in the Active Directory, which is described below.

Windows 2000 File Systems
Windows 2000 supports three disk file systems: *16-bit File Allocation Table* (FAT16), *32-bit File Allocation Table* (FAT32), and the *New Technology File System* (NTFS) version 5.0. The first two file systems hail from the days of MS-DOS. FAT16 is the file system supported under MS-DOS, and DOS-based operating systems, such as Windows 95. However, FAT16 limits partition sizes to 2 gigabytes (4 on Windows NT) and therefore a 32-bit version of FAT was developed for a service release of Windows 95. FAT32 supports volumes as large as 2 terabytes, although the maximum size volume that can be formatted under Windows 2000 is 32 GB. This is not a technical limitation, but rather a limitation related to the Windows 2000 format command. Windows 2000 can access larger FAT32 volumes created under Window 98. NTFS should, however, be used on all volumes in Windows 2000. Windows 2000 maintains *Discretionary Access Control Lists* (DACL) and *System Access Control Lists* (SACL) on files and directories within all NTFS volumes. Without NTFS, object security and user accountability are not possible. Windows 2000 also maintains DACLs and SACLs on other system objects, including memory constructs, the Active Directory, and the Registry. Section 4 in the next paper in the series will discuss ACLs in Windows 2000.

User Identification - The Active Directory

Perhaps the most important difference between Windows 2000 and Windows NT 4.0 is the move to a new method of tracking user and computer accounts; these are now tracked in the *Active Directory* (AD). The term *Active Directory* is really a marketing term denoting the database that organizes and stores user account information for a Windows 2000 domain. It consists of two parts; the storage mechanism itself, and a location mechanism used to locate entries in the database. The database, stored in a file called *ntds.dit*, is managed by the *Extensible Storage Engine* (ESE), which is also used in Microsoft Exchange. That engine, in turn, is based on the *Microsoft Jet Engine*. Access to the data store is accomplished almost exclusively through the *Directory System Agent* (ntdsa.dll). The only exception is that *Messaging API* (MAPI) clients may access the database layer directly. The main interfaces to the directory are either the *Lightweight Directory Access Protocol* (LDAP) [3] or *Active Directory Services Interface* (ADSI). ADSI is an API that provides an interface to LDAP from a wide range of languages, such as C++, Visual Basic, Visual Basic Script, JavaScript, and others.

The Active Directory is created when the first server is promoted to become a DC. At that time, the SAM database is replaced by a "stub SAM;" the Registry-based SAM database on a DC holds only a few security accounts. All the operational accounts are now stored in AD. The few accounts left in the SAM are used when the DC is started in *Directory Services Restore* mode. The disposition of the accounts in the SAM when a server is promoted depends on whether a new domain was created, or whether the server was added to an existing domain. If the server is promoted to manage a new domain, the accounts in the SAM become accounts in the new AD domain. If the server is added to an existing domain, the local accounts are removed.

Objects in Active Directory are uniquely identified by a *Globally Unique Identifier* (GUID). In previous versions of Windows NT objects were identified by a *Security Identifier* (SID). However, the Active Directory stores entries from several domains, and it is possible that SIDs are identical across domains. Each object still has a SID, but the SIDs are no longer static. A SID for a particular object may change, and the Active Directory maintains the binding between the GUID and the SID if necessary.

One of the main shortcomings of Windows NT 4.0 was its limitation of being capable of holding only 40,000 objects in the SAM database. Active Directory removes that limitation. The Windows 2000 implementation of the ESE can support databases up to 16 terabytes in size. Microsoft has tested the Active Directory with 40 million objects.

Active Directory Data Model

The Active Directory data model is based on the X.500 [4] model. However, AD is not an X.500 compliant directory, nor does it support X.500 protocols. It is simply related in that the data models are similar. Items are identified using the

X.500/LDAP distinguished name convention [5]. AD is based on an object-oriented model of classes of objects containing attributes. Each class, in turn, is an instance of the classSchema class, while each attribute is an instance of the attributeSchema class. Thus, the entire schema definition of AD is stored in AD itself. The schema definition is stored in the *Schema Container* of the *Directory Information Tree* (DIT). The default DIT that ships with Windows 2000, called the "base-DIT," contains 142 classes. However, not all of these are used to define user and computer objects. For example, the above mentioned classSchema and attributeSchema classes are part of that number. Many of the classes have hundreds of attributes. For example, the user class has 212 attributes if all the attributes of its superclasses are included. To show a drawing of all the classes and attributes is obviously impossible. A subset of the classes and associated attributes that are relevant to user and entity identification is shown instead in Figure 1 - please refer to the legend for an explanation of the semantics used.

