Analysis of the Security of Windows NT

Hans Hedbom, Stefan Lindskog
{Hans.Hedbom, Stefan.Lindskog}@hks.se

Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, SWEDEN
and
Department of Computer Science
University of Karlstad
S-651 88 Karlstad, SWEDEN

Stefan Axelsson, Erland Jonsson
{sax, erland.jonsson}@ce.chalmers.se

Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, SWEDEN

Abstract

This paper presents an analysis of the security in Windows NT 4.0, working in both stand-alone and networking mode. The objective of the work was to find out how secure this operating system actually is. A technical overview of the system, and in particular its security features is given. The system security was analyzed and practical intrusion attempts were made in order to verify vulnerabilities or to find new ones. All vulnerbilities are described in detail and classified according to a classification scheme. A comparison to commonly known UNIX weaknesses was made. It revealed generic similarities between the two systems to a surprisingly high degree. Finally a number of recommendations are given. The paper concludes that there are ample opportunities to improve the security of Windows NT. We have reason to believe that it is probably not higher than that of UNIX.

Table of Contents

Appendix

List of Figures

List of Tables

1. Scope

This report is a study of the security of the Windows NT (called "NT") operating system. The aim is to locate and use sources of information both on the Internet and elsewhere, to gain an understanding of the security features in NT and to find out how secure the system really is. This study is both focused on NT installed on a stand-alone computer as well as in a networking environment.

The work is done on behalf of Telia Research in cooperation between Chalmers University **o**f Technology and Telia Research**.**

2. Introduction

Computer security is traditionally defined by three attributes: (1) confidentiality (or secrecy), (2) integrity, and (3) availability [19] also known as “the CIA”. **Secrecy** is the aspect of keeping information protected from unauthorized users. **Integrity** implies that data only can be modified by authorized parties. Finally, **availability** means that the services are provided to any authorized user. A violation to any of these attributes is considered to be a successful attack.

Windows NT was designed with security in mind. In fact, it has been classified as a C2 level operating system (OS) by the National Security Agency (NSA) [43]. There were two major goals with this study: first, to find as many vulnerabilities as possible in NT, within a limited time. Second, to gather information about NT and its flaws.

A common method to evaluate the security of a system is to use a so called *Tiger Team*. For example, see [7], which describes a Tiger Team analysis of VM/370 operating system from the mid-seventies. Such a team is very skilled and has deep knowledge about the system and its potential vulnerabilities. In our case, however, there is a difference. We were novice NT users.

There is not much in-depth literature on NT or its security design. Lately, a number of security handbooks have been published, see for example [64], [58], [44] and [18], but most of them lack technical depth. However, there is one evaluation performed by the NSA, and the corresponding report has been issued by the National Computer Security Center (NCSC) [43]. The evaluation is mostly based on the design of the system as defined in the Orange Book [65]. We have, on the other hand, studied the operational security of NT using penetration experiments. These may reveal vulnerabilities in the design, implementation as well as in the installation.

Other penetration experiments have been carried out at our department, the most similar being a security analysis of a secure database [68]. However, most of our previous studies differ from the present. Firstly, the object systems were different: a networked UNIX operating system [62], [63] and a PC Network [66]. Secondly, the attackers were final year university students. Thirdly, the attackers had to follow some rules, since the result of these studies were used for mathematical modelling purposes.

Section 3 provides a detailed system overview of NT, while section 4 describes the networking part of the system. Section 5 concentrates on the security features, whereas section 6 focus on utility programs for NT and UNIX. Section 7 presents our intrusion experiments and section 8 is a comparison between the weaknesses found in NT and known UNIX weaknesses. In section 9 we discuss the lessons learned during this study and section 10 concludes. Appendix A presents some of the cryptography used in NT and in appendix B we explain the domain logon process. The SMB data structure as well as a detailed description of the flag fields used in SMB is the topic of appendix C. Appendix D gives examples of utility programs for NT. A list of useful web sites is presented in appendix E. In appendix F, there is a list of security related news groups. Appendix G is a list of abbreviations.

3. System Overview

3.1 Background

The NT operating system (OS) was developed by Microsoft Inc. and was first released in 1992. It is expected to replace Windows 3.x. Unlike Windows 3.x, NT is a full-fledged 32-bit OS with support for: processes, multiple threads, symmetric multiprocessing, distributed computing. NT is a single user system with support for multiple concurrent or parallel processes. Moreover, the system is designed to be able to execute on a variety of hardware platforms, e.g. Intel 386 (or later), MIPS and Alpha AXP. It is mainly implemented in C with the exception of the software that communicate directly with the hardware. These software components are written in assembler. NT uses an object model to manage its resources. Therefore, the term *object* is used instead of resource. Microsoft claims that NT is designed to be a secure operating system, e.g. one of the goals is to meet the C2 evaluation criteria. A C2 system must implement discretionary access control at user level, and it must provide mechanisms for tracking all accesses (or attempted access) to an individual object [19]. Another C2 level requirement is elimination of residue exposure. The user interface (Win32) is designed to give the user the impression of working in the familiar Windows 3.x, but behind the façade Windows 3.x and NT are totally different.

3.2 System Architecture

An OS can be designed in different ways. In one approach, common for small systems, such as MS-DOS, the OS consists of a set of procedures, which may call each other. This structure, often called a monolithic structure, has a number of shortcomings, e.g. a change in one procedure can result in an error in a seemingly unrelated part of the system [28].

Another design strategy is to divide the system into modules and layers. These systems are known as layered systems. Each module provides a set of functions that other modules (on a higher level) can use. Systems designed in this way are more easily modified and tested [28], [3]. Moreover, it is possible to replace a layer.

A third approach is known as the client/server model [28], [3]. In this model, the OS is divided into one or more processes. Each process is called a server. Such a process provides a particular service, for example memory management. An executing application is called a client. A client can, by sending a message to a specific server, request a service. All messages in the system are sent via the microkernel, which is executing in kernel mode. If multiple servers exist, they all share a single microkernel. On the other hand, both clients and servers are executing in user mode. One advantage with the client/server structure is, that a server can fail, and perhaps be restarted, without influencing the rest of the system. MACH [2] is constructed in this way.

The structure of NT is a hybrid between the layered model and the client/server model, see Figure 1. NT uses the later to provide the user with multiple OS environments (Windows, MS-DOS, OS/2 and POSIX (Portable Operating System Interface based on uniX).

3.3 Executive

The executive is the only part of the system that executes in kernel mode, and is divided into three levels. The lowest level is called HAL, which provides an abstract view of the underlying machine architecture. The motive for having this layer is to make the system (more) portable.

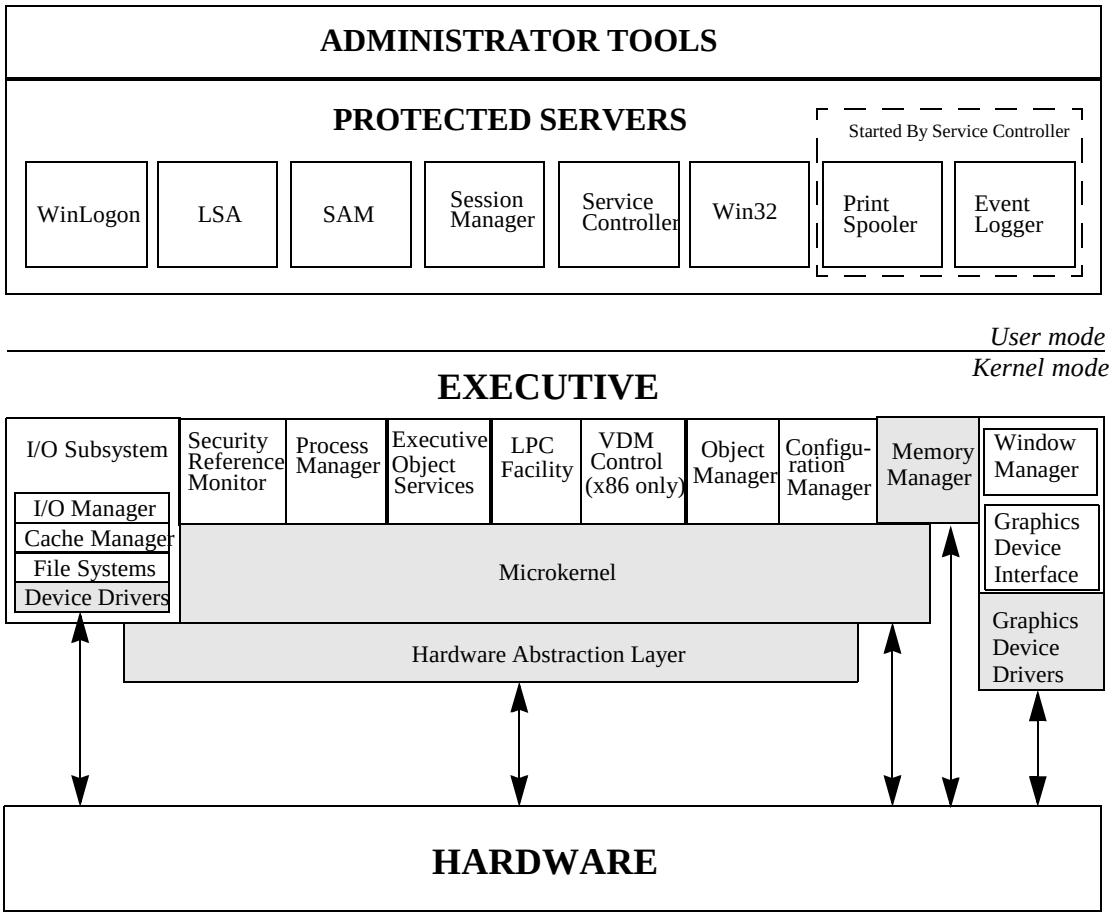

FIGURE 1. NT system overview

Above HAL is the microkernel. This is responsible for low-level support for execution, interrupts and exception handling, and synchronization [43].

The top-most layer in the executive consists of a number of components (modules) implementing basic OS services, such as: virtual memory management, object management, process and thread management, I/O management, Interprocess Communication (IPC), and security reference monitoring. Communication between these components works through a set of well defined functions in each component.

3.4 Protected Subsystems

A protected subsystem provides an Application Programming Interface (API) which programs can call [3]. Such protected subsystems are sometimes called servers, or protected servers, and are executed in user mode as processes with certain privileges [43]. When an application calls an API routine, a message is routed to the server implementing the API routine via the LPC facility. Later, the server replies by sending a message back to the caller. Trusted Computer Base (TCB) servers [43] are protected servers which execute as a process with a SYSTEM security context, which implies that the process possesses an access token, see section 5, as defined in Table 1.

TABLE 1. Token for processes running within the SYSTEM security context

Field	**SYSTEM Token Value**	
User ID	SYSTEM	
Group ID array	Everyone	
	Administrators	
Owner ID	Points to Administrator group ID	
Privilege(s)	TCB (enabled)	CreateToken (disabled)
	TakeOwnership (disabled)	CreatePageFile (enabled)
	LockMemory (enabled)	AssignPrimaryToken (disabled)
	IncreaseQuota (disabled)	IncreaseBasePriority (enabled)
	CreatePermanent (enabled)	Debug (enabled)
	Audit (enabled)	Security (disabled)
	SystemEnvrionment (disabled)	ChangeNotify (enabled)
	Backup (disabled)	Restore (disabled)
	Shutdown (disabled)	LoadDriver (disabled)
	ProfileSingleProcess (enabled)	Systemtime (disabled)
Default DACL	SYSTEM GENERIC ALL	
	Everyone GENERIC EXECUTE	
Source	Not used for SYSTEM token	
Type	Primary	

In this paper, we will describe some of the standard servers including: *Session Manager*, *WinLogon*, *Win32*, *LSA*, and *SAM*.

Session Manager is the first server to start in an NT system. It is responsible for loading DOS device drivers, subsystems registered in the Registry[1], and initialization of Dynamic Linked Libraries (DLLs), after which, it starts the WinLogon server.

1. The Registry is a collection of system information for the actual system

WinLogon is the logon process. It is responsible for coordinating and providing interfaces for interactive logon/logoff. Moreover, it manages the *Desktops*. WinLogon registers itself with Win32, see below, during system initialization as the logon process.

Win32 makes Microsoft's 32-bit Windows API available to application programs. In addition, it provides the graphical user interface and controls all user input and output. Only two objects are exported from this server: *WindowStation*, i.e. user input/output system (mouse, keyboard and screen), and a Desktop object.

LSA (Local Security Authority) has its main responsibilities centered around security. It plays a major part in the logon process, see section 5.3, and the security event logging process as well as upholding the security policy of the local system. The security policy is implemented by the local security policy database that keeps information on trusted domains, privileges and access rights for users and user-groups, security events [43]. This database is managed by LSA and accessed only through LSA.

TABLE 2. User information

Information	Purpose
Account Expiration	Time and date when account will expire; an expired account is automatically disabled (see User Account Control below).
Administrator Comment	A string containing an administrative remark or comment.
Bad Password Count	Number of attempts since the last successful logon where the authentication process resulted in a bad password.
Full Name	Full name of the user.
Home Directory and Drive	User's home directory and drive on which it is located; may be NULL.
Last Logoff/Logon	Time and date when the user last logged off/on.
Logon Count	Number of times the user is currently logged in.
Logon Hours	Hours when the user is allowed to logon.
Password	User's password stored in hashed, then encrypted form.
Password Last Set	Time the password was last changed.
Password Changeability	Two flags that indicates whether the password can change or must change.
Primary Group SID	The relative SID of the primary group to assign, or assigned to the user. For assign operations, this group must be one of which the user is a member.
Profile and Script Path	Pathnames of the user's windows profile script and logon script.
User Account Control	A bitmap used to identify account status such as account disabled, home directory required, password not required.
User Comment	A user-controlled field for comments to other users; part of the general information that can be retrieved about a user.
User SID	Relative SID of the user.
Username	Name of the user as well as the account name.
WorkStations	List of workstations from which the user is permitted to logon; NULL means all workstations.

SAM (Security Accounts Manager) is responsible for managing information about accounts for users and usergroups either locally or domain wide depending on its role. It also provides support for the authentication package, see section 5.3. The secure accounts are stored as subobject in a database in the registry. This database is accessed and managed only by SAM. Table 2 shows some of the information stored in the subobjects [43]. The hash function used to store the passwords is MD4.

3.5 Objects

In NT, both software and hardware resources are represented by objects, e.g. files, semaphores, timers, threads, processes, and memory. In fact, there are two kinds of objects [3]:

Microkernel objects which are created by the microkernel and are exported to the rest of the executive.

Executive objects which are visible in user mode. Most executive objects encapsulate (contain) one or more microkernel objects.

3.5.1 Microkernel objects

Microkernel objects, sometimes simply called kernel objects, are the most primitive set of objects implemented by the kernel and are not user-visible. They provide fundamental capabilities, that can only be accomplished by the kernel, which resides at the lowest layer of the OS. There are two types of kernel objects [43]:

Dispatcher objects control scheduling and synchronization. *Mutant*, *Event and Event Pair*, *Semaphore*, *Timer*, *Thread*, *Process*, and *Queue* form the set of dispatcher objects in NT. These dispatcher objects have a *signalled state*, allowing threads to suspend their own execution while waiting for the signalled state to change.

Control objects are passive objects used for executive and device driver control. These are not waitable, and therefore, they do not have a signal state. Control objects include: *Interrupts*, *Device queues*, *Profiles*, *Asynchronous Procedure Calls* (APCs), and *Deferred Procedure Calls* (DPCs).

3.5.2 Executive Objects

The executive provides a number of objects for the servers, e.g. Win32, and WinLogon. These objects, which are listed in Table 3, are called executive objects. They should not be confused with the objects provided to application programs through the Win32 API, the POSIX API, or the OS/2 API. In some cases, servers directly supply executive objects to their client applications. In addition, a server can construct a new type of object, for the clients, based on one or more of the primitive ones. Processes, threads and access tokens are further described in section 5, because these are key components in the security system.

TABLE 3. Executive objects

Object type	Description
Process	An instance of a running program, including the address space and resources.
Thread	An executable unit within a process.
Section	A region of shared memory.
File	An instance of a file or an I/O device.
Port	A destination for messages sent by one thread in a process to another thread in another process.
Access token	An identification containing information about a logged-on user.
Event	A notification that a system event has occurred.
Event pair	A notification that a dedicated application thread has copied a message to the Win32 server.
Semaphore	A counter that regulates the number of threads that can simultaneously use a resource.
Mutant	A mechanism that gives mutual exclusion capabilities.
Timer	A counter that records the passage of time.
Object directory	A memory-based repository for object names.
Symbolic link	A mechanism for indirect reference to an object (view also symbolic links in UNIX).
Profile	A mechanism used for performance tuning.
Key	An index for referring to records in the configuration database.

3.6 Users and groups

In NT, a person that needs access to objects on a computer must have a valid user account. In addition, to access a specific object, the user has to have access rights on it. Every user account contains information such as: username, password, full name, logon hours, logon workstations, expiration date, home directory, logon script, profile and account type. Some of these need to be further explained. Full name is the user's full name, for example Hans' full name is Hans Hedbom. The logon hours specify for how long a user is allowed to gain access to the system. The set of computers, from which a user can log on to the system is given by logon workstations. The expiration date field is the date when the account automatically becomes disabled. The profile is a file containing information about the user's Desktop environment (i.e. program groups and colors), that follows the user from one workstation to another. Finally, the account type is in most cases user accounts.

By default there are two accounts: Administrator and Guest. Within the Administrator account, new accounts can be created. To add new user accounts, the User Manager tool is used. This is a standard tool, distributed and installed together with the OS.

NT supports the concept of groups, similar to those of UNIX. Microsoft [40] are of the opinion that NT groups are more powerful than groups in UNIX. With groups, permissions can be granted to a set of related users, which makes the procedure of granting

rights and permissions easier. Moreover, a single user can belong to more than one group at the same time. Furthermore, NT provides a number of built-in groups: Administrator, Backup Operators, Printer Operators, Power Users, Users, and Guest. A user belonging to the group Backup Operators, for example, has the rights to backup the system. In UNIX, it is not by default possible to do this task without being superuser (*root*). Altogether, this implicates that different types of system administrators, with a different set of access rights can be defined in NT.

In addition to these built-in groups, a system administrator is able to define new types of groups, which can be done in the User Manager.

There are basically two types of accounts: global accounts and local accounts. A global account is accessible and visible inside the whole domain, see section 4.2, while the local accounts are accessible and visible only on the machine they were created on. There are also two types of groups: local groups and global groups. Local groups can only be used inside the domain, while global groups can be used, and are visible, in all the trusted and trusting domains of the system, see section 4.3.

3.7 File Systems

Below we describe five different file systems that are supported by NT. The first three systems, FAT, HPFS and NTFS [28], are conventional disk based file systems, while the two latter, NPFS and MSFS, are file systems used in process communication, either between processes on local or remote machines. Also, there is another supported file system in NT, CDFS, which is used when accessing files on CDs. We will not describe the latter further in this report.

In NT, file systems are accessed through file system drivers, which can be dynamically loaded into (and out of) the system. For each supported file system there is a driver, see Figure 2.

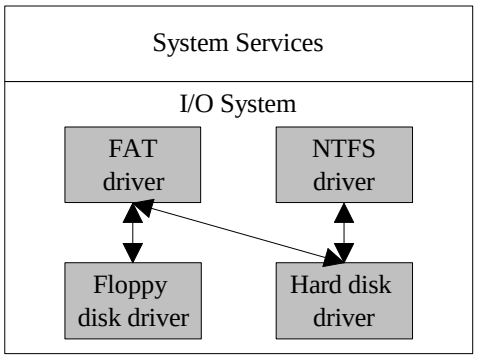

FIGURE 2. Drivers for different file systems

3.7.1 File Allocation Table (FAT)

The FAT file system was designed in 1976 by Bill Gates. Later, it was chosen for the MS-DOS operating system.The file system is simple. Moreover, it works well with small partitions. One drawback is that the FAT performs extra administrative I/O on different areas of the disk partition. Another drawback is that when a new dataset is

written to the disk, the disk arm must be constantly moved between the location of the directory and the location of the data.

One problem is external fragmentation [2]. Internal fragmentation is another problem when using large partition, e.g. greater then 32 MB. In its design FAT supports a maximum of 65536 allocation units, which implies that when large partitions are used, the size of the allocation unit is bigger then it is when small partitions are used. For example, a 1 GB partition has 16 kbyte units.

The classical FAT directory structure limits file names to eight characters plus a three character extension. Other attributes that are maintained for each file are a number of flags that indicate if the file is hidden, read-only, a system file, or should be archived the next time the disk is backed up. Finally, there is a date and time stamp when the file was last changed.

3.7.2 High Performance File System (HPFS)

HPFS is designed by Microsoft for OS/2 1.2, mostly to support the LAN Manager file server. It was constructed with performance in mind, while FAT was design to be reliable. In HPFS, better performance is obtained due to a shorter physical distance between the location of the directory and the location of the data, and frequent caching of data in the memory. The tables that describe the location of files are positioned at regular intervals throughout the dataset. When new datasets are added, they are written to areas where there is a large amount of free space. This reduces both fragmentation and disk arm movement. In addition, HPFS maintains a 512 byte allocation unit no matter how large the partition becomes. HPFS requires a large amount of data in memory. This gives the advantage of high performance, but the system is more sensitive to a system crash. After a system crash, all the partitions that were in use during the crash are marked *dirty*, and before such a partition can be used again, the CHKDSK utility program must be run. As partitions get larger, the recovery process time increases. HPFS is not recommended on small systems (say with 4-8 MB of RAM) due to the extra memory required for the disk cache.

Another advantage with HPFS compared to FAT, is that HPFS allows filenames to be long, to contain several dots, and lowercase letters. Filenames are extended to up to 254 double byte characters.

3.7.3 NT File System (NTFS)

NTFS is a completely new file system for NT. NTFS is designed to be all things to all people and to include all features of every other file system in common use. It includes features not found in other file systems supported by NT, and is designed to be a reliable and efficient system on large partitions (up to 2^{64} bytes). One of its strengths is that it supports long file names, another is the ability to extend a disk volume after it is formatted, and a third that it also offers many security features, which are unavailable in both the FAT and HPFS file systems. NTFS uses security attributes (SIDs) available at logon time to allow or disallow access to files and directories, see section 5.

A number of permissions are used to control access to file system objects, i.e. files and directories. These are very similar to UNIX permissions, and consist of: Read (R),

Write (W), Execute (X), Delete (D), Change Permissions (C), and Take Ownership (O). In NT terminology, they are called individual permissions, which must be distinguished from standard permissions. The latter are a combination of specific types of the former. Examples of standard permissions are: Read (RX), Change (RWXD), No Access (None), and Full Control (All) [40], [23]. A user may give different permissions to different users and groups for the same file system object. This is possible since NTFS uses Access Control Lists (ACLs), see section 5.

As opposed to both HPFS and UNIX file systems, NTFS supports fast file recovery. The recovery technique used is based on the transaction processing concepts, found in the database theory. This category of file systems is often called journalling file systems [40], and is formed by the system's writing updates to a log area corresponding to each volume. After a system crash, this area can be used to cleanup problems almost instantaneously.

In [20], efficiency problems, due to fragmentation of files and free space, have been reported. The degree of performance degradation is steadily worsening, as fragmentation increases. The study shows for example that the amount of time to read and write increases with the level of fragmentation by 20% to 100%. One solution to this problem is to defragment the whole volume.

3.7.4 Namned Pipe File System (NPFS)

Named pipes give support for high-level communication between two processes. A pipe is only a portion of memory that provides a full duplex channel. Thus, data written on one end of the pipe can be read on the other end and vice versa. Named pipes are implemented as a file system driver.

The security of a named pipe relies on an ACL stored on the server side of the pipe. Before a pipe can be used, the server side must be connected. Hence, named pipes provide connection-oriented messaging.

3.7.5 Mailslot File System (MSFS)

Like NPFS, the mailslot file system is implemented as a pseudo-file system, i.e. no data are backed by disk. The mailslot implementation is derived from the Microsoft OS/2 LAN Manager implementation. Although, NT implements only a subset of the latter.

Mailslots provide a connectionless and unidirectional message passing facility, used to broadcast messages. The security of mailslots is implemented in a manner similar to named pipes.

4. Networking

To meet the expected market requirements for the 1990s, one of the primary goals was to add network capabilities into the NT operating system. The communication architecture, however, is not a new one. Many of the ideas are derived from a product called Microsoft Network, or MS-NET for short, which was announced with MS-DOS 3.1 in 1984. The ideas were later carried over into LAN Manager, and now into NT [3]. Still, there is a great difference between the two latter. LAN Manager consists of a set of applications and drivers that add networking capabilities to an existing operating system, e.g. MS-DOS or OS/2. In NT, however, the communication architecture has been integrated with the operating system.

The key components in NT's network architecture are the redirector and the network server. The former is triggered when a user, or application, on a client machine requests a file on a remote machine. In such case, the redirector will receive a request and forward it to the remote machine. The network server resides on a remote computer, awaiting requests from network redirectors. The protocol used between redirectors and network servers is by default based on Server Message Block (SMB), see section 4.5, which was originally one of the pieces in MS-NET, see Figure 3. SMB is a high-level specification for formatting messages that are to be sent over the network. Choosing the existing SMB protocol will give interoperability with the old systems.

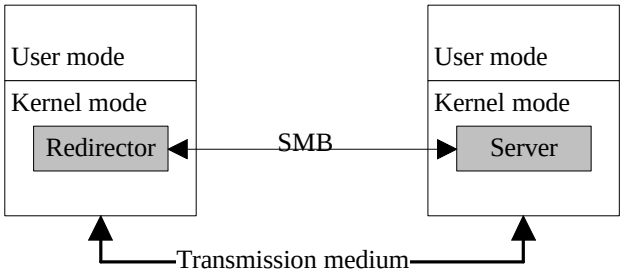

FIGURE 3. Communication between a network redirector and a network server

Both the redirector and the network server are implemented as file system drivers, and can thus be loaded or unloaded from the system at any time. This implies that they may be replaced with other components, e.g. NFS client and server. Normally, there is at least one redirector and one network server loaded on an NT system.

As mentioned above, one of NT's primary goals was to add network capabilities to the base operating system. Another primary goal was to fulfil the C2 requirements. These two, however, cannot be achieved at the same time, since Trusted Computer System Evaluation Criteria [65] assume a standalone computer. In [43], NCSC expresses this very clearly. Still, this fact has sometimes been misunderstood, see for example [44], were the authors claims that even the networking part is C2 compliant.

4.1 Network Architecture

In the early days of computing, it was nearly impossible to communicate between computers from different vendors as they all follow their own standard(s). For two computers to communicate, they had to follow the same rules, i.e. use the same protocol. In late 1970s, International Standard Organization (ISO) addressed this problem through a model for Open Systems Interconnect (OSI), which became an international standard. The OSI reference model (RM) consists of seven layers, see Figure 4, where the physical layer is the lowest one, and the application layer is closest to the applications.

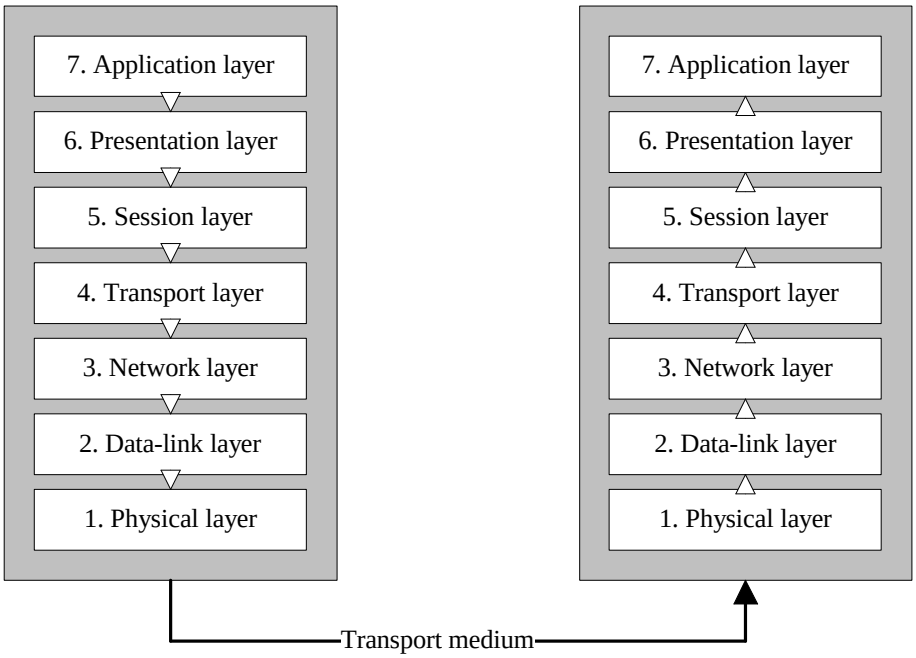

FIGURE 4. The OSI layered model

Network architectures of today are also normally implemented in a layered fashion, and NT's network architecture is not an exception. Although few companies strictly follow the OSI model, it is still used as a reference model.

4.1.1 OSI versus NT Layer Model

The various NT layers are presented in Figure 5. On the right side of the figure, the corresponding OSI layers are given.

Note. The scope of this report does not permit in-depth coverage of each layer. More details can be found in [42]. Below, however, we will describe NDIS, Transport Protocols, STREAMS, TDI and finally, some network APIs, including the WinSock API, the NetBIOS API and the RPC facility.

4.1.2 Network Driver Interface Specification (NDIS)

Microsoft and 3Com jointly specified NDIS in 1989. The idea behind the specification was to allow transport protocols to communicate with any third-party network interface card driver, given that the network adapter card drivers are implemented according to NDIS 3.0. In NT, NDIS is implemented as a wrapper, that surrounds all of the network card drivers.

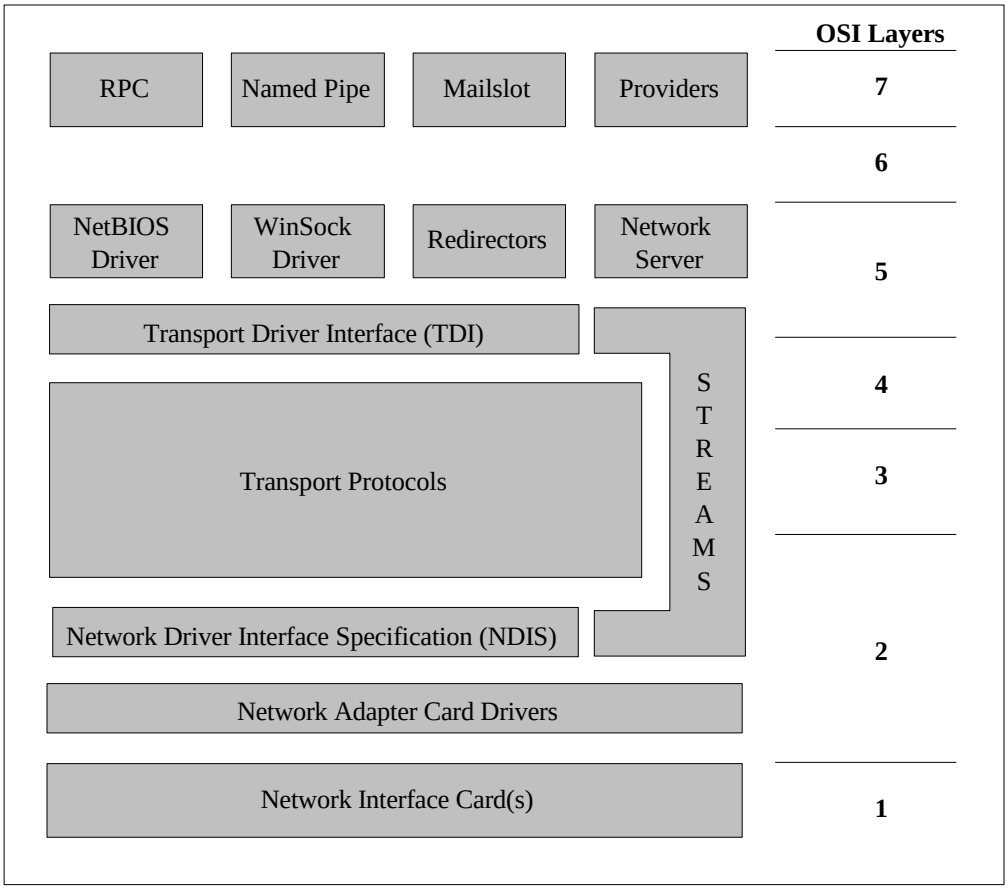

FIGURE 5. NT communication architecture

4.1.3 Transport Protocols

NT supports a large set of transport protocols, including NetBEUI, NWLink, TCP/IP, DECnet, AppleTalk and XNS. The former three are shipped with the base operating system.

NetBEUI is an acronym for NetBIOS Extended User Interface. It was developed by IBM in the mid 1980s, and was intended to be used in small local area networks. The protocol implementation, version 3.0, has slightly been modified to, among other things, map to the TDI. This version is referred to as NetBEUI Frame Protocol (NFP). The fact that NetBEUI is not routable is its major drawback.

NWLink is Microsoft's implementation of the IPX/SPX protocol. It was the default protocol until NT version 3.5 [44]. To fully utilize this protocol, a NetWare redirector is needed, which will give the opportunity to access files and printers on a NetWare server.

TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol. It is a whole suite of protocols, including TCP, UDP, IP and ICMP. TCP/IP is, since version 3.5, the default transport protocol in NT.

Yet another supported and included protocol is the Data Link Control (DLC) protocol, which provides an interface to the data-link layer (level 2 in the OSI model). In fact, it is not a true transport protocol. There are two reasons for using the DLC protocol [42]: (1) accessing IBM mainframes, and (2) printing to HP printers connected directly to the network.

4.1.4 STREAMS

STREAMS is a driver environment that has been implemented to simplify porting of transport protocols from one operating system to another. UNIX System V is another operating system supporting this environment [3]. STREAMS maps to TDI at its upper boundary, and to NDIS at its lower boundary. Thus, Figure 5 is somewhat misleading.

In NT version 3.1, both TCP/IP and NWLink were implemented as STREAM-based drivers. Due to the extra overhead involved with such implementation, these two protocols nowadays do not use STREAMS.

4.1.5 Transport Driver Interface (TDI)

TDI was created to provide a uniform interface to the various transport protocols. The selected name is somewhat misleading. Essentially, it is not a driver, it is simply a standard for message passing between the session layer and the transport layer.

4.1.6 Network APIs

NT provides a rich set of network APIs. Each API finds its own way to the network through the set of protocols in the network architecture. In section 3.7, we described named pipes and mailslots. In the introduction to section 4, the redirector and the network server were presented. However, we have not yet mentioned the WinSock API, the NetBIOS API and RPC facility.

The WinSock API is derived from the very popular standard UNIX networking interface. The interface was originally developed at the University of California at Berkeley, in the early 1980s. TCP/IP and NWLink can both be accessed through the WinSock API.

The NetBIOS API is a standard network programming interface for PCs, and has been used since the early 1980s. In NT, the NetBIOS interface is implemented as a driver, sometimes referred to as an emulator, and can be used over NetBEUI, NWLink and TCP/IP.

Both the above APIs consists of two components [42]:

- A DLL (WSOCK32.DLL respectively NETAPI32.DLL), which shares memory with the user application.
- A driver (emulator), which implements the session layer according to the OSI model.

The RPC facility consists of a run-time library and a compiler (MIDL). This mechanism allows programmers to easily write distributed applications. RPCs were introduced in [1], and has since then been adopted by Open Software Foundation (OSF) in their Distributed Computing Environment (DCE) [5]. Microsoft claims [42] that the RPCs in NT is completely compatible with the RPCs in DCE. This implies that a client, or a server, developed on an NT system, can interoperate with a server, or client, developed on a DCE-based system.

The RPC facility is different from both the WinSock API and the NetBIOS API in that it uses other IPC mechanisms as the transport medium. The following IPCs can be used:

- Named pipes
- NetBIOS
- WinSocks

4.2 Domains

Domains are basically a way of centralizing the management of user accounts. It consists of a primary domain controller (PDC) and zero or more backup domain controllers (BDC). All user management is carried out on the SAM database on the PDC and are replicated to the BDCs. This implies that, no matter where in the domain or how the administrator is managing domain information the changes are done to the PDC SAM database. If the PDC is down for some reason no changes of user information or adding of new accounts in that domain is possible. But users can log on as usual since the information in the database is replicated on the BDCs. A BDC can, however, be promoted to a PDC in the absence of the original one and thus take over its role. When creating a domain the PDC has to be the first server installed in this domain. This is due to the fact that the role of PDC or BDC is decided at the time the server software is installed on the computer. During this process there is a choice of becoming PDC or joining an existing domain. NT Severs that join an existing domain will become BDCs for that domain. A Server can also choose not to join a domain as PDC or BDC but will then, if it joins a domain, be treated as a workstation in the domain. These types of servers are sometimes referred to as member servers. A member server can not become a BDC or PDC without full reinstallation of the software. It is also impossible for a PDC or a BDC to leave a domain without complete reinstallation of the server software. According to [36] there is no way to avoid the reinstallation. This is due to the fact that a unique domain SID is created when the PDC is installed. This SID is later used as a prefix in the SIDs of the BDCs. So the servers will identify membership in the domain based on these SIDs and not based on names. These SIDs can not be changed. To join a domain a workstation has to have a computer account on the PDC. This account is created with the help of the System Manager program. The account is

later used in the login process, see section 5.4. After the creation of this account it is possible for a user to login to the domain from this workstation.

Domain are handled by the Windows NT Server Directory Service and according to [36] it provides the following functions:

- Single logon to file, printer and network applications
- Centralized administration
- Secure accounts database.

The single logon function is, however, a bit dubious since actual remote logons, see section 5.4.3, to every non-local service do occur. The user is just not aware or alerted to this fact. An NT workstation that we monitored spontaneously sent twenty remote logon requests during a two day period. The workstation had a domain user interactively logged on to it but no work was carried out. When a user wants to have access to a non-local service the redirector will perform a remote login to this service on the users behalf using the user name and the password given in the interactive logon, see section 5.3. The user will only be aware of this fact if s/he is accessing services outside the domain and s/he does not have an account there with the same username or password, or if s/he is trying to connect to the service under a different username by for example typing a name in the *connect as* box in the map network drive dialog.

4.3 Trust Relationships

Trust is a concept that lets one domain server authenticate a user to a domain server in another domain. It is mainly used to centralize user management in large companies. In this case, one domain is responsible for administering user information while the others mainly administer resources. It is however possible to think of other configurations where trust is needed. Microsoft have designed a number of Domain Strategies where trust plays an important part [42].

A trust relation is always one way, i.e. one domain trusts another domain to authenticate users. In the cases where two-way trust is needed, one simply established two one-way trust relationships, one in each direction.

To establish a trust relation, the administrator of the trusted domain has to create an account for the trusting domain, this account is the *interdomain trust account*. This is done by adding the trusting domains name and a password in *user manager for domains.* By doing that, an account for the trusting domain is created. This account will not show up as a normal account and it is not possible to log in to the account in a normal way. The account will, however, show up in the registry under *HKEY_LOCAL_MACHINE\security\sam\domains\account\users\name* as the name of the trusting domain followed by a $-sign.

To complete the trust relation the trusting domain has to set up a connection to the trusted domain. This is done when the administrator of the trusting domain adds the trusted domains name in *user manager for domains*, supplies the correct password and presses OK. By doing this two things happens. Firstly, LSA creates a secret object and a trusted-domain object where the password and the name and SID of the trusted domain is stored respectively, and secondly the server tries to logon to the interdomain

trust account on the trusted domain. This logon attempt will not succeed since these type of accounts does not allow interactive logons. Instead an error message is returned stating that this is an illegal operation. The error message will, however, tell the system that the account exists and that a trust relationship should be established between the two domain controllers. The events that takes place now is somewhat similar to the events taking place when a client is first started and a user logs on to a domain, see section 5.4.1. First a type of discovery process is carried out by the trusting PDC to locate the trusted PDC. The trusting PDC will then establish a secure channel to the trusted PDC. This channel is then used in the same way as the secure channel in a normal logon and any attempt, remote or interactive, to logon to a machin in the trusting domain will be handled as a pass-through-logon.

4.4 Shares

A share is basically an exported resource, i.e. a printer or a file system tree. A folder, a drive or a printer can be shared by an administrator, power user or server operator. There are two types of security on shares: user-level security or share-level security. In user-level security users are given access to a share dependent of their user name and password. This type of share is the one used in NT and optionally in Window 95. Every share has an ACL attached to it. Depending on the rights of the user, or a group of which s/he is a member of in this ACL, s/he can do different things in this share. If the share is a NTFS file system tree, the least of the rights in the combination of the ACL for the share and the ACL for the manipulated file or directory will be used when granting access to the user to this object. When the share is created it will by default have *Everyone full control* as its permissions. The shares that a computer export can be seen in the *network neighbourhood* or by typing **net view \\computername** in the command prompt. If the share name of a share is ended with a $-sign, it will not show up in any of the previously mentioned methods. In share-level security users are given access depending on if they can give the right password for a particular share or not. Every share has an optional password for reading and one for writing. This type of security is used in Windows for workgroups and Windows 95.

In NT, there exists a special kind of shares called administrative shares or system shares. They have all a $-sign at the end of their names and are created by the system when it is booted. Some of the shares that are created in this way are C$-Zx$, IPC$, PRINT$ and ADMIN$. The shares with volume names are the local drives of the exporting computer. IPC$ is the interprocess communication share which is used among other things for establishing secure channels on Domain Control (DC). PRINT$ is the local spooler service and ADMIN$ is the root of the local system directory, e.g. **c:\winnt**. The rights on these shares can not be set and the access rights on the drive shares are limited to administrators. These shares can be removed or stopped from being created at boot time. However, if IPC$ is removed from a PDC or a BDC no user will be able to logon to that domain using that DC. This might even take some time to discover since the last ten logons on a workstation is cached by default to enable logon if a DC should fail.

4.5 Server Message Block (SMB)

SMB is a application level protocol used by Microsoft for a number of thinks. Among those are authentication, RPC and the Common Internet File System protocol (CIFS).

In the NT environment, an SMB is carried on top of a NetBIOS over TCP/IP (NBT) session, including UDP as a carrier for NetBIOS as well. A Server Message has a structure as described in Figure 6. For a pseudo-C description, see appendix C.

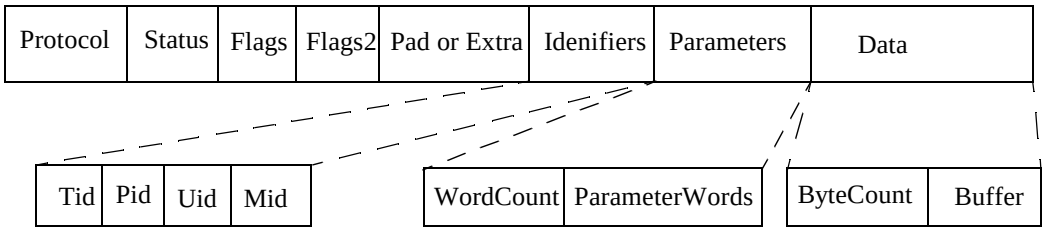

FIGURE 6. The structure of an SMB

The *protocol* field contains the identifier **0xFF, 'SMB'**.

The *Status* field contains the error codes if the request was not successful. Two types of error responses are supported. A DOS error type with error classes and error codes and a 32-bit error code.

The *flags* and *flags2* fields contain flags for different options such as the use of security signatures and long filenames, see appendix C for more details.

The *Pad or Extra field* contains the signature if SMB signing is used otherwise it will contain a padding field.

The *identifiers* field contains four types of identifiers: the tree identifier *Tid*, the process identifier *Pid*, the user identifier *Uid* and a multiplex identifier *Mid*. The *Tid* is used to identify the "mounted" share in case the user has more shares mounted on the same session. It also acts as the root of the mounted share so every request is relative to the *Tid*. The *Tid* is set by the server in reply to a *C TreeConnect request*. The *Pid* is used to decide which process, thread or task that opened a file or requested a lock. It is set by the Client. The *Uid* identifies the user to the server see section 5.4.3. Finally, the *Mid* is used to multiplex multiple outstanding requests on one session.

The *parameter* field contains a word count and a number of parameter words. The word count tells how many parameter words that follows. The number and types of parameters differs among the different commands, see [31] for more details.

Finally, the *data* field. It consists of a byte count and a buffer. The byte count tells how many byte of data that is present in the buffer and the buffer contains the actual data e.g records read from a file.

The SMB protocol has the ability to group together commands. This is called *AndX* batching. In this case, only the parameter and data fields from the batched command is added after the first command. *AndX* batching is permitted as long as the size of the SMB package does not exceed the negotiated size and all commands have to refer to the same identifiers.

4.6 Common Internet File System (CIFS)

Common Internet File System is a protocol designed by Microsoft to be used in a distributed file system. It relies heavily on TCP, NBT [49], [50] and SMB. Currently the protocol is an Internet Draft [31], and Microsoft hopes that it will be accepted as an Internet standard. This implies that the protocol currently is under constant revision and some of the parts presented here could be subject to change. The draft consists of a number of SMB commands used to connect to and manipulate an exported file system. The idea, at leased as implemented in NT, is that TCP and NBT sessions is used to carry the SMB commands.