The data model in Figure 1 is only a subset of the classes available in Active Directory. It is a subset that relates to user identification, and hence contains the entire inheritance hierarchy (see below) for users, as well as relevant containers that can hold user accounts, such as domains and organizational units. In addition, a few of the attributes for each class are shown, where they are relevant to security. Attributes shown in bold face are required; all others are optional.

The Active Directory inheritance model is analogous to that of Java, and is based on the 1993 X.500 specification. It contains three kinds of classes:

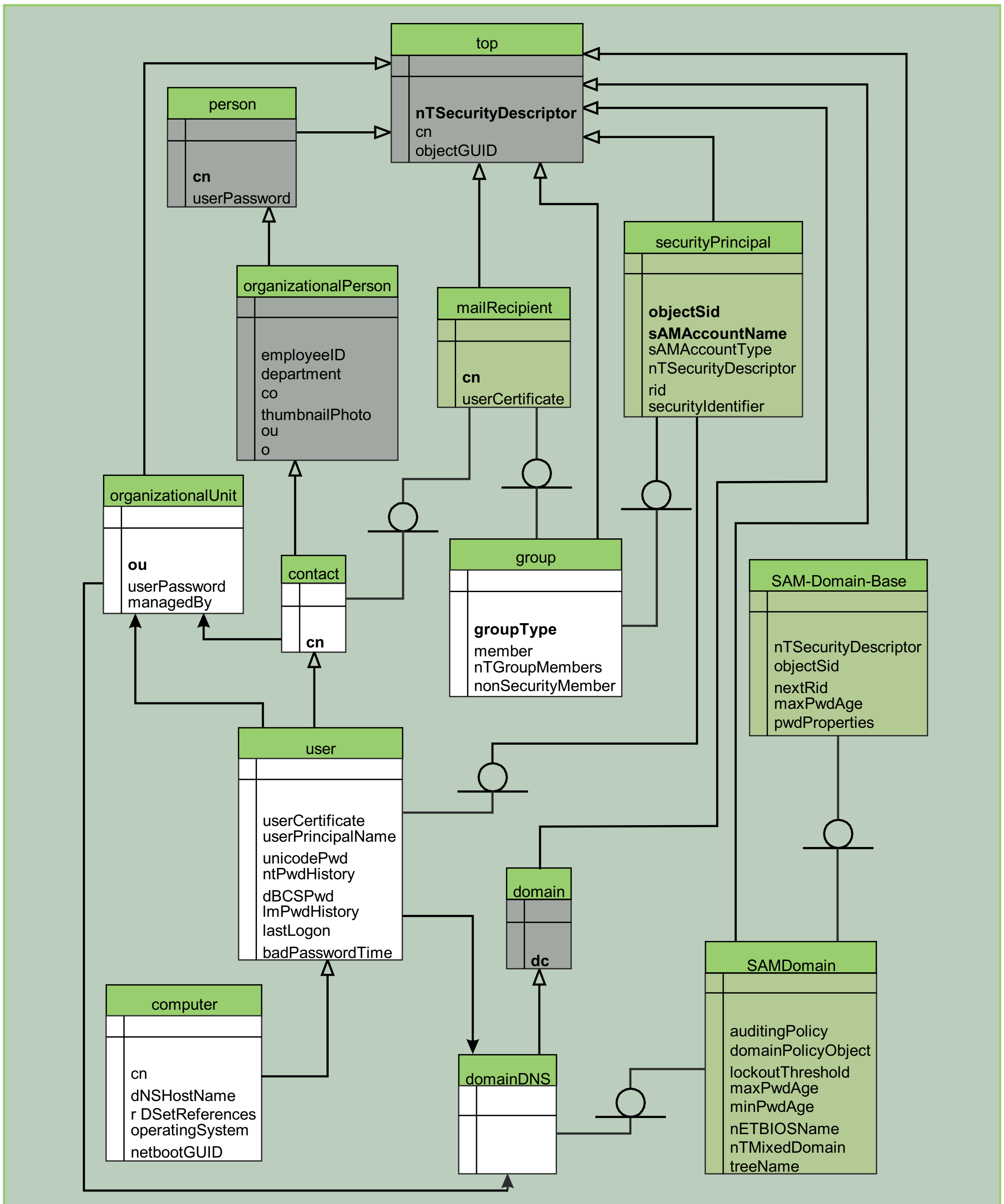
- Structural (type 1)
- Abstract (type 2)
- Auxiliary (type 3)