To establish a CIFS connection to a file system, four types of connections needs to be established all in all. Assuming that all name resolutions are made. First, a TCP connection to port 139 is needed. Next, an NBT session is set up over the TCP connection. After that, an SMB session is established over the NBT session, and last a *TreeConnect* is made with the help of SMB commands. The first SMB connection is a logon on the server that exports the file system, and the *TreeConnect* is the equivalent of mounting the file system. The SMB commands is often attached back to back as a combined *SessionSetUpandX* and *TreeConnect* message. If the file system is shared in user level mode, the *SessionSetupandX* command will carry the user name and password of the user. If share level is used, the *TreeConnect* will carry the share level password for that share. If the authentication is passed, the result of the *SessionSetupandX* message is a *UID* that is used to identify the user in subsequent SMB messages and the result from the *TreeConnect* is a *TID* that is used for SMBs referring to the connected resource. The messages exchanged is described in Figure 7.

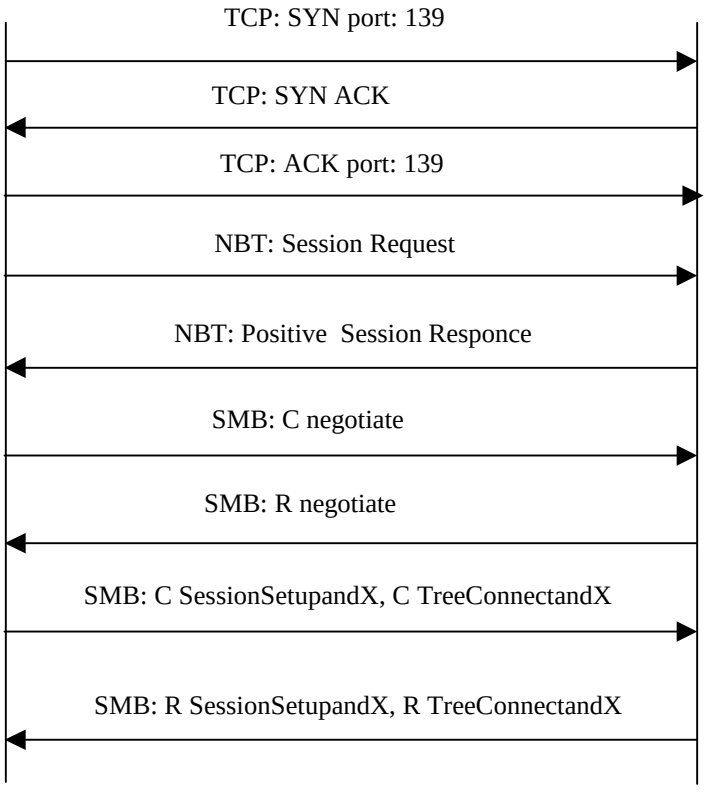

FIGURE 7. Connecting to an exported service

During the negotiation phase SMB is determining the authentication method to use, e.g. NTLM 0.12, LAN Manager 2. It is the connecting party that determines the authentication method to be used. The server will, however, always accept plain-text passwords regardless of the method proposed.

After the file system is mounted, a number of operations, e.g. open, read, write, can be performed on that file system. All possible commands are described in [31]. The messages involved in unmounting a file system and tearing down the connection is described in Figure 8. Multiple file systems can be individually mounted and unmounted over the same SMB session. It is the *TID* that keeps them apart. It is also possible to establish consecutive SMB sessions over the same TCP connection. This makes any protocol using SMB sensitive to brute force attacks or guessing games.

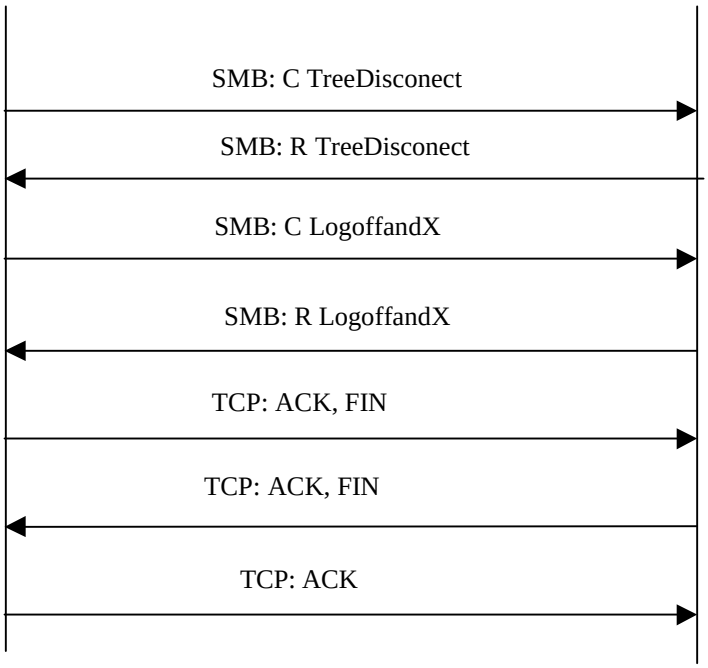

FIGURE 8. Disconnecting from an exported service

There are a number of things to be said about CIFS, or rather SMB, regarding security. First, it is possible to establish so called NULL sessions, i.e. sessions with no user name. This is frequently done by machines that exchange RPC calls through named pipes. Second, all logging in NT and most other checks is done on computer names and not IP addresses. This means that there is no way of telling from where a computer is accessing. We are fully aware of that IP addresses can be forged, but it is even simpler to forge a computer name, Samba's[1] *smbconnect* command even has a switch to set the computer name to an arbitrary value. This situation is further aggravated by the possibilities of using relays on the route to the server. In these cases, it will look like it is the relay that does the accessing even if IP addresses are logged. Third, the protocol is very loud mouthed and will give away quite a lot of information on what is going wrong. Fourth, there is the possibility of doing brute force attacks, due to the fact that the TCP session stays open even if the logon attempt fails. Last, all in all, there is to much trust

1. samba is a UNIX implementation of CIFS

in client machines behaving in a non malicious manner. In [29], there is a more in-depth description on some of the weaknesses in CIFS.

4.7 Remote Access Service (RAS)

Since version 3.1, NT server contains a service called RAS, short for remote access service. RAS, pronounced "razz", offers mobile workers dial-in functionality from a Wide Area Network (WAN), i.e. it opens your network to the world. Before a remote user can connect to the host or Local Area Network (LAN), the RAS has to be installed and configured on a local NT server. On this computer, there must also be some communication equipment(s), e.g. one or more modems if the WAN is an analog telephone network. For other types of networks, other types of communication equipment are necessary. The requirement on the remote client is support for the Point-to-Point Protocol (PPP), and a modem connected to the computer. The RAS client software gives support for PPP. Furthermore, RAS clients can also provide remote access to NetWare Communication servers. A sample configuration is illustrated in Figure 9.

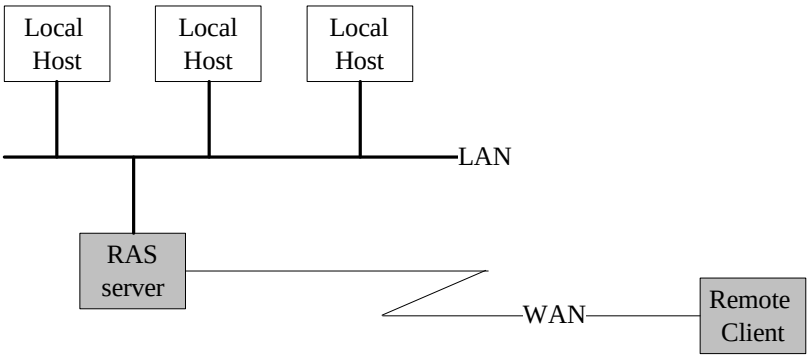

FIGURE 9. A sample RAS configuration

4.7.1 Remote Client Requirements

As mentioned above, the operating system requirement on the remote machine is support for PPP, which implies that both NT and Windows 95 clients, as well as UNIX clients are able to utilize a RAS server [44]. The maximum number of simultaneously connected clients that a RAS server can handle is 256. However, there exist other third party products that do not have this limitation.

4.7.2 Wide Area Network Connectivity

A client can connect to the RAS server in a number of different ways. The most common way is through a modem connected to a standard analog telephone line, which is also called Public Switching Telephone Network (PSTN) or Plain-Old Telephone Service (POTS). Instead of a single modem, it is also possible to have a modem pool on either the client or the server side. A PSTN connection gives mobility freedom, due to its worldwide availability.

If faster links are desired, Integrated Services Digital Network (ISDN) can be used. ISDN provides transmission speeds of 64 kbps (with one B-channel) or 128 kbps (with two B-channels). To be able to use ISDN a phone company must install ISDN lines at

both ends. Moreover, both sides must have an ISDN card in place of modems. The cost for ISDN equipment and lines is still higher than standard modems and phone lines.

Another possibility is connecting clients with a RAS server through the standard X.25 protocol.

Assume a company with two LANs at the same location, but not physically connected. Then, assume that there is one computer connected to one of the networks, that also need resources on the other. How could this be solved? One simple and cost effective method is to configure a RAS client on the computer that wants access to resources on both networks and then select a computer on the other network, on which a RAS server is installed. Further, connect these two computers with a RS-232C Null modem cable, see Figure 10. Now the RAS client in the figure is able to utilize resources on both networks.

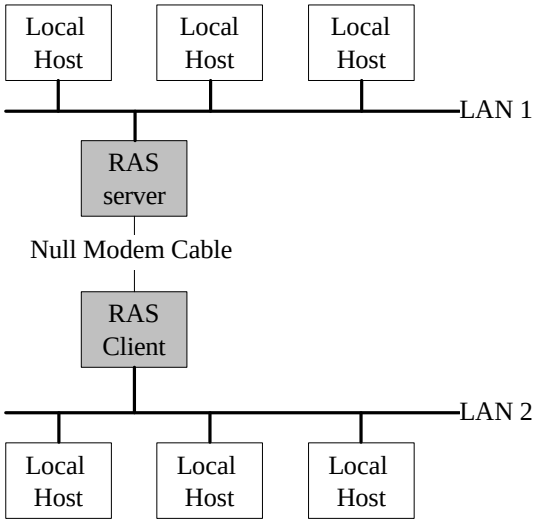

FIGURE 10. Connecting two LANs using RAS as gateway

4.7.3 Telephone Application Programming Interface (TAPI)

NT 4.0 provides a new interface called TAPI. This interface allows programmers to build programs, which are (more) independent, on how a specific modem works. With TAPI, various behaviors of a modem can be controlled. Modem vendors will be encouraged to write NT modem drivers [34].

4.7.4 Remote Access Protocols

RAS supports several modem protocols, including:

- Point-to-Point Protocol (PPP)
- PPP Multi-Link Protocol (MP)
- Point-to-Point Tunneling Protocol (PPTP)
- RAS protocol
- Serial Line Internet Protocol (SLIP)

PPP is a set of industry-standard framing and authentication protocols [53], [54], [55], [43] that enable remote clients to connect to remote servers over a WAN. PPP supports a number of transport protocols, including TCP/IP, NWLink and NetBEUI.

To understand PPP, we will now describe what will happen when a client tries to set up a connection to a RAS server.

1. Framing rules are established between the client and the server. This allows continued communication to occur.
2. The server then authenticates the remote user using one of PPP's authentication protocols, see the example in appendix A.7.
3. If step 2 succeeds, then the server is configured according to the remote client.

When all these steps are successfully completed, the remote client and RAS server can begin transferring data using for example NetBIOS, WinSocks or RPC.

Note that not all users on the server side are allowed to use RAS. After RAS is installed, all users are denied access to RAS. This is often referred to as default deny in the literature [22]. An administrator has to grant dial-in permissions for each user that is allowed to access the system from a remote site.

MP [57] is a protocol that can be used to increase the communication bandwidth between a remote client and a RAS server. The idea is to combine (or aggregate) a number of communications channels. For example, suppose that two PSTN 14.4 kbps modems are available on both the client and the server side. Then, by using MP a 28.8 kbps line can be provided.

PPTP is new to RAS in NT 4.0. This protocol allows clients to connect to a RAS server via the Internet. The protocol allows Virtual Private Networks (VPNs) to be built on the top of current networks. The problem associated with data transfers over public networks is maintenance of data confidentiality. In PPTP, this is handled through encryption of the connections. The encryption method is RSA's RC4, which use a 40-bit key. To connect to a RAS server with PPTP, two connections must be established. The first one is to the ISP (with PPP) or to a RAS server in order to establish connectivity with the network. The second establishes the tunnel through the Internet (PPTP packets are encapsulated in IP packets), se Figure 11. Indeed, both connections have their own security.

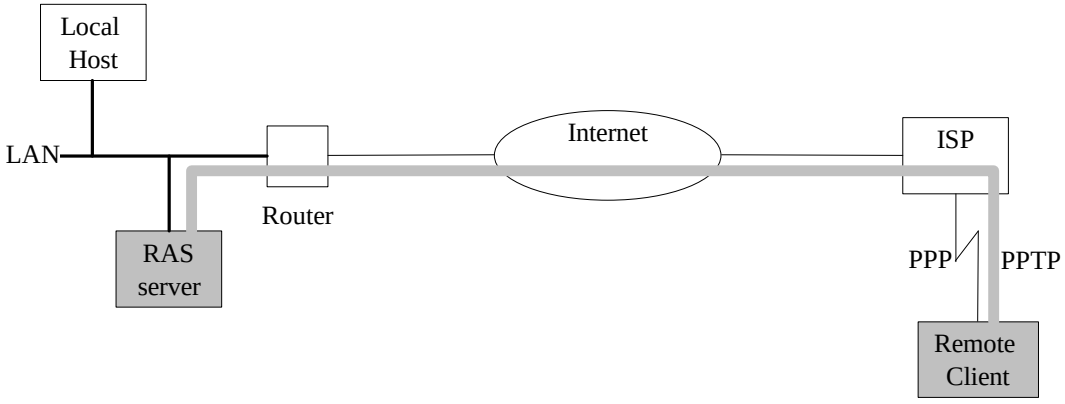

FIGURE 11. A typical PPTP connection

Another attractive feature with PPTP is the fact that it enables Internet as a backbone network for other protocols then TCP/IP. For example, both NwLink and NetBEUI can be encapsulated in PPTP packets.

The RAS protocol is an elder protocol used in NT 3.1, which is used in other operating systems' environments as well. Due to backward compatibility promises, it is still supported in NT 4.0, but we will not describe it further in this report.

SLIP [51], [52] is an old communication protocol used originally on UNIX systems. This protocol is supported in NT RAS clients, but it cannot be used to connect to a RAS server. However, it can be used to connect to, for example, an Internet Service Provider (ISP), assumed that the ISP supports SLIP. Note, SLIP supports only TCP/IP.

5. Security Features

NT provides a unified access control facility which applies to processes as well as other objects in the system. The access control is built around two components, an access token associated with every process, and a security descriptor connected to every object where interprocess access is possible. When a user logs onto the system s/he has to provide a username and a password to authenticate him/herself to NT. If s/he is accepted, a process and an access token are created for him/her. The access token is associated with the process. An important attribute in the access token is the Security ID (SID). Which is an identifier that identifies the user to the system in areas concerning security. A process inherits the access token of its creator. There are two reasons for maintaining access tokens [33]:

1. It keeps all the necessary security information in one place in order to speed up access validation.
2. It allows every process to modify its own security attributes, in a restricted manner, without affecting other processes in the system.

It might seem strange at first glance to let the process change its own security attributes, but normally a newly created process is started with all attributes disabled. Later when a thread wants to perform an allowed privileged operation the thread enables the necessary attributes before the operation is executed. Associated with every object that has the possibility of interprocess operation is a security descriptor.

5.1 Subjects and Objects

A subject in the NT system is a process with one or more threads that execute on behalf of a user and objects are all other NT resources that can be accessed and used by a subject. Note that incidentally a process in NT is also called an object. The privileges, user SID and group SID of a user's processes are stored in an access token held by the process. This token, called the primary token, is used to grant threads different types of access to different objects. Threads can also hold one additional access token called an impersonation token. This is a token given to the thread by another subject which allows the thread to act on that subjects behalf and is a full or restricted variant of that subject's primary token. The information in the token is compared to the information stored in the object's ACL. This comparison is handled by the Security Reference Monitor (SRM) and performed when the object is first opened by the process. The privileges defined and those entities granted these privileges are stored in the Local Security Policy Database managed by the LSA which is also responsible for creating the primary token for a user during logon. The primary token (or restricted variants of it) will later be inherited by the processes created or used by the user. Since the first token is created during logon, it is not possible to add or remove privileges while this user is logged on (or rather it is possible to change them but the change will not take effect until next logon). However, they can be enabled or disabled.

5.2 Access Control Lists (ACLs)

As mentioned before, every named object in NT together with unnamed process, thread or token object has a security descriptor in its object header. The security descriptor have information about logging for and access to the object. The access con-

trol is implemented by an ACL. The ACL consists of a list of Access Control Entries (ACEs). An ACE contains a security identifier (SID) and the operation that the owner of this SID is allowed (or disallowed) to perform on the object. ACEs that disallow operations is generally placed before ACEs that allow operations in the ACL.

The ACL is tied to the object when it is created in one of three different ways. One way is for the creator of the object to specify an ACL in the call that creates the object. If this is not the case the system will try and find out if the object can inherit an ACL from a directory object. As a last resort the system will use the default ACL in the creators access token, see section 5.1.

When a process wants access to an object it first has to open the object. In this open call, the process must specify what operations it intends to perform on the object. The system will then search the ACL of the object to try and find an ACE that contains the SID of the caller (or any group of which she is a member) and allows or disallows the requested operations. The search is done on a “first match” basis and is terminated as soon as a match is found. If the call is a success an handle to the object is passed to the process. All subsequent calls to the object is done through this handle so no further checks are done as long as the process only performs the operations specified in the open call. If it wants to perform new operations the object needs to be opened anew. This means that it is impossible to revoke access rights to an object from a process as long as that process has an active handle for that object.

5.3 User Logon and Authentication

The logon procedure in NT is fairly complicated and involves a number of executives and protected servers to authenticate and grant privileges to a user. The details of the logon procedure are given in [43], and the following is a shortened version of the events that take place. The coordinator of the process is the WinLogon server. When the system is in its initialization phase, WinLogon registers itself at the Win32 server as the login process, see Figure 12, and ensures that no other process has access to the WindowStation object created by Win32. This is done by assigning a security descriptor to the WindowStation with only one entry - the WinLogon SID. It establishes a link to LSA, where it also registers as the logon process, makes the WinLogon Desktop active, and waits.

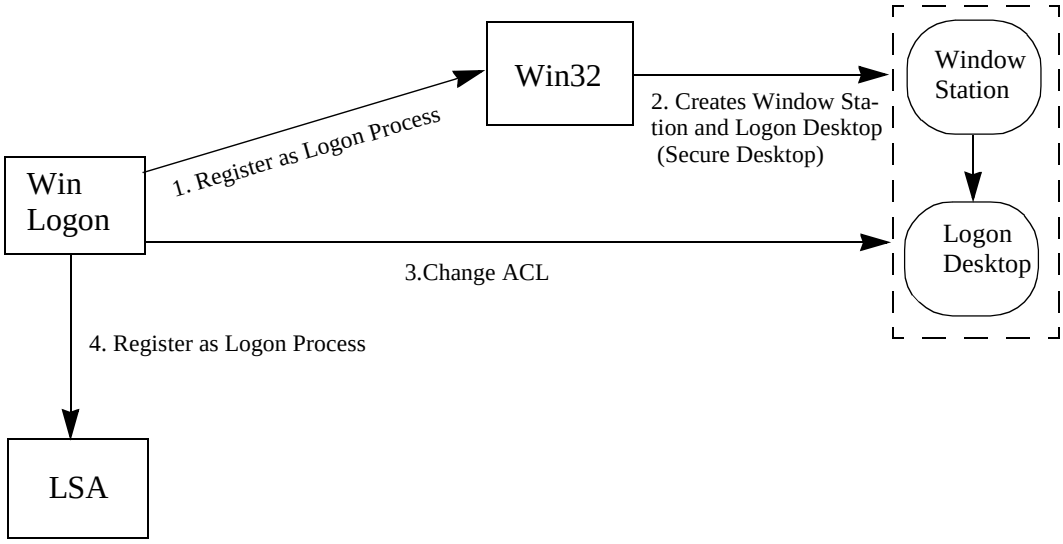

FIGURE 12. Schematic view of the login procedure: step 1 (Numbers state chain of events)

When a user presses the Ctrl-Alt-Delete key combination (called the Secure Attention Sequence (SAS)) WinLogon initiates the identification and authentication process and brings up a secure Desktop that prompts for username and password, see Figure 13. The password is collected by Win32 which encrypts the password with an internal encryption, later decrypts it and passes the password to WinLogon, see Figure 14. When WinLogon receives the username and the password it collects the domain of the user, creates a unique logon SID, creates and suspends a new process, and calls LSA, see Figure 15.

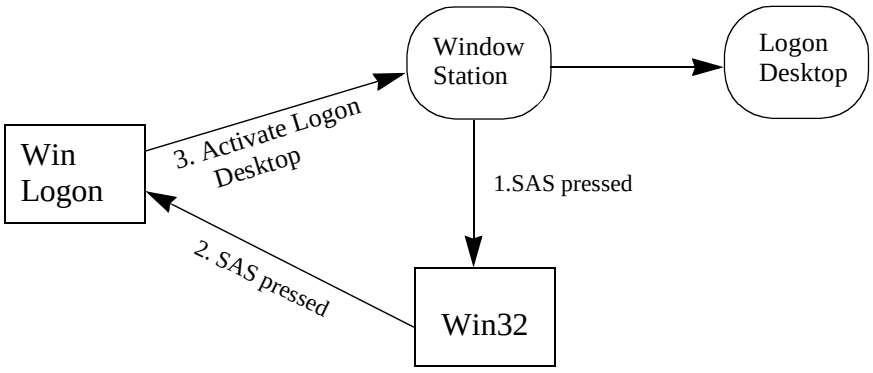

FIGURE 13. Schematic view of the login procedure: step 2 (Numbers state chain of events)

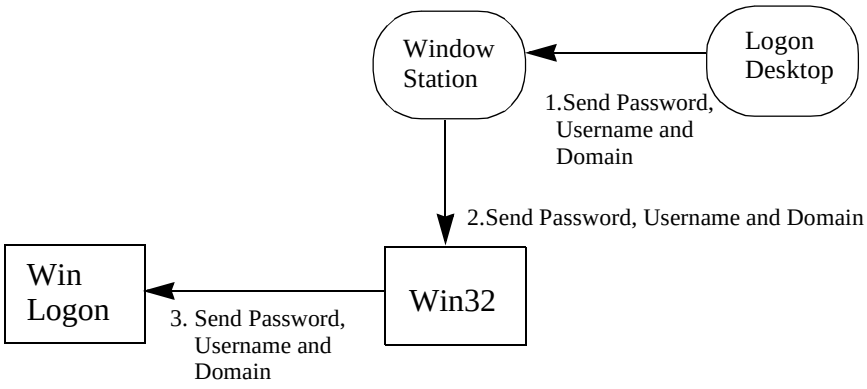

FIGURE 14. Schematic view of the login procedure: step 3 (Numbers state chain of events)

The parameters passed to LSA in the call are those collected and created above and the Authentication package that WinLogon wants to use. Since it is possible to use more than one Authentication package this information is important. Before the password is passed to LSA it is once again encrypted with an encryption known only to WinLogon and the Authentication package.

Once LSA been called it calls the Authentication package. This is done because LSA can not use some of the parameters because they are encrypted. It is instead the Authentication package that verifies the username and the password. This is done by retrieving some information from the SAM database, including the hashed password, the user SID and the group SIDs that are associated with this user, see Figure 15.

The authentication is done by comparing the two hashed passwords with each other. The hashed clear-text password, the username and the hashed case-sensitive password is then stored in a variable supplied by LSA.

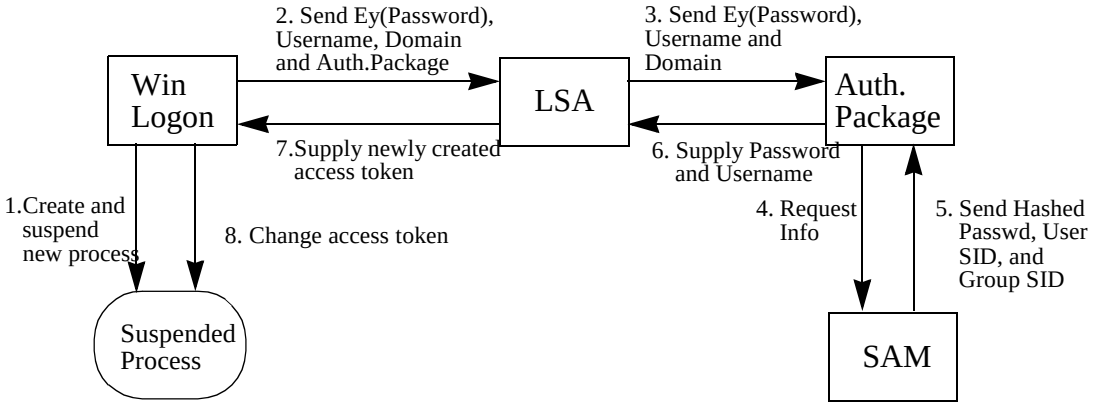

FIGURE 15. Schematic view of the login procedure: step 4 (Numbers state chain of events)

If the authentication package part of the login process succeeds the LSA will check the privileges held by the user and the groups of which s/he is a member and determine if the user is allowed to perform this type of login (there are three types available: interactive, network and service logons). If s/he has the right privileges the logon SID men-

tioned above is added to the list of local groups and a default ACL is constructed for the user together with the user's primary token, see section 5.1. A handle to the primary token is passed to WinLogon which uses the handle to create an ACL for the user in the DACL of the WindowStation and to set a new primary token on the suspended process, see Figure 15. After that, the WinLogon creates an application Desktop and a screen-saver Desktop and change the DACL on both of them so that WinLogon and the user have access to the two desktops.

If the authentication should fail in any way the LSA will notify the WinLogon Server and the server will display an error message through the Desktop and prompt for new information.

5.4 Network Logon

There are basically two types of network logons, interactive logon and remote logon. However, there are some variants of remote logon. An interactive logon to a domain on a machine that is not a DC for that domain, will result in a pass-through-logon. Most of the remote logon variants and the pass-through-logon rely on a service called netlogon.

5.4.1 Netlogon

The *netlogon* service is started when the machine is booted. To be able to authenticate a user in a domain *netlogon* needs to have contact with a domain server for that domain. This is accomplished through the discovery process. During this process *netlogon* will send a query to the NetBIOS name service and ask it to send the name of a domain server for the needed domain. The reply is cached on the machine for subsequent use. The next thing *netlogon* will do is to try to establish a secure channel to the DC, see Figure 16.

Every DC in the domain will have a user account for every computer that belongs to the domain. This account will show up in the registry of the DC in HKEY_LOCAL_MACHINE\Security\Sam\Domains\Accounts\Users\Names***
where *** is the name of the computer followed by the $-sign. Associated with this account is a password. The default start password is the lmhash of the lowercase computer name, but the client will try and change that password as soon as possible. The client will have the password information in the LSA secret object $machine.acc. The creation of the secure channel is largely undocumented (or very hard to find documentation on) so there could be some misunderstandings of the mechanisms on our part.

Below we will describe the events that takes place as we have understood them. The netlogon service will create the secure channel by connecting to the IPC$ service on the DC using a NULL SMB session, i.e. no username or password is used. After this connection an RPC pipe will be opened over this connection and both the DC and the client will throw challenges to each other using the RPC pipe. After the challenge both the DC and the client will construct a session key, see appendix A.1. This key is later used to encrypt some of the conversation between the DC and the client. Next the client will calculate its credentials, see appendix A.2 and send them to the DC. The DC will calculate the same credentials and compare the two. It will also calculate credentials of its own and send those to the client. After these steps the secure channel is set up. And

netlogon will wait for a request to log on a user. An example of the event stated above is described in appendix B.

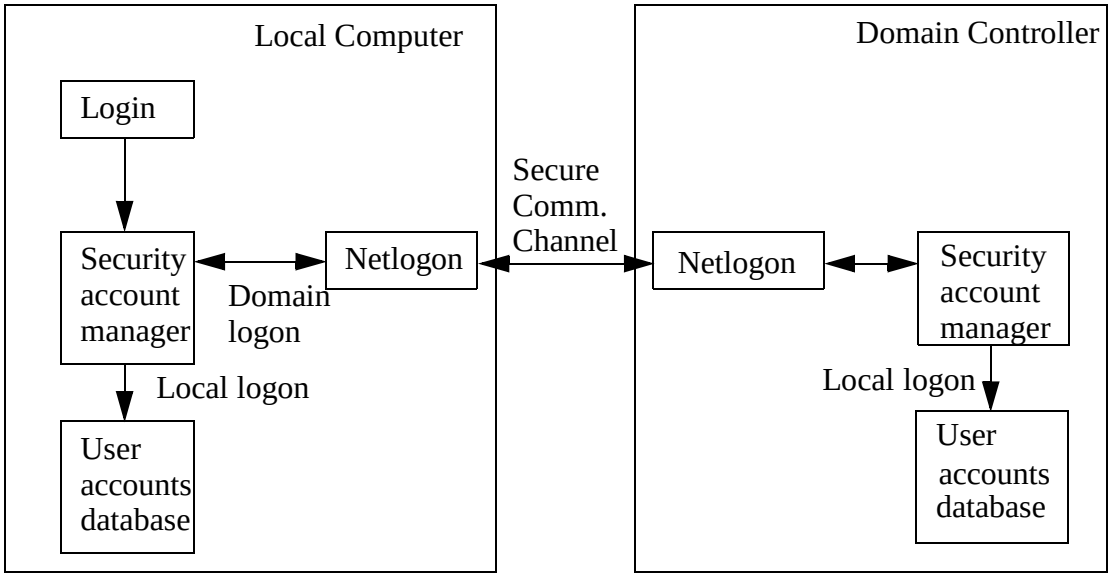

FIGURE 16. Passthrough authentication in NT

5.4.2 Pass-through Logon

When a user is trying to logon to a computer that is not a DC and gives a domain name that is different from the name of the computer a pass-through logon will occur. The steps involved in this type of logon is similar to does described in, see section 5.3, the only difference is that the authentication package will ask the DC for the required domain to authenticate the user by using *netlogon*. The events that occur are the following (assuming *netlogon* already has a secure channel established to the DC):

1. The authentication package will send the hashed password, the domain name and the user name to *netlogon*.
2. *Netlogon* will send a NetLogonSamLoggon RPC message to the DC using the secure channel. This message will contain among other things the 16 byte RC4 encryption of the passwords, both LM and NT-native, using the session key as the encryption key, the domain and user names and a time stamp. See appendix A.4
3. The DC will compare the information it got with the information stored in its SAM database. It will then reply and if the logon was successful send a copy (almost) of the SAM entries for that user to the netlogon process on the client.
4. *Netlogon* will pass the information it received to the authentication package and the logon process will continue as described earlier.

5.4.3 Remote Logon

This type of logon has unfortunately been called a number of things, remote logon, network logon and secondary logon. This type of logon will occur when a user accesses resources on other computers than the on s/he is interactively logged in to even if that computer is part of the domain. In this case netlogons secure channel is not used

between the client and the computer exporting the service, but it might be used between the exporting computer and its DC. The events that occur are the following:

1. The client tries to setup a SMB connection to the exported service on the remote computer.
2. If challenge/response is used the remote computer will send the client a 8-byte challenge (even though some authors claim that a 16 byte challenge is used [64], we have always got a 8 byte challenge and [45] also clearly state that a 8 byte challenge is used). Otherwise the client will send the "plain" hashed passwords or even clear text ones in step 3 depending on the result of the SMB negotiation, see section 4.6.
3. The client will calculate two 24-byte strings using the challenge and the LM and NT passwords, see appendix A.4, and send those in an SMB message to the remote computer.
4. Netlogon on the remote computer will send this information together with the challenge, the domain name and the username to the *LsaUserAPI* and it is further passed to the authentication package.
5. The authentication package will try and locate the user account for the user. If it is in the domain the netlogon secret channel will be used to authenticate the user otherwise a local account is used. If no account matches the user the guest account, if available, will be used.
6. If the user passed the authentication process, an access token will be created on the remote machine. A UID will be created and associated with the access token. The UID is then sent to the client. The remote machine will also send a session key. This session key is later used together with the 24-byte response to sign any further SMB messages, see appendix A.6.
7. The clients redirector will store this UID and use it as an identifier in any further SMB command blocks sent to the remote computer.

5.5 User Rights and Account Policies

As mentioned in section 3.6, only legitimate users are allowed to logon to an NT system, and each user belongs to one or more groups. User accounts, except Administrator and Guest, are defined by users that have the required permissions to create new accounts, e.g. users in the Administrator group. In addition to account information, e.g. user name, password and full name, a set of attributes called Rights can be specified. They define what a user (or group) can do on an NT system. The rights are:

- Access this computer from network
- Acts as part of the operating system
- Add workstation to domain
- Back up files and directories
- Bypass traverse checking
- Change the system time
- Create a pagefile
- Create a token object

- Create permanent shared objects
- Debug programs
- Force shutdown from a remote system
- Generate security audits
- Increase quotas
- Increase scheduling priority
- Load and unload device drivers
- Lock pages in memory
- Log on as a batch job
- Log on as a service
- Log on locally
- Manage auditing and security log
- Modify firmware environment values
- Profile single process
- Profile system performance
- Replace a process level token
- Restore files and directories
- Shut down the system
- Take ownership of files or other objects

Most of the rights are self-explanatory, for example a user account with the *Log on locally rights* specified is allowed to logon to the system from a local machine. NT has a standard policy for assigning rights to certain groups. For example, on an NT Workstation everyone has by default the right to logon locally, while at an NT Server only Administrators and Operators are able to do that.

The account policies regulate password restrictions and account lockouts. These policies are first defined by the user that creates the account, but can later be changed by a user with permissions do to such a task. The following password options can be set:

- Maximal password age, sets limitations for how long time a password can be used before it has to be changed.
- Minimal password age, sets limitation for how long time a password has to be used before a user are allowed to change it.
- Minimal password length, sets limitation for how many characters a password must at least consist of.
- Password Uniqueness, where an NT system can be configured to remember old passwords, and enforce users to choose new passwords that have not recently been used.

An account can also be specified to be locked out after a given number of unsuccessful logon attempts. If lockout is used, there are a few options:

- Lockout after n bad logon attempts, where n must be assigned a positive integer value.
- Reset count after m minutes, where m specifies how many minutes shall pass before the bad logon counter is cleared.
- Lockout duration, where forever (i.e. until an administrator unlocks) or duration in minutes are the possible choices.

5.6 Port Filtering

The TCP/IP protocol suite was not specified to be particularly secure. Lately, a number of successful network attacks have been described. Many of these intrusion attempts have utilized different protocols in the TCP/IP family, e.g. TCP and UDP.

A common way to minimize weaknesses in a system is only to permit services that are proved secure and necessary, see [22] for a detailed discussion on this topic. In NT, blocking communication to both TCP ports and UDP ports is possible. This implies that a system can be configured to accept only packets sent to specific ports on which secure and necessary servers listen. This feature is referred to as TCP security in NT terminology.

5.7 Security Features in RAS

As already stated, RAS opens a network to the world. Some users will appreciate this feature while some will not. For example, sales persons and consultants might do their job more efficiently if they have the possibility to access files and other resources from the office when they are out on the field. Administrators as well as people responsible for security know fairly well that this feature may also be taken advantage of by an intruder. For the latter reason, RAS has a variety of mechanisms to protect against attackers.

5.7.1 Authentication

In PPP, three different authentication protocols are supported. The first and least secure option is enabling the RAS server to allow clear text passwords. In this case, the Password Authentication Protocol (PAP) is used. A more secure authentication protocol is the Shiva Password Authentication Protocol (SPAP). Unlike PAP, SPAP encrypts passwords before sending them over the wire. SPAP is used if either side of the communication uses a product from Shiva, Inc. However, the most secure authentication protocol [64] is the Challenge-Handshake Authentication Protocol (CHAP), which uses DES-encrypted authentication. This protocol is described in appendix A.7.

5.7.2 Callback

With the callback facility, a user will be called back after s/he has provided a valid user name and a valid password. This provides added security, since NT now knows from where the call came from. There are two different kinds of callbacks:

- Predefined number, which implies that a predefined number will be used in the callback.

- Set number, which enables a user to set the number at which s/he will be called back at login. Clearly, this method is less secure then the first one, but at least it offers tracing of calls.

5.7.3 Default Deny

After RAS has been installed on a server, it must be configured. The first thing to do is to start RAS. By default every user is denied access to the system through this service. An administrator must grant dial-in permission for a user to be able to remotely utilize the server, which is done on a peer user basis. This can be done either with the Remote Access Admin utility or with the User Manager utility.

5.7.4 Access Restrictions

In the installation procedure, several access restrictions can be enforced. First, dial-in and dial-out restrictions must be specified. Three options exists:

- Dial-out only
- Receive calls only
- Dial-out and Receive calls

A RAS server must at least be able to receive calls, while a RAS client must be able to Dial-out. Second, select which protocol(s) (NetBEUI, TCP/IP and/or IPX) can be used for a connection. Third, select and configure one or more protocol(s) (NetBEUI, TCP/IP and/or IPX) that are to be allowed on the server. Fourth, decide whether the remote users shall have access to the entire network or to the computer running the RAS server only. Fifth, select which authentication method shall be used. There are three options:

- Allow any authentication including clear text (PAP, SPAP or CHAP are allowed)
- Require encrypted authentication (SPAP or CHAP are allowed)
- Require Microsoft encrypted authentication (only CHAP is allowed)

One of the above must be selected. Another setting, which can be specified, is whether or not data encryption is required.

5.7.5 PPTP Filtering

If PPTP filtering is enabled on a RAS server, the only way for clients to connect to the server is through PPTP. The server will refuse other types of connections.

5.8 Auditing

The auditing in NT is handled by SRM and LSA together with the Event Logger. Different types of events are grouped into event categories and auditing is then done based on these groups. There are seven types of event groups. These are: System, Logon/Logoff, Object Access, Privilege Use, Detailed Tracking, Policy Change and Account Management. For details on event groups see [43] and Table 4. If auditing applies and what is to be audited is determined by the *Audit Policy* which is handled by the LSA and given to the SRM by LSA.

The auditing is based on audit records constructed on request from the responsible sub-system or server by SRM (in some cases by LSA). Requests from the executive is always carried out while servers need the Audit privilege for SRM to honour there request. The request must be sent for each occurrence of an event. The audit record is then sent to the LSA, which in turn sends it to the Event Logger after it expanded some fields and compressed others. The final writing to disk is made by the Event Logger.

The audit record contains the following header information [43]:

- Time event was generated
- SID associated with the process causing event to be generated. If impersonating this is the impersonating SID (both primary and impersonation SID are stored in the body)
- Name of component or module that submitted the event
- Event ID
- Event type: error, warning, success, information, success audit, failure audit
- Event Category
- Computer

TABLE 4. Audit event types

Category	Event Type	Constructor
System	System Restart	SRM
	System Shutdown	SRM
	Authentication Package Loaded	LSA
	Registered Logon Process	LSA
	AuditLog Cleared	SRM
	#Audits Discarded (due to full internal queues)	SRM
Logon/Logoff	Logon Successful	LSA
	Unknown User or Password	LSA
	Time Restricted Logon Failure	LSA
	Account Disabled	LSA
	Account Expired	LSA
	Invalid Workstation	LSA
	LogonType Restricted	LSA
	Password Expired	LSA
	Failed Logon	LSA
	Logoff	LSA
Object Access	Open Object	SRM
	Close Handle	SRM
Privilege Use	Assign Special Privileges	SRM
	Privileged Service	SRM
	Privileged Object Access	SRM
Detailed Tracking	Process Created	SRM

TABLE 4. Audit event types

Category	Event Type	Constructor
	Process Exit	SRM
	Duplicate Handle	SRM
	Indirect Reference	SRM
Policy Changes	Privilege Assigned	LSA
	Privilege Removed	LSA
	Audit Policy Changed	LSA
Account Management	Domain Changed	LSA
	User Changed	LSA
	User Created	LSA
	User Deleted	LSA
	Global Group Member Removed	LSA
	Global Group Member Added	LSA
	Local Group Changed	LSA
	Local Group Created	LSA
	Local Group Member Removed	LSA
	Local Group Member Added	LSA
	Local Group Deleted	LSA

6. Utility Programs

Utility programs are programs designed for the benefit of someone or something. Security analysis programs are made to help the administrator keep system security at an acceptable level, information retrieval tools are meant to help the user find relevant information, maintenance tools are for the aid of the maintenance engineer, etc. However, there is a dangerous duality with these programs: they can also be used to put somebody in a better position to penetrate the system or even cracking it directly. This is well-known, at least in the hacker community and among system administrators. The problem is further aggravated by the fact that many of these programs are available from Internet as public domain software.

In appendix D, we present some utility programs that have been useful to us in our study of NT. For example NTFilemon was an excellent program that gave us the opportunity to learn more about the activity in the file system, when the system was up and running. Many of the utilities presented are publicly available on the Internet.

6.1 Different Types of Utility Programs

Utility programs are designed to assist system administrators and normal users in their work. Some are used for monitoring the activities in a computer or on a network, others for security analysis, and still others for debugging. In this section, we will give examples of such utilities, with focus on functionality and the intended usage.

6.1.1 Security Analysis Programs

Security analysis programs are designed to help the system administrator (primarily) to improve or verify security on the system. Typically, these programs analyze the system to detect security flaws, and the idea is that the administrator shall find these flaws and remove them. Typical examples are *Tiger* and *Cops* [22], which are tools for internal system analysis looking for weak spots in UNIX. *SATAN* and *ISS* for UNIX and *IS* for NT perform the same thing but are taking a networked approach.

Some programs focus on password security, ensuring that it must not be possible to reveal a password by making a brute force attack or a dictionary attack. Such programs are *Crack* for UNIX and *L0phtCrack* for NT, even if some sources claim that the latter was developed for cracking purposes, to show that (even) a NT password is breakable.

6.1.2 Information Retrieval Programs

Information retrieval tools basically come in two flavors, those intended for the system administrator and those intended for the normal user. The first type typically collects and displays information about the status and the activities of the system. A system administrator needs this information primarily to verify that the system and its components are behaving as expected. These types of utilities are mostly referred to as monitors. Examples of this category is *NTRegmon,* a monitor for registry activity in NT, *top,* a monitor for processes in UNIX and *NetMon,* a network monitor.

The second type gives information about users and programs. One of the most famous utilities in this category is the UNIX *finger* program.

6.1.3 Maintenance programs

The purpose of maintenance programs is twofold: to alter system functionality, or to attend bugs and shortcomings. Among the former we find programs for recovering a computer after system failure or when it fails to boot, possibly due to a faulty configuration. An example of such a utility is *NTRecover*.

NTLocksmith is a program that circumvents the NT security system to allow replacement of an administrative password in case it was lost. Other examples are *netinstall* for UNIX and *Ghost* for NT, which are tools that can be used to install a new operating system on a client machine over the network.

6.2 The Source Code Problem

Many of the public domain programs have, what we might call, open source code which is a nice way of learning how they work and how to construct these types of tools. One of the problem here is that some of these tools rely on undocumented features or loopholes in the operating system. All of these types of programs will teach an attacker how to construct software to exploit the hidden functionality and loopholes without knowing much about the operating system or, in some cases, programming.

Another problem is that some, seemingly harmless, utilities could be quite dangerous after modification. An example of the latter is *Ctrl2Cap*. With *Ctrl2cap* for NT it is possible to convert the control key to the shift key, which can be useful for people that migrated from UNIX to NT and are used to a different location of the control key. Using this principle it is possible to catch and manipulate any key on the keyboard, which makes it possible to sniff passwords when they are entered.

7. Vulnerabilities

We have found several sites on the Internet that present programs that utilizes flaws that endangers the security of the NT operating system. In this section we will describe some of these programs and use them in order to gain unauthorized access or information from an NT system. Together with the descriptions of how the program works there is a description of how we intended to use the program and the reward we hoped to get as well as the results we really got. In the end of this chapter, we have tried to classify the weaknesses according to the taxonomy presented in [66].

7.1 Methodology

In this report, we try to find security problems within NT, and to prepare the ground for further investigations. We have mainly concentrated on studying the relevant literature. The sources of information are published books, Usenet newsgroups, mailing lists and other material published on Internet. Firstly, we tried to get a general view on how NT is constructed and works. Secondly, we looked deeper into four areas of the system: file system, account management, access control and networking. Thirdly, we tried to find as many documented weaknesses as possible by searching the World Wide Web (WWW) and newsgroups. All programs that we found were tested on our target system to verify that they did indeed do what the authors claimed. In some cases certain privileges were required. On those occasions, the program was tested both with and without these privileges.