Only structural classes can be instantiated, in other words, all objects in the Active Directory belong to a structural class. Structural classes are shown in white in Figure 1. A structural class can be derived from an abstract class. Abstract classes are classes that define a template for new classes only. These classes are shown in gray-green in Figure 1. Auxiliary classes are classes that contain only attributes and that cannot be instantiated. They are essentially conceived of as a container for attributes common to other classes, which can derive from an auxiliary class. The model is strictly single-inheritance based. A class can have only one parent, but can implement the attributes of several auxiliary classes. The auxiliary classes are shown in light green in Figure 1. Abstract and auxiliary classes can be sub-classes of other abstract or auxiliary classes, but may not be subclasses of a structural class, since any subclass of a structural class is also an instantiable structural class.

As can be seen in Figure 1, all classes in Active Directory derive from the root class top. Top defines a number of attributes, such as the *Object-GUID* (Globally Unique Identifier), which is used to uniquely identify all objects in the Active Directory. The top class also has a required *nTSecurityDescriptor*, which contains the security information about an object. It contains the ID of the object's owner, as well as the optional DACL and SACL. Users are not directly derived from top. Rather, a person is the class derived from top. A person is simply a class to hold all instances of people objects. It contains little of interest, except that there is an attribute called *userPassword* in the person class. That string holds a plain-text password used by LDAP clients. It does not appear to be related to the user's NT password at all. A person also has a *cn* (LDAP common name). From a person, an *organizationalPerson* is derived. An organizationalPerson has many attributes related to employees, such as Employee IDs, and even photos. However, no security related information is defined in organizationalPerson. Since Active Directory is deeply tied into the object model for Microsoft Exchange, contacts are derived from organizationalPerson. Contacts implement the interface in the auxiliary class mailRecipient. Finally, *user* is derived from contact. This means that all users by definition are contacts, and hence e-mail users. The user class contains the user's name (userPrincipalName), any X.509 certificates held by the user, in addition to those held by the mailRecipient, that is, that user, and various security-related information.

Among the more interesting attributes are the two passwords - *unicodePwd*, and *dBCSPwd* - with their associated password histories. The unicodePwd is the descendant of the NT hash from NT 4.0. It contains a hashed representation of the user's Unicode password. The hash is further encrypted using the syskey technology. The dBCSPwd is the LanMan hash that was also created in Windows NT 4.0. In other words, even Windows 2000 is backward compatible with LanMan clients, at the expense of the security implications of storing the LanMan hash. The user class also stores information regarding the last logon time of the user, the last logon computer, etc.

Most of the security related information for users is actually not contained in the user class itself, but rather in the *securityPrincipal* auxiliary class. The latter class contains the Security Identifier for each object, as well as the relative identifier of the Active Directory domain in which the object is defined. The security principal also has an nTSecurityDescriptor. This raises an obvious question in how conflicts between the nTSecurityDescriptor in the top class and that in the securityPrincipal class are handled. Each object oriented language has to define how such conflicts are handled. Normally, the attribute defined in the sub-class will override the attribute



Legend:

⊥ Category - The class connected to the bottom of one of these is a category of the class at the top of one of these. The top is typically an auxiliary class

→ Relationship - essentially a many-to-one relationship. The class pointed to has an attribute that can hold one or more of the related class.

▷ Inheritance - all classes, except the root class top, are derived from another class. The class pointed to in one of these relationships is the parent class.

⬜ Structural class

▦ Abstract class

🟩 Auxiliary class

Figure 1 - The Active Directory Data Model Subset

defined in the super-class; it would thus be reasonable to assume that this would happen in the AD as well. However, no documentation on that process is currently available. The attribute classes of the two nTSecurityDescriptors are identical, so there appears to be no additional functionality added by having another nTSecurityDescriptor in the securityPrincipal class. The same issue arises with the sAMDomainBase class.

Users can be grouped into *organizational units* (ous). An ou is a construct that represents a group of users who have similar security requirements. An ou can contain separate group policies, but other than that can be conceived of simply as a container. It is essentially used to group those users together that have particular characteristics, such as needing access to a certain printer, or having certain password requirements. These things can be implemented using a group policy that applies only to the ou. The ou has a managedBy attribute, which is the GUID of the user that is charged with managing the ou.