7.2 Experimental System

The target systems used to test the software have been standard PCs. Both Windows NT 4.0 Workstation and Windows NT 4.0 Server versions of the operating system have been used. The out of the box configuration with Service Pack 3 (SP3) have been used, if nothing else is explicitly stated. The network used has been a standard 10Base-T Ethernet network. The user profiles used have all been members of the standard user group except when special privileges have been needed. On those occasions we have also tested them as Administrator.

7.3 Known Security Problems

There are a number of known security problems that are general and not specific for the NT system. They are, however, interesting in this report since some of them do appear in the NT operating system. We believe that the reasons for this are the following: some of them were introduced by mistake (e.g. missing parameter checks) others in order to achieve the customers demand for backward compatibility and flexibility (e.g. foreign file systems) and some because they where not known at the time of implementation (e.g. collisions in MD4). In the following paragraphs, we will describe and explain some of these problems.

7.3.1 Installation Problems

Security weaknesses may be introduced in various stages, i.e. in the requirement specification, in the design, in the implementation, or when the system is eventually

installed. In these respects, NT is not different from other operating systems. In this section, we focus on weaknesses introduced in the installation procedure.

It is rather simple to install NT on a computer from the box. However, such installation is not secure. The system administrator has to perform a number of tasks. First, install the latest Service Pack announced by Microsoft. Second, fetch and install hotfixes for problems found after a Service Pack has been shipped. Third, since the standard configuration of the system is rather relaxed, probably due to the fact there is less complaining if things work right away, the system administrator really needs to go through the registry carefully after an installation and change the configuration to suite her/his needs. Microsoft could definitely have made the standard installation more rigorous, especially the Everyone group could be more restricted.

7.3.2 Collisions in MD4

MD4 is a oneway hash function developed by RSA Inc.. It is mainly used to create hashed checksums in mail systems for digital signatures where a variable sized text is hashed into a 16 byte value. In the context of digital signatures constructed by checksum it is important that different texts do not produce the same checksum, i.e. all the checksums are unique or it is at least impossible to alter a given text and still produce a checksum that is identical to the one for the original text. This was thought true for MD4 until Hans Dobbertin discovered collisions, i.e. two texts producing the same checksum, in the algorithm [21]. He also showed how this could be done fairly easily with any text as long as it had a certain structure [25]. This lead to the conclusion that MD4 is to be considered unsafe and a recommendation to stop using it.

In NT, MD4 is used as the main encryption algorithm for passwords. The reason being that it is considered impossible to decrypt a oneway hash function. The encrypted password is compared with the stored encrypted password to identify the user at logon, see section 5.3. However, since only the encrypted passwords are compared and since collisions have been detected in MD4 it is possible, but probably unlikely, that two different passwords can give access to the same user account. This in turn decreases the number of passwords that need to be tested in a brute force password attack, with how much is hard to tell, and therefore weakens the encryption.

7.3.3 Parameter Checks in System Calls

One of the oldest programming errors in the book is insufficient error control. This includes checks to make sure that the value of variables and parameters are within the allowed range. In NT 4.0, as well as 3.51, there exist system calls with insufficient validity checks for parameters. These system calls make the system crash if they are called with parameter values outside the allowed range. One program which we found (NTCrash) is primarily designed to detect and log system calls with these kinds of flaws. However, the program can also be used to exploit these weaknesses, see section 7.5.1. Hopefully, Microsoft will correct these programming errors in the near future.

7.3.4 Undocumented System Variables and Functions

An operating system exports a number of system functions, often called system calls, and variables to be used by application programs.

We have found a program package (NTExport) for NT that can be used to generate a kernel mode device driver library, which includes undocumented kernel functions and variables. We are not aware of any program that uses this package for intrusion purposes. However, there is a program called CacheMan, which relies on this package, that can be used to tune the file system cache. For more information about NTExport, see appendix D.2.3.

We can see two different reasons why Microsoft kept all this undocumented. One reason may be that they might gain competition benefits vis-à-vis other vendors, when application programs are developed. The other possible reason is that they may not be aware of what is exported and what is not. From a security point of view, we believe that the latter would be most serious.

7.3.5 Plain-text Passwords over the Network

NT supports eight different variants of authentication for backward compatibility reasons. The party that suggests which variant to use is the client in the SMB_C_NEGOTIATE message. The elder of these variants sends plain-text passwords over the network and some of the newer ones have weak encryption schemes. The server can always try and suggest a higher variant but it is the client that, unless there is a share-level versus user-level incompatibility, sets the level of security. According to among others [29] the server as default will always accept plaintext passwords as valid authentication. This could be used in man-in-the-middle attacks to snatch passwords directly from the network.

7.3.6 Non NTFS File Systems

Each file system has a device object that is created when the file system is loaded. A thread is able to access a file system, if it has traverse access rights to the device object representing the particular file system. The ACL on each device object is by default set to traverse access for *everyone*, and read/write for administrators. Subsequently, file objects for an individual file may be created. File objects for, say, FAT files are based on the ACL on the file system device object. This is not the case for file objects for NTFS files and directories, because these do have a security descriptor [43].

7.3.7 System Initialization

Another problem is the fact that a PC easily can be booted by other operating systems than NT, which may lead to intrusions. For example, a PC that is bootable from a floppy drive can be booted by MS-DOS, and after that, the attacker can use some utility program, e.g. NTFSDOS, see appendix D.3.1, or NTRecover, appendix D.3.4, to access NTFS, including the SAM database.

7.4 Suggested Attacks

There are a number of attacks that have been suggested by different persons but, to our knowledge, has not been proven by concept code or the equivalent there of. There are also some attacks which we, for some reasons, have not been able to perform and eval-

uate. In this section, we will present some of these attacks. Even if we have not verified them, we are convinced that most of them would work if they were to be carried out.

7.4.1 Weaknesses in SMB and Challenge/Response

There are some weaknesses in the challenge/response mechanisms that could be used to carrie out attacks.

The first attack is a man-in-the-middle attack, suggested by David Loudon, that is based on the fact that the server always accepts plain-text passwords and that the client can be fooled into using a weaker authentication protocol. It is carried out as follows.

Scan the network for negotiate request messages. When a request is intercepted, send a reply to that request masquerading as the server that is the recipient of the request, and claim that you only understand an authentication variant that uses plain-text passwords. Snatch the plain-text password from the network when the client later sends the session setup message. None of the attacked parties will know that the attack has taken place, since the user on the client is not aware of the fact that the security has been lowered, and the server sees nothing wrong in the fact that the client sends plain-text passwords, even though it originally suggested an encrypted variant. There is a program called *c2myazz* that demonstrates this weakness.

The second attack is also a man-in-the middle attack. It is described by Dominique Brezinski in [24]. This attack relies on the fact that there is nothing that prevents an attacker from masquerading as a arbitrary host. [24] describes the attack as follows (Mallory is the attacker and Alice the user):

1. Mallory sends a connection request to the host
2. The host responds with a random string
3. Mallory waits for Alice to send a connection request to the host
4. When Alice sends a connection request to the host, Mallory forges a response to Alice containing the random string sent to Mallory by the host
5. Alice encrypts the random string using the hash of her password as the key and sends it to the host
6. Mallory intercepts (or just copies it off the wire) Alice response and repackages it as a response to the connection request made in 1 and sends it to the host claiming to be Alice
7. The host looks up the hash of Alice’s password in the security database and encrypts the random string sent to Mallory
8. If the host’s encrypted string matches the encrypted string sent by Mallory, claiming to be Alice, to the host, Mallory is allowed into the system under Alice credentials

It is interesting to compare this attack with Microsofts suggestions in [46], on how a pass-through authentication should bee carried out in a CIFS environment. The Draft state that the following events should take place (the Domain Controller is the server holding the authentication database, and the CIFS server is the server that the user wants to connect to):

1. The CIFS client sends a negotiate SMB to the CIFS server

2. The CIFS server verifies the cached Domain Controller name
3. If the cached name is invalid, the CIFS server does a Domain Controller Discovery
4. The CIFS server sends a NEGOTIATE SMB to the Domain Controller
5. The NEGOTIATE response along with the challenge is saved by the CIFS server
6. The CIFS server sends a NEGOTIATE response (to client) using this saved challenge
7. The CIFS client computes the challenge response as detailed in the CIFS specification, and then challenge response is sent as part of a SessionSetupAndX SMB
8. The CIFS server extracts the challenge response from above SMB
9. The CIFS server sends it's own SessionSetupAndX SMB to the domain controller using the extracted challenge response
10. The Domain Controller sends a SessionSetupAndX response to the CIFS server. This response will be successful if the CIFS client had provided the correct response.
11. The CIFS server tears down the session with the Domain Controller that was established by means of a LogOffAndX SMB.
12. The CIFS sever sends a SessionSetupAndX response to the CIFS client. This response is based upon the response from the Domain Controller.

Keep in mind that this is not a description of an attack it is the recommended behavior! In an environment where the only means of identification is the TCP/IP address and a computer name that can be changed at will.

7.4.2 Weaknesses in SMB Signing

The last weakness that we will describe is not really an attack. It is a general weakness that could be used in a number of ways. It is described by Hobbit in [30] and by Mudge (a member of the L0pht) in a posting to the bugtraq mailing list. It points out the weaknesses in the challenge/response encryption when the LAN Manager password is sent as well as the NT hash, which is the default behavior in NT. In appendix A.5, we describe how the response is calculated from the password hash and the challenge. The problem here is the Lan Manager hash. When the password is less then 8 characters then the last 8 bytes of the hash will always be 0xAAD3B435B51404EE. This means that 2/3 of the key used in the challenge response is already known together with the challenge and the response if they are snatched from the network. Keep in mind that DES ECB mode is used with the 21-byte padded hash split in 3 pieces. Under the circumstances mentioned, the third key is already known, it is 0x05EE0000000000. The second key needs to be found by brute force but since we know a large part of it, the structure of it is 0x??AAD3B435B514, it will only take 256 iterations in the worst case. To gain the first key a brute force approach on the whole keyspace is needed, this could however bee speeded up by using table lookup. Under the circumstances described above we now have the Lan Manager hash of the password. This password could be used to authenticate to a server as the cracked user by normal challenge response, or to fake signed SMB messages from that user, or feed it into a password cracker, e.g. L0phtCrack, to find the plain-text password. In the signing case the

attacker has to guess sequence numbers as well, but since a CIFS session always starts at 0 as the first sequence number, according to [45], it should not be to hard to guess.

7.5 Availability Attacks

7.5.1 NTCrash

Description. NT programs use the NTOSKRNL by invoking functions through calls to certain libraries (DLLs). In some of these calls the parameters are not checked properly. The missing checks are primarily range checks and legality of addresses. NTCrash is a program written by Mark Russinovich and Bryce Cogswell that exploits certain implementation flaws in NTOSKRNL. It is loaded from NTOSKRNL.EXE and contains the majority of the OS components that are executed in kernel mode. By invoking these functions with illegal or *out of range* or *out of bounds* parameters, NT will crash.

Intent. We will try to bring our target system to a grinding halt by invoking this program. What we want to achieve is to force a restart of the system and thereby deny service to others. In the target system, this would only stop our selves from using the system, but if it was executed on a server or a domain server this program could cause denial of service for a number of users as well as potential loss of data.

Result. We executed the program as follows:

```
ntcrash -n
```

After a few seconds the computer crashed, and therefore in accordance with our intentions the attack was a success.

7.5.2 Rollback

Description. Rollback is a utility tool that unintentionally was installed on the CD of NT 4.0. It is originally an OEM Pre-installation Kit tool. The effect of using it is that the whole system except the data files are destroyed. And there is no recovery except reinstalling the system and any program from scratch if you do not have a backup of the system. To make things even nastier there is no way of stopping the program once it is started and it does not give you a second chance when it is invoked.

Intent. We will try to destroy our system by invoking rollback.

Result. This attempt was unsuccessful, because we were unable to run this program as a normal user. However, we tried it as administrator and we were prompted to restart the computer. After restart, we had to reinstall NT in order to get the system operational.

7.5.3 CpuHog

Description. Normally, a thread in NT has a priority value between 1 and 15, where 1 is the least priority. It is not normal for a program to have a high priority value (>10). Furthermore, NT has an aging mechanism but will only age threads up to priority 14.

CpuHog is a small program written by Mark Russinovich which uses the priority mechanism of NT to hang the system. What CpuHog does is it sets priority 15 on itself and then enters an infinite WHILE loop. This will cause NT to hang so that it is impossible to start any other program including the Task Manager. The strange thing here is that you need no special privileges to be able to do this. Microsoft has in NT 4.0 Service Pack 2 and later addressed this problem by allowing aging up to priority level 15 which means that CpuHog will only slow down the system considerably. However, a user program can still set priority without special privileges.

Intent. The intention with this attempt is the same as with NTCrash (see above), i.e. the availability of the system will probably drop to zero.

Result. We executed the program as follows:

```
cpuhog
```

After confirming the initial question, the computer was unable to service any user. Therefore, the attack was successful.

7.5.4 Teardrop

Description. Teardrop is an attack or program that uses missing checks in the fragmentation handling of the IP stack. The whole idea is to send two IP packets; one that is normal but has the MF flag set, and another that has a fragmentation offset that is inside the first packet, but a total size that makes this fragment smaller than the first packet, i.e. the second packet is only a small piece of the data in the first packet. However, this time the MF flag is not set, so the system will treat the second packet as the last in the fragmentation run. When the system tries to align these packets it will end up with an offset that is larger than the end mark and therefore read to much data, and by doing this crash the system. Microsoft has offered a fix to this attack.

Intent. We will try to remotely crash the target machine by using the teardrop program written by "klepto" with the remote machines IP address. Since the program gives the user the ability to set both sending and receiving address we could remain totally anonymous. The attack will be tested both with and without the hot fix.

Result. We executed the program on a LINUX machine as follows (IP addresses changed for security reasons):

```
teardrop 10.0.0.1 10.0.0.2 -t 138 -n 10
```

Where the target system has IP address 10.0.0.2, -t is the port number and -n specifies the number of consecutive attacks. Without the hotfix the target machine froze and had to be rebooted. The hotfix offered by Microsoft seems to work.

7.5.5 Teardrop2 (bonk and boink)

Description. This attack has been called a number of thinks, Teardrop2 due to the fact that it is a variation of the teardrop code, bonk or boink which is the name of the programs, written by Jiva DeVoe, that utilizes this bug. The attack is similar to the Teardrop attack. Two IP packets are sent; one normal but with the MF bit set and, one

where the fragment offset is larger then the total size of the IP packet and no MF bit set. The fragment offset and the total size difference has, however, nothing to do with the problem, since the two are not related. The whole scheme works because the last fragment has an offset that is part of the UDP header and will therefore partially overwrite the header and the result is a incomplete UDP packet. These packets will take up memory and eventually cause a crash. The difference between bonk and boink is that bonk attacks only one port, namely port 55 while boink gives the user the option to define a range of ports to attack. Microsoft has offered a fix to the problem.

Intent. The same as in Teardrop but we will use both bonk and boink.

Result. We executed the programs on a LINUX machine as follows (IP addresses changed for security reasons):

```
bonk 10.0.0.1 10.0.0.2 100
```

The last argument specify the number of consecutive attacks. Without the hotfix the system crashed giving us a blue screen and had to be rebooted. The hotfix seems to work.

```
boink 10.0.0.1 10.0.0.2 100 200 10
```

The number 100 and 200 defines the port interval and 10 the number of consecutive attacks. Without the hotfix the system crashed giving us a blue screen and had to be rebooted. The hotfix seems to work.

7.5.6 Land and LaTierra

Description. Land and LaTierra is two programs that utilizes the same vulnerability. The only difference is that LaTierra has a number of options that can be set, e.g. which TCP flags to set, whether TCP or UDP should be used. The attack is simply to put the same IP address as both sender and receiver. This will cause a Windows 95 machine to crash and an NT machine to freeze for a while. This could be nasty, if one is the victim of a number of these attacks over a time period, especially if the intervals between the attacks are random. They will eventually make the user think that something is wrong and s/he will try to reboot to solve the problem. The attack could also be used as an excellent blocker in a man-in-the-middle attack. Microsoft offers a fix for the problem.

Intent. We will try to remotely freeze the target system by using both Land and LaTierra. The attack will be launched against a target, both with and without the hotfix.

Result. We executed the programs on a LINUX machine as follows (IP addresses changed for security reasons):

```
land 10.0.0.2 139
```

10.0.0.2 is the target computer's IP address, and 139 is the attacked port. With just SP1 installed in the system, the machine froze, and we had to reboot it. With just SP3, the machine froze for approximately 45 seconds. The hotfix offered by Microsoft seems to work.

```
latierra -i 10.0.0.2 -b 139
```

Arguments are the same as in the previous experiment. With just SP1 installed in the system, the machine froze, and we had to reboot it. With just SP3, the machine froze for approximately 45 seconds. The hotfix offered by Microsoft seems to work.

7.6 Confidentiality Attacks

7.6.1 NTFSDOS

Description. As previously mentioned it is possible to boot an NT system with a different operating system from a floppy drive, and then mount an NTFS volume from this operating system. The implications of this is that any system with a floppy drive, or which is bootable with another OS, is insecure. One example of such a driver is NTFSDOS

Intent. With the NTFSDOS utility, see appendix D.3.1, we will try to read an NTFS volume. We intend to carry out the attack as follows.

1. We will create a MSDOS boot floppy disk and put a copy of NTFSDOS on it.
2. We will then boot the target system from this floppy.
3. When the OS (now MS-DOS) is up and running, we will start NTFSDOS and hopefully every file on any drive on the target system will be readable by us.

Result. We did as specified above. After that, all files in the NTFS partition were readable.

Comment. If it is possible to read the NT file system, then it is also possible to read the SAM database. Information in this database includes hashed and encrypted passwords, which can be gathered by an attacker once s/he knows the internal file structure. Both the encryption methods and the way encryption keys are generated are known.

7.6.2 NTRecover (read-only version)

Description. Apparently it is possible, under certain circumstances, to remotely mount an NTFS partition over a serial line. This implies that any person with physical access to an NT machine, a (portable) computer and a serial cable can read the NT machine's NTFS volume. One example of such a tool is NTRecover.

Intent. We will install and configure the system as specified in appendix D.3.4. In this experiment we will use the read-only version of NTRecover.

Result. We installed and configured the utility and the equipment, which was an easy task and took less then 20 minutes. We were, after this phase, able to read the whole NTFS volume on the client machine.

7.6.3 RedButton

Description. The RedButton bug makes it possible to connect anonymously to an NT machine and enumerate users and shares as well as reading and modifying some parts of the registry. One of the reasons for anonymous logon, as we understand it, is the following. In order to manage trust in a convenient way the administrator in the trusting

domain needs some way of knowing which possible users or groups that is to be given rights in the trusting domain. But since the trusted domain does not trust the trusting domain, remember trust is one-way, s/he needs a way of getting access to this information. In this case the trusting domain will open anonymous channel to the trusted domain to obtain this information. The problem is that the anonymous user will have the rights of the everyone group and that group has far to much privileges. It is also not possible to just block the everyone group without creating a new group with almost the same privileges and then replace every occurrences of everyone to that group. If the group is just blocked the system will not function properly.

Microsoft has released a hotfix and instructions on how to handle this situation but it is still dependent on a lot of configuration. See Microsofts knowledge base article Q143474 for more information.

Red Button is a program written by NTsecurity.com (Midwestern Commerce, inc.) that demonstrates this weakness. According to them the program does the following.

- logs on remotely to a Target computer without presenting any Username and Password.
- shows that unauthorized access to sensitive information stored in file system and registry available to EVERYONE group can be obtained.
- determines the current name of Built –in Administrator account (thus demonstrating that it is useless to rename it).
- reads several registry entries (i.e it displays the name of Registered Owner).
- lists all shares (including the hidden ones).
- shows that identifier Everyone includes not only legitimate users of the network but everyone.

Some of this information can also be obtained without using the program if one has access to an NT server, e.g. at home. The idea here is to try to establish a trust relation with the attacked server as the trusted domain an the attacking server as the trusting domain. This trust relation will probably fail since the trusting server does not posses the right password, but the server will register the attacked servers as one of its trusted domains and will by the anonymous channel get information such as account and user names from the attacked server.

Intent. We will use the RedButton program on our target machine and see if it indeed gives us the information claimed above.

Result. We executed the RedButton program. It worked nearly as stated above. However, it did not give us some of the promised Registry information.

7.7 Integrity Attacks

7.7.1 ERD Commander

Description. The possibility of booting a machine from a floppy, and then mount the NTFS filesystem not only affects the confidentiality of the system. If it is possible to write to the mounted filesystem, as well as read from it, then the integrity of the system

is threatened as well. One example of this is ERD Commander that actually boots a stripped down NT system.

Intent. The goal with this attempt is to overwrite the SAM database with a new one, which has been created by us beforehand, and therefore includes known user name and passwords. When NT is up and running, it is impossible to replace the SAM database.

Result. We downloaded the program, and did installation as specified on the Web site. The program seems to work as promised. However, we were only able to test the free-ware version. This version has certain limitations, see appendix D.3.3. Therefore, we where not able to replace the SAM database.

7.7.2 NTRecover

Description. Even filesystem remotely mounted as mentioned in section 7.6.2 are writable with the same consequences as stated above.

Intent. The same as for ERD Commander.

Result. We installed and configured the utility and the equipment. We were, after this phase, able to both read from and write to the whole NTFS volume on the client machine. We checked this by replacing the SAM database.

7.7.3 NTLocksmith

Description. NTLocksmith is a program that must be used together with NTRecover with write permissions to specify a new password, i.e. overwrites the old one, for the administrator on the system. We have tried this program, and the authors claim that it works 100% of the time. This program will destroy both the integrity and the secrecy of any NT system with NTRecover installed. We believe that all this program does is to scan the SAM database for a certain string, and then overwrite the content a certain off-set from this point without knowing the real structure of the database. It is therefore highly unlikely that it will work on systems with SYSKEY, i.e. an extra level of encryption on the password, set in the registry.

Intent. We will try and overwrite the Administrators password with one that we have selected.

Result. After installation, we executed the utility. By doing this we were able to change the password for the Administrator on the target computer.

7.7.4 GetAdmin

Description. GetAdmin is a program written by Konstantin Sobolev that will add any user to the local administrator group. It is therefore possible for any user to become administrator on local machine. This in turn opens up for a number of attacks, e.g. installing trojans such as keyboard sniffers or network sniffers as automatic services. In a posting to NTsecurity.Net Thomas Lopatic gives the following analysis of the bug and the program.

I thought I'd share my findings with the net community. Hopefully someone will be able to fill in the gaps so that together we'll get the whole picture of what GetAdmin really does.

- *The hole: the system service NtAddAtom does not pay attention where it stores the result it returns. GetAdmin abuses this to set bit 0 of the byte at address NtGlobalFlag + 2. GetAdmin is traversing the list of exports in NTOSKRNL.EXE until it hits the 'NtGlobalFlag' entry. However, a slightly modified version of the exploit could contain the hard coded address of NtGlobalFlag. This would make any access to NTOSKRNL.EXE unnecessary, but it would make the exploit 'less portable' at the same time.*
- *By setting this bit, it seems, the check for debug privileges is switched off in NtOpenProcess and NtOpenThread. Thus, everyone is able to open any process currently in the system. When you open a process or a thread in Windows NT, your function call is eventually passed to the kernel services NtOpenProcess or NtOpenThread. These services verify, whether you are entitled to open the corresponding process or thread object. It seems, that the debug privilege enables you to open processes or threads not owned by you. When having a closer look at the kernel services, you notice that the above mentioned flag overrides the debug privilege, i.e. you do not need debug privileges if the above flag is set. Seems to be some kind of global debug privilege. But then again, maybe it's more than that and I just haven't looked closely enough. :) Could anyone from MS elaborate on this, please?*
- *GetAdmin opens the Winlogon process and adds a suspended thread to it. After that, it does some fiddling with the stack of the newly created thread. Guess it inserts some code and a bogus return address to make calls to GASYS.DLL, which eventually grants the administrator privileges after the thread is resumed. [Opening processes] is easy to verify with PView. Start PView and double click on the Winlogon process - you won't be shown its memory statistics. Now manually set this flag, e.g. with SoftIce, and re-run PView. Now you'll be able to look at the statistics. That means that PView is able to open the process, doesn't it? Maybe I am missing something here.*
- *The NT* functions are kernel services. User-land processes are not able to write to kernel memory. So, the problem is that NtAddAtom does not check the arguments it is passed thoroughly enough. Some background: The NT* functions are invoked via INT 2E. EAX contains the service number (for NtAddAtom this is 3) and EDX contains a pointer to the stack frame to be used when calling NtAddAtom, i.e. the arguments, to be passed to it. NtAddAtom takes two arguments: the string representing the atom and a pointer to the memory location the result, i.e. the atom handle, is to be stored in. NtAddAtom fails to assure that this location lies within the address space of the process. Hence it is possible to have the kernel service write the atom handle anywhere we want, e.g. somewhere near NtGlobalFlag. If other services are equally sloppy, then I am afraid that this has just been the beginning in a series of bugs (the buffer overflows on Unix come to mind).*
- *The...correct fix is to modify the kernel so that it cannot be tricked into abusing its privileges (SUID PROGRAMS ON UNIX COME TO MIND HERE).*

Microsoft offers a hotfix for the problem.

Intent. We intend to try and get local administrator rights on the target machine by invoking GetAdmin. The attack will be carried out both with and without the hotfix.

Result. We logged in on the target computer as the normal user “nisse” and executed the program as follows:

```
getadmin nisse
```

After rebooting the user “nisse” was a member of the administrator group and had local administrator privileges. The hotfix offered by Microsoft seems to work.

7.7.5 Password Cracking

In a system with weak password encryption it is feasible to crack passwords with a brute force attack. The problem with NT is that the passwords are stored in Lan Manager format as well as in NT native format. The Lan Manager format has a much weaker encryption algorithm than the native format e.g. in Lan Manager only uppercase characters are allowed. The other problem is how to get hold of the encrypted password for later cracking. In NT, one can either try to get hold of a copy of the SAM database or sniff the network for them. Programs exists for both purposes e.g. SAMDump and readsmb (a program that comes with L0phtCrack 2.0). Below we will present three different types of password cracking attacks.

7.7.5.1 ScanNT

Description. ScanNT is a simple password cracker written by Andy Baron. It uses a dictionary and multiple calls to the *Login()* system call to try and figure out the password of a user. The technique used is a brute force trial and error attack. The program can only be used if you already are logged in to the system as a user.

Intent. We intend to try and find the password of the administrator account on the target system by using ScanNT. The administrator is the “superuser” of the NT system and the account can never be deleted. However, it can be renamed so one advantage that we have in this attack is that we know that the username of the account is indeed Administrator. This means that we need not try different usernames. If the attack succeeds we will have unlimited access to all the resources of the system.

Result. This attempt was not successful. After executing the command as follows:

```
scannt -p scannt.dic
Administrator, You must have SeTcbPrivilege privilege (see readme.txt)
```

This implies that we do not have sufficient privilege to execute the *Login()* system call.

7.7.5.2 L0phtcrack 1.x

Description. See appendix D.1.1 for a description of L0phtCrack.

Intent. The same as for ScanNT.

Result. This attempt was successful. To produce a password file, we used the SAM-Dump Utility on the expanded copy of the SAM database found under the repair folder.

Comment. The Lan Manager password is padded with NULLs if it is not 14 characters long. This together with the encryption method used, creates recognizable patterns if the password contains less than eight characters.

7.7.5.3 L0phtCrack 2.x

Description. See appendix D.1.1 for a description of L0phtCrack.

Intent. We will use the program *readsmb* that comes with L0phtCrack 2.0 to snatch encrypted passwords from the network. After that we will use L0phtCrack to try and get the clear-text passwords.

Result. We ran readsmb for 24-hours and got 10 different encrypted passwords, among them the administrator password (readsmb where used on our “live” network and not on the experimental network so wee believe that the results from this is not biased). After that, L0phtCrack 2.0 was run on the file from the sniffer.

7.8 Classification of weaknesses

A classification of the weaknesses is presented in the table below. The classification is taken from [66] and adapts the viewpoint of intrusion techniques. It is based on a scheme originally suggested by [48].

TABLE 5. Classification of weaknesses in NT

Category			**Weakness**
NP5 Bypassing Intended Controls	Password attacks	Capture	Plain-text passwords 7.3.5
		Guessing	Collisions in MD4 7.3.2 Deriving passwords from challenge/ response 7.4.2 L0phtCrack 7.7.5.2 and 7.7.5.3
	Spoofing privileged programs		Parameter checks in system calls 7.3.3 Undocumented functions 7.3.4 NTCrash 7.5.1 Teardrop 7.5.4 GetAdmin 7.7.4
	Utilizing weak authentication		Client Downgrading 7.4.1 Brezinskis man-in-the-middle 7.4.1 NTFSDOS 7.6.1 NTRecover 7.6.2 and 7.7.2 ERD Commander 7.7.1 NTLockSmith 7.7.3 RedButton 7.6.3
NP6 Active misuse of resources	Exploiting inadvertent write permissions		
	Resource exhaustion		CPUHog 7.5.3 Teardrop2 7.5.5
NP7 Passive misuse of resources	Manual Browsing		
	Automated searching	Using a personal tool	
		Using a publicly available tool	

In our study, we have not presented any weaknesses in the last categories. That does not imply that they do not exist. IS, DumpACL, KSA and C2CONFIG, see appendix D.1, are all examples of utilities that satisfies this categories.

In the next section, we will compare the weaknesses found in NT with known UNIX exploits.

8. NT versus UNIX with NFS and NIS

The UNIX operating system was designed in the early seventies at AT&T Bell research laboratories. The authors had previously worked on the Multics system and they applied many lessons learned in that project when designing UNIX. The name incidentally was meant as a pun on Multics. The design was simplified in most respects however. Traditionally, UNIX implementation has been centred around a single monolithic kernel with two access control rings, supervisory state and user state. The kernel contains all code necessary to operate the system, such as filesystem, device drivers, as well as ”traditional” code for process handling etc.

In the late seventies ARPA decided to fund research, into the field of long distance computer communication, and as a part of that research researchers at Berkeley developed the Berkeley Software Distribution, or BSD for short. Beside many other inventions, BSD contained the first implementation of what was to become TCP/IP. Since the code for the BSD TCP/IP implementation was in the government domain it was, and is widely available, many (most) other vendors have based their implementations of the TCP/IP stack on the code from Berkeley. Berkeley worked steadily on removing all original AT&T code from the Berkeley software release and has now achieved a code base totally free from any obligations originally made to AT&T.

With networking in place the researchers at Berkeley designed a set of protocols and applications to be able to access computers over the local area network, these are collectively known as the ‘r’ protocols since the first letter of the commands that invoke them typically begin with the letter ‘r.’ These protocols were designed to permit remote terminal access (via ‘rlogin’ and ‘telnet,’) remote file transfer, as opposed to file access, (via ‘rcp’ and ‘ftp,’) and remote command execution (via ‘rsh.’) These protocols are all very similar from a security perspective.

The Berkeley line of UNIX started one of the earliest divisions of UNIX into what is today a multitude of different systems from different vendors. The lineage of any UNIX system can be traced back to either AT&T UNIX, or Berkeley UNIX. Today it is estimated that some 10-15 different UNIX implementations is in use world wide. Even though they all share the same ancestry it is difficult, especially from certain security perspectives to discuss “UNIX” as a homogenous system. Many of the implementations available today differ in vital respects from each other.

With the market shifting towards single user workstations much research was being conducted in the field of distributed computing. In the early eighties the vendor Sun Microsystems published a set of standards on how to make files on a file server available remotely via TCP/IP, specifically via remote procedure call, RPC (another Sun development). Sun named this NFS, the Networking File System, and this was soon made the de facto standard for remote file access, with many vendors implementing compatible products.

In order to make it possible to treat a large number of workstations as a single homogenous installation, remote file access was not enough however. There are a number of other relevant datastructures, traditionally stored in configuration files, that has to be distributed to the workstations. Moreover, distributing these data in its entirety every time a single request was made would be prohibitive in terms of network traffic, since

the advent of networked computing had made many of these data structures rather large.

To combat this problem, again Sun Microsystems, developed a client server structure for answering queries from individual workstations. Sun originally named this service the Sun Yellow Pages, but when it became apparent that the name was already registered, Sun changed the name to NIS or Network Information System. NIS consists of a number of primary and secondary servers that answer individual questions from workstations about user accounts, hostname to IP address resolution, Ethernet network addresses etc. For an in depth treatment of UNIX internals see for instance [58], or [59].

8.1 Security

Neither of the systems above were designed with security as one of the design goals. They all stem from the time when “first make it work” was the order of the day. However, even though UNIX eliminated many of the advanced security features available in Multics, must still be considered a workable platform on which to build secure systems. Some basic security mechanisms are present; different processes have different memory spaces, and they cannot read or write memory belonging to another process. Calls into the kernel are checked against the user owning the process, only the super user (an all powerful administrative account) can perform certain security critical operations. Processes can inherit super user privileges to perform some security critical function (such as logging in another user) in a limited and secure manner. The filesystem keeps record of ownership of all files and objects and permissible operations (read/write/execute) can be determined on an individual basis for each file.

The security mechanisms above, while they are the very building blocks that can make a UNIX system secure, also leave a lot to be desired. The introduction of an all powerful administrative account introduces a single point of failure into the system. When an intruder has gained super user privileges little can be done to trace his actions or to eject him from the system.

The process privilege inheritance mechanism, the set UID, or SUID mechanism, is in many respects a too coarse grained mechanism, it’s all or nothing. Many security flaws are introduced into UNIX system because a SUID program is tricked into doing something other than it was originally intended to do. Since all it’s actions are performed with maximum privilege, it can affect the system in any way the intruder sees fit, especially in areas that have no connection with the original privileged operation the program was designed to mediate access to.

The file permission system was originally somewhat coarse grained and did not facilitate the enforcement of many reasonable security policies. Permissions could be set to allow read, write, and execute access according to the criteria ”owner”, ”group”, and ”others.” However, today all major manufacturers of UNIX support some kind of ACL (Access Control List) feature similar to NT. The various systems do not follow any standard of how to accomplish this, and perhaps as a result, the majority of UNIX installations do not activate the ACL feature provided them.

The Berkeley ‘r’ protocols mentioned earlier were designed for an environment where all machines and the network connecting them were located in a single computer room, and all access to the network was via UNIX machines. Thus the authentication mechanism for these protocols is based on the transmission of clear text passwords via the network. The concept of an all powerful administrator is carried over in the Berkeley implementation of the TCP/IP protocol suite, and the ‘r’ protocols. Any request coming from a TCP/IP port with a port number below 1024 is considered to have come from a process with administrative privileges on that host. Since none of the above is not typically true of today’s installations these security “features” leave a lot to be desired.

NIS and NFS contain few security mechanisms, if any. What little security NIS and NFS can be configured to have is mostly derived from the reserved port numbers “feature” mentioned above, and a simple IP-address host based address check. The overwhelming majority of information transmitted via the network from these services are furthermore in the clear, and thus anyone with access to the network is free to listen in, and/or modify the traffic.

The situation is made worse by the fact that as noted above the trust relationship between server and client is a trifle naive. If you have a host under your control anywhere on the network, both NFS and NIS will most likely trust it implicitly, and you can for instance impersonate users, to gain access to their files etc. In the case of NIS it is easy to gain access to the data it distributes, even though you are not among the hosts originally intended as a recipient of that information. Some, if not to say most, of this information; user accounts etc are of a security critical nature. The interested reader is referred to the standard reference of the subject of UNIX security [59].

8.2 UNIX versus NT

Even though Microsoft Inc. would like to have us believe otherwise, NT does not in fact contain many ideas that UNIX has not either pioneered or picked up during the seventies or early eighties. Both systems have from a research perspective a distinctive seventies feel to them. Especially from a security perspective one is struck by the similarities between the respective systems. There are in fact many more similarities than differences. NT has adopted the concept of an all powerful administrative account, even though the corresponding SUID feature is not as heavily depended upon for administratively critical tasks. NT has the same basic structure when it comes to processes and privileges. The file system has the same basic layout even though a few more operations besides (read/write/execute) can be specified.

About the only relatively recent technological advance that Microsoft has incorporated is microkernel technology. However, Microsoft’s implementation is several megabytes per kernel server, much to large to gain any real security advantage. It is interesting to note that the NT kernel is of approximately the same size as a typical UNIX kernel, in terms of services provided. A UNIX kernel on the same hardware is typically smaller in terms of memory consumption. Furthermore, in later releases of the NT operating system, much code from the servers, most notably the GUI, have been moved into the kernel proper, with negative effects on security. Some performance has been gained by this however.

When it comes to networking, the NT ”domain” concept is remarkably similar to NIS in its structure.

Hence, one would not be surprised if one saw the same security problems, that has plagued UNIX over the years, recur in NT.

8.3 Intrusion comparison

We have chosen to base the comparison between the various security flaws present in NT on experience gained in our own UNIX intrusion experience, and publicly available material from the Computer Emergency response team, CERT, of the Carnegie Mellon University. The following compares each flaw found in NT with its counterpart, if any, in UNIX systems. Since UNIX is a more mature system than NT, the overwhelming majority of flaws presented here have been addressed in current UNIX systems. The comparison with many of the UNIX flaws thus has a distinct historic flavour.

8.3.1 Installation Problems (see 7.3.1)

Many UNIX systems have suffered the same flaws, i.e. the installation process has left security critical features inactive, or misconfigured. A typical example of this is found in the early SunOS installations that contained the file /etc/hosts.equiv containing a single ”+”. This meant that anyone, anywhere on the network, could log into the machine, with very little effort [57].

8.3.2 MD4 Collisions (see 7.3.2)

No UNIX system uses the MD4 algorithm for hashing passwords. The traditional UNIX password hashing algorithm is a modified DES, though some use MD5. The modifications are supposed to prevent the intruder from gaining access to fast hardware to simplify his task. Originally the modified DES algorithm was sufficiently slow to implement on the current computers for the scheme to provide some security. This is less true today. The basic problem of relatively easy access to hashed passwords and the subsequent ability to mount a brute force attack against them have always been present in UNIX systems however. Both operating systems are relatively similar in this respect [3].

8.3.3 Failed parameter checks on critical system calls (see 7.3.3)

Historically there has been some instances of these in UNIX systems, even though most (all) have been eliminated by now. UNIX today is remarkably resistant to attacks that systematically provides random arguments to system calls. In library routines and SUID utilities the problem persists however. Most notably, this software does not consistently check that the arguments supplied by the attacker fit in buffers supplied by the program. The attacker then typically uses this oversight to supply the program with carefully crafted arguments that overrun the stack of the running program, and when it tires to return from the subroutine it is diverted to execute the carefully crafted argument as code, and is subverted into performing actions on the behalf of the attacker. Historically these problems where reported as crashes, since it is improbable that a ran-

dom argument will contain valid machine code. The prevalence of source code to the various UNIX implementations has made this kind of devastating attack wide spread however. NT will probably see a similar development, there are no technical differences to prevent the same kind of attack. The absence of source code will make it more difficult for the attackers to find such flaws, but not impossible [10] and [14].

8.3.4 Undocumented system variables and functions (see 7.3.4)

These are not really a security problem in UNIX, the authors know of no such attack against a UNIX system. This is probably due to the wide spread distribution of source code, and a long standing tradition of openness in UNIX system development. All functions of the kernel are typically documented. Again utilities have had their share of such problems however, classic examples are the ‘sendmail’ the electronic mail transfer agent, DEBUG and WIZ commands, that existed in early ‘sendmail’ versions, and that granted unlimited access to the system via the network. These are of a historic nature however [6].

8.3.5 SMB challenge response (see 7.4.1, 7.4.2)

There is no direct counterpart in UNIX. Old protocols typically do not use challenge/response, and the modern ones that do typically get it right.

8.3.6 Plain text passwords over the network (see 7.3.5)

The Berkeley ‘r’ protocols pioneered the idea of sending unencrypted passwords over the network. This actually made some sense at the time, since the local network consisted of a typically homogeneous installation of UNIX machines, none of which would allow access to the network with out super user privileges. This assumption has changed today however, but despite this the Berkeley ‘r’ protocols are still in wide spread use. Password sniffing is one of the most common attacks carried out against UNIX installations [10] and [11].

8.3.7 Non NTFS file systems (see 7.3.6)

Traditionally UNIX has implemented its own file system, for many years there was only one. Berkeley corrected a number of performance deficiencies with the Berkeley fast file system, and variations thereof are still common today. Even though most modern UNIX systems allow access to MS-DOS file systems, these accesses are typically limited in such a way as to present no serious security implications. It is impossible to actually run a UNIX system from anything other than a UNIX file system, and the kind of problems that NT has when a non NT file system is used for critical tasks does not appear in UNIX [58].

8.3.8 System initialization problems (see 7.3.7)

UNIX have the same boot problems that NT has. Once the intruder has gained physical access to the computer and its disks, little can be done to prevent him from forcing the system. Some UNIX implementations where also traditionally misconfigured, or could be interrupted in the middle of the boot process to provide the intruder with an admin-

istrative command interpreter, so called single user boot. This was intended for critical system administration, such as repairing disk partitions that prevented further start up of the system, but also opened an easy avenue of attack against the system [62].

8.3.9 Teardrop/Teardrop2/Land/laTierra (see 7.5.4, 7.5.5, 7.5.6)

This is, at the time of writing, a rather recent attack that caught many UNIX and router vendors out in the open. A variety of implementations were vulnerable to either one or both of these types of attacks [12].

8.3.10 NTCrash (see 7.5.1)

As stated earlier this type of attack has little success when mounted against system calls in the modern UNIX kernel. There are still serious flaws concerning buffer overrun in many applications and system libraries. It is interesting to note that this attack need not originate from an account on the system, in many cases daemons or utilities reading input from the network are vulnerable, and the attack can be successfully mounted from outside the system [10], [14], [12], [15] and [17].

8.3.11 CpuHog (see 7.5.3)

UNIX has had a similar problem, and most modern UNIX systems implement some kind of per user limits as to how many resources in terms of memory, CPU-time, number of processes etc. Before these mechanisms were in place however, many simple availability attacks where possible, such as repeatedly creating processes until the system had reached its limit [58].

8.3.12 RedButton (see 7.6.3)

Stock UNIX with NFS and NIS does not have exactly the kind of interdomain trust relationship that is described in this kind of attack. Similar problems with NIS being to easy to extract the same kind of sensitive information have been known however [8]. Besides, guessing the information required to obtain this information by the specified NIS behaviour is seldom very difficult [69]. Other UNIX utilities such as *finger* can also be used to gain valuable information of the same nature [59].

8.3.13 NTFSDOS (see 7.6.1)

Since the information on disks are not encrypted or otherwise protected against reading or writing the same kind of attack, e.g. to physically remove a disk from a machine and attach it to another that the attacker has full control over, is quite possible. UNIX requires physical security of its servers to deal with this situation [59] and [62].

8.3.14 ERD Commander/NTRecover/NTLocksmith (see 7.7.1, 7.6.2, 7.7.2, 7.7.3)

There are no similar products, per se, available for UNIX. There is nothing preventing the attacker from physically removing the disks from a non-running system and attach them to another computer, or boot the computer from removable media and thus gain access to the disk without control from the normal operating system. Commonly avail-

able maintenance software will then let the attacker to read, and modify, files on the mounted disk at will. It would be simple to modify local password files to incorporate a new supervisor account with a new password or otherwise modify user authorization data as is done with NTLocksmith [58] and [59].

8.3.15 GetAdmin (see 7.7.4)

UNIX does not handle access to processes memory (for debugging purposes etc) in the same way as NT. While there is no directly corresponding attack, there has been a flaw associated with the corresponding mechanism used for this purpose in certain implementations of UNIX. Access to the kernels (and hence vital process information) is mediated by the file /dev/kmem. In one release of SunOS this file was configured with incorrect access permission, when the system was installed as per the manufacturers instructions. Thus any user could access the kernels memory, instead of this being an exclusive privilege of the administrator. This made it a trivial matter to change UID of running processes etc [8].

8.3.16 NTRecover with write permission (see 7.7.2)

See section 8.3.14.

8.3.17 Password cracking via the network (L0phtcrack) (see 7.7.5, 7.7.5.2, 7.7.5.3)

Since the introduction of NIS this has been a possibility in UNIX installations. NIS does very little in the way of preventing unauthorized hosts from gaining access to the hashed passwords of a system. When these passwords have been obtained the attacker is free to try and crack them off line. Since there was no need to be backward compatible with any existing system when NIS was designed however, the flaws that appear in conjunction with LAN Manager passwords has no direct counterpart in UNIX/NIS [59], [3] and [8].

8.4 Summary

When we summarize the comparison above we see that all weaknesses in NT have their counterpart in UNIX with the following exceptions:

- 8.3.3 Undocumented system variables and functions
- 8.3.4 SMB challenge response
- 8.3.6 Non NTFS file systems

The two last ones, (8.3.4) and (8.3.5), fall within the area of specific system design, for which there is no direct counterpart with any feature of the design of UNIX.

A classification corresponding to that of Table 5 in section 7.8 is found in Table 6. By comparing the two tables we can see that it has been possible to find a UNIX counterpart to all NT categories.