Users are obviously members of groups. However, the implementation of that in Active Directory is rather counterintuitive for those who are not used to object oriented systems. In a standard relational database, this would be a many-to-many relationship, and hence would need an intersection entity to handle the instances. In AD, the group class contains a multi-valued attribute called nTGroupMembers, which contains zero or more GUIDs who are members of the group. Groups also contain other member attributes, such as member and nonSecurityMember. Non-security members are used for distribution groups. Windows 2000 defines two types of groups: *security groups* and *distribution groups*. Distribution groups are used by Microsoft Exchange (and possible other messaging servers in time) to define mailing lists. Security groups are the kinds of groups we had in Windows NT 4.0; they hold users and could be assigned permissions. The group type attribute is used to define the scope of the group. A *universal group* is replicated to the Global Catalog server, whereas a *limited group* is used only within the domain. The limited groups are *domain local groups*, which are used to grant access to resources within a domain, and *global groups*. Global groups are available in trusting domains and can be used to assign user permissions in that case.

Users and groups are members of a domainDNS, which is the Windows 2000 AD domain. It is derived from a domain class, and implements the sam domain interfaces, *samDomain* and *samDomainBase*. The sam domain interfaces are where all the security information for a domain is defined. They contain the relative identifiers, the password policies, auditing policies, and so forth. It seems a little strange to define a samDomainBase auxiliary class, then to derive the samDomain class from it. It is unclear why Microsoft has done this.

It is very interesting to note that computers are derived from users. That means that computers are users, and contacts, and organizationalPersons, and even persons! Apart from the anthropomorphizing of computers, there is little information that is directly related to security in the computer class. Most of that information is contained in the user parent class and the securityPrincipal class. The computer class is mostly used to set the host name, the operating system information, and a GUID for use when the computer is booted from the network. The GUID is stored on the PXE-compliant network interface card for the computer.

Conclusion

So what will Windows 2000 provide organizations interested in high levels of information security? The answer is "much." Consider the security model discussed earlier, a well-reasoned model that incorporates strong authentication, auditing, access control, and so forth. Windows 2000 appears to be the first Microsoft operating system that is potentially well-suited for high risk operational environments. At the same time, however, the security and data models for Active Directory are extremely complex, perhaps overly so. The documentation available, although copious in volume, is also strangely void on such fundamental issues as how inheritance conflicts are managed. Users are tightly integrated with the Microsoft Exchange messaging system, in preparation for Exchange Server 2000. The complexity of Windows 2000 is likely to be the greatest obstacle in successfully designing and implementing security for this new operating system.

Perhaps the largest surprise in Windows 2000, however, is the amount of backwards compatibility that is present. An example is that the LanMan hash is still created and stored, just as in Windows NT 4.0 and earlier versions. Backwards compatibility has historically been one of the most significant security problems in Windows NT as well as other operating systems. It appears that Windows 2000 is going to be no different, unfortunately. The main implication is that backwards compatibility allows attackers to find and exploit "weak links" such as vulnerabilities in legacy mechanisms used for interoperability (e.g., between Windows NT and Windows 2000 or between Windows 98 and Windows 2000). It seems as if "old ghosts never go away."

As we shall see in next part of this paper, however, complexity and backwards compatibility (and possibly also the lack of complete documentation) are not the only serious security-related concerns in Windows 2000. The default Discretionary Access ControlList (DACL) on the

Active Directory can cause a significant amount of confidential organizational information to be available to outsiders. The next part of the paper explores this and other Windows 2000 security issues.

References

1. Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*. 1985: Washington, D.C. p. 121.

2. Common Criteria Project Sponsoring Organizations, *Common Criteria for Information Technology Security Evaluation*. 1999, National Security Agency.

3. Wahl, M., T. Howes, and S. Kille, RFC 2251, *Lightweight Directory Access Protocol (v3)*. 1997, Network Working Group.

4. CCITT, X.500, *The Directory - Overview of Concepts, Models, and Services*. 1992, CCITT: Melbourne.

5. Wahl, M., S. Kille, and T. Howes, RFC 2253, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. 1997, Network Working Group.

Jesper M. Johansson is an Assistant Professor in the MIS department at Boston University. He holds a Ph.D. from the University of Minnesota, a Masters of Science in Information Systems and a Masters of Business Administration, from the University of Maryland.

His primary research interests are in the areas of information systems security and administration; and IT infrastructure design, especially the underlying networking and database designs. He teaches networking and telecommunications at Boston University. He also teaches System Administration on the Windows NT Platform for the Systems Administration, Networking and Security (SANS) Institute, for which he also edits the SANS Windows Security Digest. He has 11 years of experience in system administration and holds several professional certifications on Microsoft products, including the Microsoft Certified Systems Engineer and Microsoft Certified Professional + Internet certifications.