TABLE 6. Classification of UNIX weaknesses

Category			**Weakness**
NP5 Bypassing Intended Controls	Password attacks	Capture	Plain-text passwords over the network 8.3.6
		Guessing	Password cracking via the network (L0phtCrack) 8.3.17
	Spoofing privileged programs		Failed parameter checks on critical system calls 8.3.3 NTCrash 8.3.10 Teardrop 8.3.9 GetAdmin 8.3.15
	Utilizing weak authentication		NTFSDOS 8.3.13 ERD-Commander/NTRecover/ NTLocksmith 8.3.14 RedButton 8.3.12
NP6 Active misuse of resources	Exploiting inadvertent write permissions		Installation Problems 8.3.1
	Resource exhaustion		CPUHog 8.3.11 Teardrop2 8.3.9
NP7 Passive misuse of resources	Manual Browsing		
	Automated searching	Using a personal tool	
		Using a publicly available tool	

The similarities between UNIX and NT demonstrated above support the hypothesis that the security behavior of the two operating systems is very similar.

9. Discussion

In this section we will summarize our impressions of the NT operating system from our point of view.

9.1 Windows NT

Even though the development team of NT had a strong ambition to make it a secure operating system, this was apparently not enough. We have found weaknesses covering the whole "CIA" range. Maybe, it was put aside or diminished by the demand for backward compatibility. NT also has a lot of functionality and is therefore a complex system. It is easy to overlook shortcomings and introduce errors in such a system, but one could hardly justify missing range checks or tests for invalid parameters by this argument. These types of errors points towards deficiencies in the review and design processes. This suspicion is further strengthened by the fact that the Service Packs seams to change, more or less, half the operating system.

In some cases, Microsoft seems to really on security by obscurity, e.g. the structure of the SAM database and how the so called secure channels are established between the client and the server. This approach might work at first, but it is probably devastating in the long run. Security must be built on concepts and methods that can be described and explained fully and still be effective. Right now there are mechanisms in NT that are very hard to find documentation on.

Another matter that is interesting to note is that many weaknesses display similarities with old weaknesses in UNIX. This could be due to the fact that the two operating systems, in our opinion, has the same basic structure (NT's microkernel has the same functionality as UNIX "monolithic" kernel). The fact that the NT development team seemingly have not used the experiences from the UNIX community or the recent advances in the security area somewhat takes the edge of there intended ambitions.

9.2 Trends

In the future, we believe that the number of successful attacks against NT systems will increase dramatically. We base this assumption on two basic facts. First, since the source code of the NT system is not easily available for the system owners they have to wait on Microsoft to correct any system weaknesses. These weaknesses will be spread all across the hacker community and cause a lot of damage, usually much faster then it takes to get hold of the fix for the problem. The attacker will have an advantage in that s/he usually does not have anything against reengineering code, something system owners either are not allowed to, or have no interest in doing. The only thing the system owner can do, if s/he does not have a very good technical knowledge of the system, is to close the system in the meantime or take the risk. The only light here is that as the system gets more familiar to the community and the knowledge of its workings gets shared there should be others than Microsoft who could do fixes. Second, the NT operating system is much more homogenous than the UNIX operating system, due to the fact that there is only one developer of NT, but a lot of developers of UNIX systems. The effect of this will be that all NT installations will be affected by a newly found weakness, since the cause of the exploit is likely to be present in all of them. In a

UNIX environment, on the other hand, usually only one variant of the operating system is affected so a potential attacker has to know which target has which variant.

9.3 Recommendations

Last we would like to give some overall recommendations based on the knowledge we gained by doing this study.

- Do not connect any NT system to an outside network without putting a well working firewall between the outside network and the NT system. There are weaknesses in the protocols used to connect different machines in the system and one should not let outsiders take advantage of that. Especially block ports 135-139. No inbound or outbound traffic should be allowed on those ports.
- Get some sort of intrusion detection system, see for example KSA in appendix D.1.5. The earlier an attacker can be detected the faster one can block the hole in the defence, or at least detach the system from the outside network until a fix for the exploit is found.
- Get as much knowledge on the system, and how it works, as possible. This is especially important in the NT case since the system at a first glance seems to be so simple to administrate. But one really need to know what goes on behind the *combo boxes* and the windows to be able to judge which actions to take and to make sure that the system really does what one really wants when the button is pushed. It is especially important to learn a lot about the Registry. If one not learns, one put a lot of trust in the hands of Microsoft.
- Educate the users. Most of them will probably come from a Windows 95 or Windows for workgroups environment and will be used to almost no restrictions regarding installation of programs or access to resources. It is essential to explain to them why these restrictions are important, otherwise they might try to circumvent them.
- The standard installation is very relaxed, and therefore insecure. Go carefully through the system and make sure that the configuration is the one that you want. There are utilities that will help in this situation. Use them. There are also a lot of recommendations in books and on the Internet. Take an extra look at the security pages. See the appendix for descriptions of utilities and useful Internet addresses.
- Keep as few user accounts as possible on local machine. On a local machine the SAM database is usually much more exposed than on a Domain Controller and never use the same password on the local administrator account as on the domain administrator account. Delete the copy of the SAM in the repair folder.
- Block the functionality the users do not need. There are few users who need to install program or export shares. Unfortunately some of the NT applications needs to be able to write to directories where the user normally should not have permissions to write. This will, however, change in newer releases.
- Consider using encryption on network connection and file systems and stronger authentication mechanisms if you are very concerned about security. Programs in this area are available for NT, e.g. SSH. However, make sure that you know how they affect the performance and the function of the system.

10. Conclusion

Our analysis and tests show that there are ample opportunities to improve the security of Windows NT. The relatively few intrusions so far are due to the fact that the system has been relatively unknown, and that the principle of “security by obscurity” has been used. However, this will not maintain security in the long run.

In our opinion there are no fundamental differences between the security of UNIX and NT. It is true that there are presently more intrusions in UNIX systems, but we believe that this is due to the “aging” factor, i.e. the statement above holds when comparing the systems at the same state of development and market penetration. Thus, the only reason for more UNIX penetration should be that the system is older and more well-known. Should these assumptions be correct, we have reason to believe that there will be an increasing rate of intrusions into the NT operating systems.

11. References

[1] A. D. Birrell and B. J. Nelson, Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, vol. 2, No. 1, Feb 1984.

[2] Abraham Silberschatz and Peter B. Galvin, Operating System Concepts, 4th Edition. Addison-Wesley Publishing Company, 1994.

[3] Alec D.E. Muffett, Crack - A Sensible Password Checker for UNIX, 1992.

[4] Bertrand Meyer, Object-Oriented Software Construction: 2nd edition. Prentice Hall, 1997.

[5] Brad Curtis Johnson, A Distributed Computing Environment Framework: An OSF Perspective. June 10, 1991.

[6] Bryan Costales, Sendmail [2nd ed.], O'Reilly & Associates, Inc., 1997.

[7] C. R. Attanasio, P. W. Markstein and R. J. Phillips, Penetrating an Operating System: a study of VM/370 Integrity. IBM Systems Journal, pp. 102-116, Volume 16, Number 1, 1976.

[8] [CERT CA-92.13] SunOS NIS Vulnerability, June 4, 1992, http://www.cert.org.

[9] [CERT CA-93.03] SunOS File/Directory Permissions, February 3, 1993, http://www.cert.org.

[10] [CERT CA-94.01] Ongoing Network Monitoring Attacks, February 3, 1994, http://www.cert.org.

[11] [CERT CA-95.18] Widespread Attacks on Internet Sites, December 18, 1995, http://www.cert.org.

[12] [CERT CA-97.06] Vulnerability in rlogin/term, February 6, 1997, http://www.cert.org.

[13] [CERT CA-97.10] Vulnerability in Natural Language Service, April 24 1997, http://www.cert.org.

[14] [CERT CA-97.11] Vulnerability in libXt, May 1, 1997, http://www.cert.org.

[15] [CERT CA-97.21] SGI Buffer Overflow Vulnerabilities, July 16, 1997, http://www.cert.org.

[16] [CERT CA-97.28] IP Denial-of-Service Attacks, December 16, 1997, http://www.cert.org.

[17] [CERT CA-98.05] Multiple Vulnerabilities in BIND, April 08, 1998, http://www.cert.org.

[18] Charles B. Rutstein, Windows NT Security: a practical guide to securing Windows NT servers & workstations. McGraw-Hill, 1997.

[19] Charles P. Pfleeger, Security in Computing. Prentice-Hall International, 1989.

[20] Cleveland Systems Limited, The Effects of Disk Fragmentation on Windows NT File System (NTFS) Performance. December, Executive Software International Inc., 1995.

[21] Collisions in MD4. RSA Laboratories´ Crypto Bytes, Volume 1, Number 3,

Autumn 1995.

[22] D. Brent Chapman and Elizabeth D. Zwicky, Building Internet Firewalls. O'Reilly & Associates, Inc., 1995.

[23] Dennis Martin, Windows NT File System Security: An Overview and Comparison, Revision 1.1. November 1994.

[24] Dominique Brezinski, A Weakness in CIFS Authentication. February, 1997.

[25] Hans Dobbertin, Alf Swindles Ann, RSA Laboratories´ Crypto Bytes, Volume 1, Number 3, Autumn 1995.

[26] Helen Custer, Inside the Windows NT File System. Microsoft Press, 1994.

[27] Helen Custer, Inside Windows NT. Microsoft Press, 1993.

[28] Helen Custer, A Grand Tour of Windows NT: Portable 32-bit Multiprocessing Comes to Windows. Microsoft Systems Journal, Volume 7, Number 4, pp: 17-31, Jul-Aug, 1992.

[29] Hobbit, CIFS: Common Insecurities Fail Scrutiny. Avian Research, January, 1997.

[30] Hobbit, CIFS is a load of CACA (Exploring the security of SMB resource sharing). Presented at Defcon 5. July, 1997.

[31] I. Heizer, P. Leach and D. Perry, Common Internet File System Protocol (CIFS 1.0). Internet Draft, 1996.

[32] Ira S. Winkler and Brian Dealy, Information Security Technology?… Don't Rely on It: A Case Study in Social Engineering. 5th UNIX Security Symposium, June 5-7, 1995.

[33] Jeffrey M. Richter, Creating, Managing, and Destroying Processes and Threads under Windows NT. Microsoft Systems Journal, Volume 8, Number 7, pp: 55-76, July, 1993.

[34] John Enck, Windows NT Server 4.0 Advanced Technical Reference. Que, 1997.

[35] Larry Lange, Hackers keep the heat on Windows NT Security. DVD Technology Seminar, June 10, 1997.

[36] Lee Hadfield, Dave Hatter and Dave Bixler, Windows NT Server 4 Security Handbook. Que, 1997.

[37] Luke Kenneth Casson Leighton, Paul Ashton and Duncan Stansfield, NT Domain Authentication. November, 1997.

[38] Marshall Brain, Inside Windows NT Services. Dr. Dobb's Journal, pp: 48-51, May, 1994.

[39] Microsoft Corporation, Microsoft Windows NT Server Resource Guide: for Windows NT Server Version 4.0. Microsoft Press, 1996.

[40] Microsoft, Microsoft Windows NT from a Unix Point of View: A White Paper from the Business Systems Technology Series. 1995.

[41] Microsoft Corporation, Microsoft Windows NT Server Internet Guide: for Windows NT Server Version 4.0. Microsoft Press, 1996.

[42] Microsoft Corporation, Microsoft Windows NT Server Networking Guide: for

Windows NT Server Version 4.0. Microsoft Press, 1996.

[43] NCSC, FINAL EVALUATION REPORT Microsoft Inc.: Windows NT Workstation and Server Version 3.5 with U.S. Service Pack 3. National Computer Security Center, 1996.

[44] Nevin Lambert and Manish Patel, PCWEEK Microsoft Windows NT Security: System Administrator´s Guide. ZDPress, 1997.

[45] Paul J. Leach, CIFS Authentication Protocols Specification. Microsoft, Preliminary Draft, Author's draft: 4.

[46] Paul J. Leach and Dilip C. Naik, CIFS Logon and Pass Through Authentication. Internet Draft, 1997.

[47] Paul Yao, Windows 32-Bit API Gives Developers Advanced Operating System Capabilities. Microsoft Systems Journal, Volume 6, Number 6, pp: 15-23, Nov-Dec, 1991.

[48] Peter G Neumann and Donn B Parker. "A summary of computer misuse techniques." In *Proceedings of the 12th National Computer Security Conference*, pages 396–407, Baltimore, Maryland, USA, October 10–13, 1989.

[49] RFC 1001, Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods. March, 1987.

[50] RFC 1002, Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications. March, 1987.

[51] RFC 1055, A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP. June, 1988.

[52] RFC 1144, Compressing TCP/IP Headers. February, 1990.

[53] RFC 1332, The PPP Internet Protocol Control Protocol (IPCP). May, 1992.

[54] RFC 1334, PPP Authentication Protocols. October, 1992.

[55] RFC 1552, The PPP Internetwork Packet Exchange Control Protocol (IPXCP). December, 1993.

[56] RFC 1661, The Point-to-Point Protocol (PPP). July, 1994.

[57] RFC 1717, The PPP Multilink Protocol (MP). November, 1994.

[58] Samuel J. Leffler [et al], The design and implementation of the 4.3BSD UNIX Operating System, Addison-Wesley Publishing Company, 1989.

[59] Simson Garfinkel [et al], Practical UNIX and Internet Security [2nd ed.], O'Reilly & Associates, Inc., 1996.

[60] Stephen A. Sutton, Windows NT Security Guide. Addison-Wesley, 1997.

[61] Thuan Q. Pham and Pankaj K. Garg, Multithreaded Programming with Windows NT. Prentice Hall, 1996.

[62] Tomas Olovsson, Erland Jonsson, Sarah Brocklehurst and Bev Littlewood, Data Collection for Security Fault Forcasting: Pilot Experiment. Technical Report No 167, Department of Computer Engineering, Chalmers University of Technology, 1993 and ESPRIT/BRA Project No 6362 (PDCS2) First Year Report, Toulouse,

France, pp. 515-540, Sept. 1993.

[63] Tomas Olovsson, Erland Jonsson, Sarah Brocklehurst and Bev Littlewood, Towards Operational Measures of Computer Security: Experimentation and Modelling, in B. Randell et al. (editors): Predictably Dependable Computing Systems, Springer Verlag, 1995. pp. 555-572.

[64] Tom Sheldon, Windows NT Security Handbook. McGraw-Hill, 1997.

[65] Trusted Computer System Evaluation Criteria ("orange book"). National Computer Security Center, Department of Defense, No DOD 5200.28.STD, 1985.

[66] U. Lindqvist and E. Jonsson. "A classification of computer security intrusions: Means and ends." In *NORDSEC'96 – Nordic Workshop on Secure Computer Systems*, Göteborg, Sweden, November 7–8, 1996.

[67] Ulf Gustafsson, Erland Jonsson and Tomas Olovsson, Security Evaluation of a PC Network Based on Intrusion Experiments. Proceedings of the 14th International Congress on Computer and Communications Security, SECURICOM '96, Paris, France, pp. 187-203, June 4-6, 1996.

[68] Ulf Lindqvist, Thomas Olovsson and Erland Jonsson, An analysis of a secure system based on trusted components." In Proceedings of the Eleventh Annual Conference on Computer Assurance (COMPASS '96), pp. 213–223, Gaithersburg, Maryland, USA, June 17–21, 1996.

[69] Wietse Venema [et al], Improving the Security of Your Site by Breaking Into It, 1992.

[70] William Stallings, Operating Systems, 2nd Edition. Prentice Hall, 1995.

A Cryptographic Calculations

A.1 Calculation of Session Keys

The key is constructed by doing the following calculations.

1. Add together the two challenges by doing Intel byte order addition of the corresponding 4 byte words.
2. Compute the MD4 hash of the computer password.
3. Do a DES ECB encryption of the operation in 1 using the first 7 bytes of the result in 2 as the key.
4. Do a DES ECB encryption of the result in 3 using the last 7 bytes of the result in 2 as key.
5. The result of 4 is the session key.

A.2 Calculation of Credentials for Secure Channel Establishment

The credentials of the client is calculated in the following way:

1. Do a DES ECB encryption of the client challenge using the first 7 bytes of the session key as the key.
2. Do a DES ECB encryption of the result in 1 using the seventh byte of the session key repeated seven times as the key.
3. The result of 2 is the credentials for the client.

The calculation of the credentials of the server is carried out in the same manner except that the server challenge is used in 1.

A.3 Calculation of Credentials Used after Secure Channel Setup

The client credentials used in messages over the secure channel after setup is completed is calculated in the following manner.

1. Take the current time of the client.
2. Add the time in 1 to the original credentials.
3. Do a DES ECB encryption of the result in 2 using the first 7 bytes of the session key as the key.
4. Do a DES ECB encryption of the result in 1 using the seventh byte of the session key repeated seven times as the key.
5. The result of 4 is the credentials for the client.

The calculation of the credentials for the server is almost identical to the ones described above since the client has to send its time to the server. The only difference is that the server adds one to the time.

A.4 Calculations of RC4 Encrypted Passwords

The calculations of the RC4 encrypted password is done by taking the NT hash and the Lan Manager hash and then perform RC4 encryption on each of them using the session key concatenated with the session key as the encryption key.

A.5 Calculation of Challenge Response

The response is calculated in the same manner for both the NT password hash and the Lan manager password hash. The steps that occur are the following:

1. Add five zeros to the end of the password hash.
2. Split the result in 1 into three 7 byte strings.
3. Do 3 DES ECB encryption of the challenge using the three 7 byte strings as the keys.
4. Concatenate the results in 3 yielding a 24 byte string. This is the response.

A.6 Calculation of SMB Signing

The MAC used to sign a SMB message is calculated in the following way (this applies to the NT password hash as well as the Lan Manager password hash:

1. Concatenate the response in A.5 with the NT hash (In the Lan Manager case take the first 8 bytes of the hash and add 8 zeros to that).
2. Concatenate the key calculated in 1 with the text of the SMB.
3. Do a MD5 hash of the result in 2.
4. The first 8 bytes of 3 is the MAC.

A.7 Challenge-Handshake Authentication Protocol (CHAP)

1. A RAS-enabled client calls the RAS server.
2. The server challenges the client. It sends a one-time value, called session key, to the client.
3. The client encrypts the encrypted password with the session key using DES, and sends it back to the server.
4. The server retrieves the user's encrypted password from the User Account Database and encrypts it with the session key, which it remembers from step 2. This information is compared with the logon information returned by the client.
5. If the information matches, the server checks if the user has dial-in permissions.
6. If the user has dial-in permissions, the user is authenticated and is granted access to the RAS server or both the RAS server and the connected LAN.

B The Domain Logon Process

The following is a description of the events that takes place when a client in a domain is powered up and the first user is logged on to the domain. It contains a network trace, see Figures 19 and 20, and some time-diagrams showing the essential signals and the RPC structures used.

The first thing that happens (beside the introductory registration requests and the browser election) is that the client tries to find a DC to logon to. This is accomplished through message 15-18 in the trace, i.e. the messages in Figure 17. The last two messages in the sequence is to find out the IP address of the DC.

Now a TCP connection is setup between the client and port 139 on the DC followed by a NBT session over this connection. Message 19-23. This session is kept during the whole logon process and probably longer.

In message 24-25 the security level of the SMB session is negotiated. The reply message (25) also contains the 8 byte challenge used later for user access to shares, see section 5.4.3.

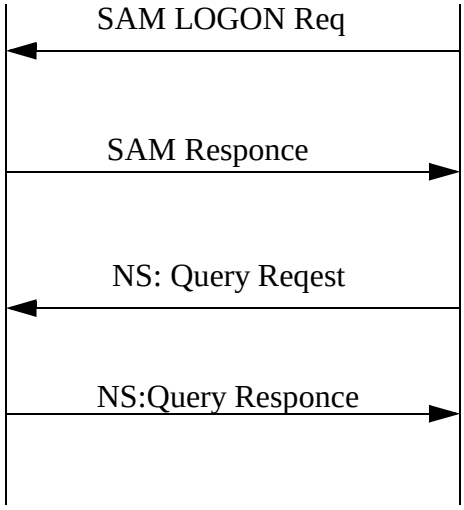

FIGURE 17. The DC discovery process

Messages 26-40 starts with a setup of a NULL session to the IPC$ share of the DC and continues with the mounting of a named pipe *\PIPE\lsarpc*. After that a number of RPC requests and responses is sent over the pipe. We believe that the purpose of these messages is to establish the domain list that is present in the interactive logon window, i.e. the client wants to find out which domains it can logon to through the DC. After that, the pipe is closed but the “lower” sessions remain active.

Messages 41-48 is the establishment of the secure channel. It starts with the mounting of the *\PIPE\Netlogon* named pipe. After the initial RPC binding the *NetrServerReqChallenge()* is sent. This is where the server and clients exchange challenges with each other see section 5.4.1. The credentials are then calculated and sent back in the RPC call *NetrServerAutheticate2()*. If these messages where correct the server and the client now trusts each other.

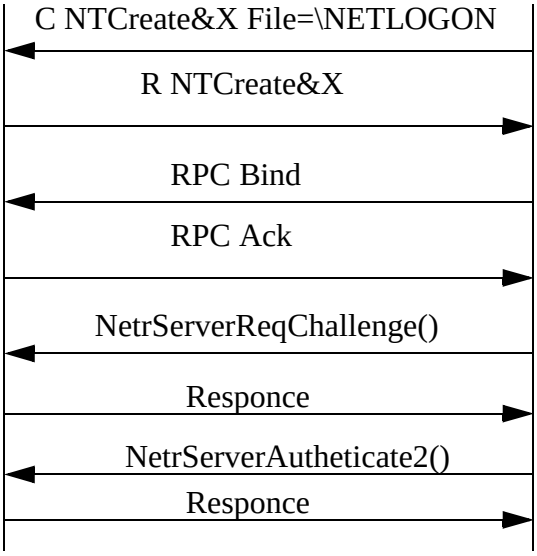

FIGURE 18. Setup of secure channel

Message 49-50 is the logon of the user. The *NetrLogonSamLogon()* will contain the username and the password for the user. The reply will contain most of the information on the user in the SAM database for cashing on the client. The cash we believe is the local SAM database.

The messages 54-57 is a remote logon, i.e. a mounting of a share. From the user to the *\NETLOGON* share. The purpose of this is probably to run a logon script that apparently is not present.

Most of the RPC structures used in the messages above are described in [37].

FIGURE 19. NetMon trace of the domain logon process part 1

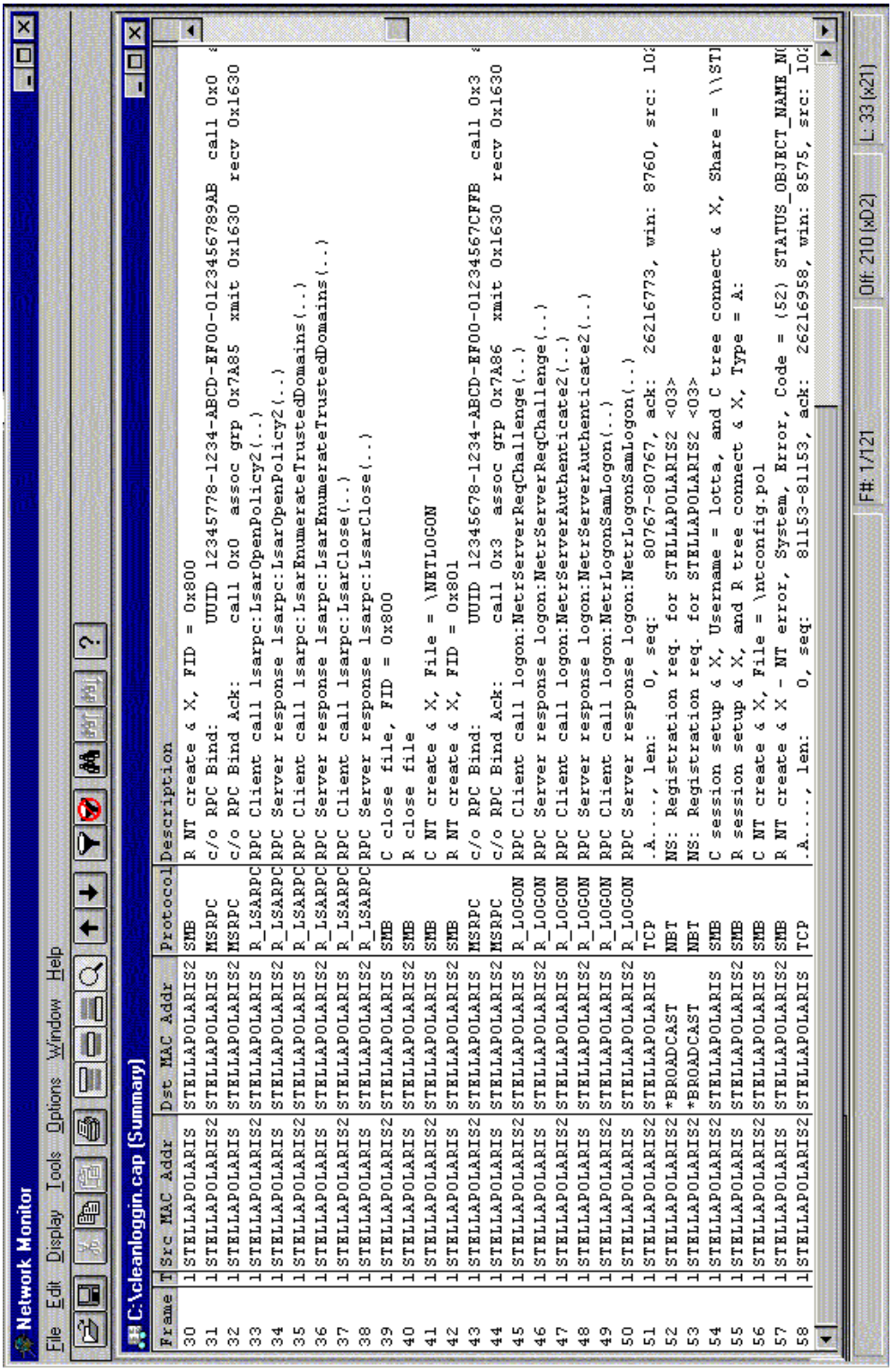

FIGURE 20. NetMon trace of the domain logon process part 2

C SMB Data Structures

The SMB data structure and the *flags* and *flags2* fields according to [31].

```
UCHAR is 8 unsigned bits
USHORT is 16 unsigned bits
ULONG is 32 unsigned bits

typededef struct {
    UCHAR Protocol [4]; // Contains 0xFF, 'SMB'
    UCHAR Command; // The SMB command code
    union {
        struct {
              UCHAR ErrorClass; //The dos error class
              UCHAR Reserved;
              USHORT Error; //The dos error code
            } DosError;
        ULONG Status; // 32-bit error code
        } Status;
        UCHAR Flags; //Flag field see below.
        USHORT Flags2; // Flag field see below.
        union {
             USHORT Pad[6] //To ensure that the section is
                    //12 bytes long.
            struct {
                 USHORT Reserved;
                  UCHAR SecuritySignature[8]; //Reserved
                                  //for MIC
                                  //when SMB
                                  //signing is
                                  //used.
                } Extra;
            };
        USHORT Tid; //Tree identifier.
        USHORT Pid; //Process identifier.
        USHORT Uid; //User identifier.
        USHORT Mid; //Multiplex identifier.
        UCHAR WordCount; // Number of parameter words.
        USHORT ParameterWords[WordCount]; //The parameter
                          //words.
        USHORT ByteCount; // Number of bytes in data buffer.
        UCHAR Buffer[ByteCount];
        }SMB_HEADER;
```

TABLE 7. The flags field

Bit	Name	Meaning
0		Reserved for obsolecent requests. (LOCK_AND_READ, WRITE_AND_CLOSE)
1		Reserved (must be zero)
2		Reserved (must be zero)

Bit	Name	Meaning
3	SMB_FLAGS_CASELESS	1, all pathnames in this SMB must be treated as caseless. 0, all pathnames in this SMB are case sensitive.
4		Reserved (clients must send as zero; servers must ignore).
5		Reserved for obsolescent requests. (SMB_COM_OPEN, SMB_COM_CREATE and SMB_COM_CREATE_NEW)
6		Reserved for obsolescent requests. (SMB_COM_OPEN, SMB_COM_CREATE and SMB_COM_CREATE_NEW)
7	SMB_FLAGS_SERVER_RESP	1, this SMB is being sent from the server in response to a client request. The command field usually contains the same value in a protocol request from the client to the server as in the server to the client. This bit unambiguously distinguishes the command request from the command response.

TABLE 8. The flags2 field

Value	Name	Meaning
0x0001	SMB_FLAGS2_KNOWS_LONG_NAMES	1 in request, the server may return long components in path names in the response.
0x0002	SMB_FLAGS2_KNOWS_EAS	1, the client is aware of extended attributes.
0x0004	SMB_FLAGS2_SECURITY_SIGNATURE	1, the SMB is integrity checked.
0x0008	SMB_FLAGS2_RESERVED1	Reserved for future use.
0x0040	SMB_FLAGS2_IS_LONG_NAME	1, the path name in the request is a long name.
0x0800	SMB_FLAGS2_EXT_SEC	1, the client is aware of Extended Security negotiation.
0x1000	SMB_FLAGS2_DFS	1, any request pathname in this SMB should be resolved in the Distributed File System.
0x2000	SMB_FLAGS2_PAGING_IO	1, indicates that a read will be permitted if the client does not have read permission but does have execute permission. This flag is only useful on read request.
0x4000	SMB_FLAGS2_ERR_STATUS	1, specifies that the returned error code is a 32-bit error code in Status.Status. Otherwise the Status.DosError.ErrorClass and Status.DosError.Error field contains the DOS-style error information. Then passing NT status codes is negotiated, this flag should be set for every SMB.
0x8000	SMB_FLAGS2_UNICODE	1, any field of datatype STRING in this SMB message are encoded as UNICODE. Otherwise, they are in ASCII.

D Utility Programs

In this appendix, we will present some utilities that we believe could be useful for an NT administrator. Some of them were used during our study. We categories them according to the structure presented in section 6.

D.1 Security Analysis Programs

D.1.1 L0phtcrack

L0phtcrack (pronounced “loftcrack”), is as far as we know, the most popular password cracking program for NT, which can be used to recover both the Lan Manager password and the NT password, stored in the SAM database. Version 1.x of this program takes as input a file with user information, including both the username and the password. Such a file can be created by the PWDump utility, see appendix D.2.3. L0phtCrack can optionally take a dictionary file as input. This type of attack is often referred to as a dictionary attack. Alternatively, L0phtCrack gives the attacker the capability to apply a brute force attack on the entire key space. The utility is distributed with both a graphic and a character user interface.

Recently, a new version, 2.x, of this program has been announced. The creators have tuned version 1.x as well as added new functionality. First, L0phtCrack takes advantage of multiprocessor machines. Second, version 2 is able to retrieve password hashes from the network. Third, the new version even accepts the SAM database as input. Forth, it has become a commercial product.

D.1.2 Crack for NT

The very popular UNIX password cracking program, called Crack, is nowadays ported to, and freely available for, NT. However, we have not used it in our experiments, because we have been fully satisfied with the functionality offered by L0phtCrack.

D.1.3 C2CONFIG

The C2 configuration tool can be used to compare the current configuration of an NT system with the C2 level security requirements, see [65]. C2CONFIG is shipped on Microsoft's Windows NT Resource Kit, which is available in bookstores.

D.1.4 DumpACL

The DumpACL utility reports various security information about a particular NT installation. A trail version of this program is available at Somarsofts Web site, see appendix E.

DumpACL offers a number of features, including:

- Dumps permissions for file systems, shares, Registry and printers.
- Dumps policies, rights and trusts.
- Dumps users and groups.

- Outputs that are suitable as input to spreadsheets.
- Can answer questions like "Show all files for which user A has Take Ownership permission".

We believe this tool to be valuable for any NT system administrator. To gain maximal profit of this utility, it should be used on a regular basis to verify that the NT system has not been compromised.

D.1.5 Kane Security Analyst (KSA)

In [64], the author describes a tool called KSA, which can produce essentially the same information as DumpACL and C2CONFIG together. In addition, KSA is able to detect intrusions based on statistical algorithms applied on gathered data. The idea is to find out whether any particular behavior is significantly anomalous. Another feature presented by Tom Sheldon is the built-in expert knowledge base, which includes security industry "best practices" to be used for comparison with the security attributes in the current system. We have not used this utility in our experiments. More information can be found on their web site (www.intrusion.com), see appendix E.

D.1.6 Internet Scanner for NT (IS)

IS for NT is an NT port of the company ISS well known utility Internet Scanner. This utility will go through the system and test for known weaknesses. Both general and NT specific weaknesses are tested. For a detailed description on exactly which tests are carried out, see appendix E.

D.2 Information Retrieval Programs

D.2.1 Windows NT Password Dump Utility (PWDump)

The PWDump utility dumps the password database of an NT computer in the following format:

<user>:<id>:<lanman pw>:<NT pw>:<comment>:<home dir>:

Where <user> is the user name on NT, <id> is the last 32 bits of the SID, <lanman pw> is the Lan Manager password hash, <NT pw> is the NT password hash. This fields are the important one. The <comment> and <home dir> fields contain information such as the user's full name, description and home directory as specified in the NT User Manager. PWDump can be used on both the local machine and remote machines. However, in order to run the program the user has to log on as Administrator. Here follows a sample output from PWDump that was distributed together with L0phtCrack, 1.x.

Administrator:500:73CC402BD3E791756C3D3B817E02809D:C7E2622D76D3F00
1CF08B0753646BBCC:Built-in account for administering the computer/domain::
Guest:501:NO PASSWORD**********************:NO PASSWORD**********************
*:Built-in account for guest access to the computer/domain::
Joe:1008:3815B591E3ADC98431A60ABC6B5DA940:0934A0D928B63A7D4AB0A7CE EC35C7F2::
:BillG:1010:5ECD9236D21095CE7584248B8D2C9F9E:C04EB42B9F5B114C86921C
4163AEB5B1:::
fredc:1011:3466C2B0487FE39A417EAF50CFAC29C3:80030E356D15FB1942772D CFD7DD3234:::

william:1012:DBC5E5CBA8028091B79AE2610DD89D4C:6B6E0FB2ED246885B985
86C73B5BFB77:::

D.2.2 SAMDump

SAMDump is a utility written by Dmitry Andrianov. It performs the same tasks as PWDump, see D.2.1, but instead of using system calls it reads the SAM file directly. This means that it has to be run with a “non live” SAM file e.g. a copy of the “live” SAM file since the “live” SAM file is looked. However, one does not need to be administrator to run the program. The author of the program points out that it is on his personal incomplete knowledge of the SAM file structure and that it performs no error checks and could therefore be buggy.

D.2.3 Access to functions and variables in NTOSKRNL (NTExport)

NTExport is a tool that can be used to generate a kernel-mode device driver library, including undocumented functions and variables which are exported from the NTOSKRNL. It supports only NT 4.0 x86 versions of NT. The question is: how can this program find information about these functions and variables? NTExport combines information in the NTOSKRNL.EXE and its corresponding debug symbol file (NTOSKRNL.DBG). The latter can be found under \support\debug\i386\symbols\exe on the distribution CD. The steps in using this tool are as follows:

1. Use NTExport to dump the names of all unexported symbols. The list (-l) option is used for this purpose.
2. All symbols that you want to export are listed in a file called exports.lst.
3. Use NTExport again to process the particular Service Pack that is used. This step will generate at least two files with information about the symbols specified in exports.lst. The number of files depends on which Service Pack is used. Note, these files are for NTExport’s internal use.
4. Execute the command: ntexport -g. It will generate a kernel-mode device driver library with the name export.lib.

It is now possible to use the newly created library. The Cacheman program, see http://www.sysinternals.com/, is an example of a program that relies on this tool. Cacheman allows a user to tune caching parameters on the fly.

D.2.4 NT Registry Monitor (NTRegmon)

NTRegmon is a program that displays all registry activity taking place on a NT system. Version 2 of this program can be used on both NT 3.51 and NT 4.0. NTRegmon consists of two parts: a device driver which uses a technique called kernel-mode system call hooking, and a graphical user interface (GUI). When the GUI program is started, it dynamically loads the driver. This registry monitor sees all user level registry activity, and even most kernel registry calls.

D.2.5 NT File System Monitor (NTFilemon)

NTFilemon can be used to monitor file system activity for NT 3.51 and NT 4.0. This utility consists of a device driver and GUI, like NTRegmon. The device driver is a type of driver known as a filter driver, which means that it layers itself above the file system drivers. After installation, NTFilemon can see I/O requests pass to, and from, the file systems. All types of file system drivers that have an associated driver letter may be monitored, e.g. FAT and NTFS partitions. The GUI application will automatically load the driver.

D.2.6 NT Process Monitor (NTPmon)

With NTPmon, information about all process activity on an NT 4.0 system can be gathered and displayed in a window. The utility program consists of a device driver and a GUI, like NTRegmon and NTFilemon. The driver uses a number of undocumented hooking functions, which collects information about activity such as process creation, process deletion, thread creation, thread deletion, and optional context switches. The latter is only present in multiprocessor builds of NT, and is by default disabled.

D.2.7 Object Manager Name Space Viewer (WinObj)

WinObj is an NT application that displays information on the NT object manager's name space. This means that the program shows information about various operating system components. WinObj uses the native NT API, which, for example, provides routines to allow user programs to browse the name space. It is also possible to query the status of objects located there.

D.2.8 Microsoft Network Monitor

The Microsoft Network Monitor is Microsoft’s own program for monitoring network activities. It comes in two variants. One of the variants is on the server installation CD. This program has sever limitations. It can only read broadcast messages and traffic directed to the machine that runs the program. Microsoft claims that this is for security reasons. It also has limitations in how it interprets the messages it receives. The copy we tried did for example not translate the RPC messages. The other variant is part of Microsoft SMS. This program is a proper monitor that monitors all traffic on the net as long as the network interface card can handle promiscuous mode. It also, as far as we could see, parses the messages fully except for password fields.

The network monitor also has the ability to fetch information from network agents across the network. This could come in handy, e.g. in a switched network environment where an agent could pickup traffic on a switched segment. The program can also detect other instances of Microsoft Network Monitor that is active on the network.

D.3 Maintenance Programs

D.3.1 NTFSDOS

NTFSDOS is a program that makes it possible to read NTFS volumes from MS-DOS or Windows 95. Today, there exist programs that are also able to modify files on NTFS volumes, see descriptions of NTFS Tools, ERC Commander and NTRecover below, from MS-DOS. The implications of this are that any system with a floppy drive that can boot from another OS is insecure.

D.3.2 NTFS Tools

NTFS Tools is a package with two utilities, NTFSCopy and NTFSRen. It is a add-on for NTFSDOS that provides limited modify capabilities for NTFS volumes from MS-DOS. The package is intended to be used to repair system errors in NT. Each utility supports a single operation, i.e. file copy and file rename. However, both have certain limitations. The copy command is used to overwrite an existing file, and will only success if the source file and the destination file are of the same size. The rename command will not let a user specify a new file name, it is chosen by the program. Furthermore, the chosen name must not already exist.

D.3.3 ERD Commander

ERD, short for Emergency Repair Disk, Commander let an NT system be booted from a floppy. The utility is shipped with a setup program, which will allow the user to create a set of NT boot floppies. After the boot floppies are created, a stripped down version of NT can be started from the floppy. Indeed, NT does not even need to be present on the computer. All NT supported file systems will be visible. The complete set of commands ERD commander provides is in Table 9.

TABLE 9. Commands in ERD Commander as described on the Web site.

Command	Description
ACCESS	Gives Everyone full access to a files or directories.
CD	Displays the name of or changes the current directory.
CLS	Clears the screen.
COPY	Copies one or more files to another location.
DEL	Deletes one or more files.
DIR	Displays a list of files and subdirectories in a directory.
EXIT	Quits ERD Commander and reboots the system.
HELP	Provides Help information for ERD Commander commands.
MAP	Displays drive letter to partition mapping.
MKDIR	Creates a directory.
MORE	Displays the contents of a file.
MOVE	Moves one or more files from one directory to another directory.
RENAME	Renames a file or files.
RMDIR	Removes a directory.

TABLE 9. Commands in ERD Commander as described on the Web site.

Command	Description
XCOPY	Copies files and directory trees.
VER	Displays ERD Commander version number

Both a read-only and a read/write version of the program exists. If the read-only version is used, the following commands are disabled: ACCESS, COPY, DEL, MOVE, RENAME and XCOPY.

D.3.4 NTRecover

NTRecover is an NT utility that can be used to recover a dead-system. It can only be used on x86 based NT installations. Using NTRecover, it is possible to access file systems even when the computer fails to boot NT. The following equipment is necessary:

- the non-booting system (client)
- a working NT system with NT version 3.51 or 4.0 (host)
- a standard null-modem serial cable, used to connect the client with the host.

The utility consists of both client and host software, where the client program is executed on the failed system, and the host is running on the working system. The failed system is booted off a floppy directly to the NTRecover program. Once the client and the host program are started and configured on their respective computers, native NT file systems such as NTFS and FAT on the client machine are visible on the host computer. There are two versions of the utility. A free trail version that only permits read-only access to the damaged system, and a retail version which offers both read and write access.

D.3.5 Ghost

Ghost is a utility that can be used to install completely new copies of operating systems, such as Windows 95, NT and OS/2, on a local partition. The source files can reside on either local or on remote computers. The program runs under MS-DOS. However, it can be used to reinstall operating systems on FAT partitions as well as on NTFS partitions. Ghost can also be used for backup purposes. An evaluation copy of the product can be retrieved from the Internet, see appendix E.

D.4 Keyboard Filtering (ctrl2cap)

Ctrl2Cap is a program written by Mark Russinovich that demonstrates how to write kernel mode drivers for NT. Its main purpose is to convert the control (ctrl) key to the shift key, so that the program in itself is not harmful, but the implications inherent in this are. We believe that by extending this driver or writing a similar program it is possible to catch and manipulate any key pressed on the keyboard before NT sees it, which means that it is feasible to snoop, and save passwords when they are entered. It is also possible to filter out the ctrl-alt-delete key combination which would totally lock the workstation since no user will be able to logon to the that station. This in turn would cause trouble on workstations shared among several users, or could be used by a mali-

cious user to gain access to the administrator password on his personal workstation by doing as follows.

1. Install a keyboard filter on his personal workstation.
2. Claim that something is wrong (or cause something to be wrong) with his workstation and ask the system administrator for help.
3. When the system administrator enters his password to fix the claimed problem that password is filtered and stored by the filtering program.

This type of combined attack of harmful tools and social engineering [32] might not be common but they are definitely possible as long as it is possible to snoop the keyboard.

E URLs

This is collection of some useful URLs consulted during our work.

http://www.sysinternals.com

http://www.omna.com/Yes/MWC/PRS-index.htm

http://www.r2m.com/windev/

http://www.microsoft.com/security

ftp://www.microsoft.com

http://www.somarsoft.com/security.htm

ftp://ftp.ora.com/pub/examples/windows/win95.update/schulman.html

http://www.geek-girl.com

http://www.l0pht.com/advisories/l0phtcrack.txt

http://www.intrusion.com

http://www.iss.net

http://www.ghostsoft.com

http://www.ntsecurity.net

http://www.sans.org

F Newsgroups

This is a list of usenet newsgroups used in our work.

alt.2600Hz.crack

alt.2600Hz.hackers

comp.os.nt.admin.security

G Terminology

ACE	Access Control Entry
ACL	Access Control List
APC	Asynchronous Procedure Call
API	Application Programming Interface
ARPA	Advanced Research Projects Agency
BDC	Backup Domain Controler
CERT	Computer Emergency Responce Team
CHAP	Challenge-Handshake Authentication Protocol
CDFS	Compact Disc File System
CHKDSK	CHecK DiSK; A consistency checking program.
CIA	Confidentiality, Integrity and Availability
CIFS	Common Internet File System
DACL	Discretionary ACL
DC	Domain Controler
DCE	Distributed Computing Environment
DLC	Data Link Control
DLL	Dynamic Linked Libraries
DES	Data Encryption Standard
DPC	Deferred Procedure Call
ECB	Electronic Code Book
FAT	File Allocation Table
GUI	Graphic User Interface
HAL	Hardware Abstraction Layer
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPC	InterProcess Communication
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
ISO	International Organisation of Standardisation
LAN	Local Area Network
LM	Lan Manager
LPC	Local Procedure Call
LSA	Local Security Authority
MAC	Message Authentication Code
MIDL	Microsoft Interface Definition Language compiler
MP	PPP Multi-Link Protocol
MS-DOS	Microsoft Disk Operating System
MS-NET	Microsoft Network
MSFS	Mail-Slot File System
NBT	NetBIOS over TCP.
NCSC	National Computer Security Center
NDIS	Network Driver Interface Specification

NetBEUI	NetBIOS Extended User Interface
NetBIOS	Network Basic Input/Output System
NFP	NetBEUI Frame Protocol
NFS	Network File System
NIS	Network Information Service
NPFS	Named Pipe File System
NSA	National Security Agency
NT	Windows New Technology
NTFS	NT File System
NTLM	NT Lan Manager
NTOSKRNL	NT Operating System KeRNeL
OS	Operating System
OSF	Open Software Foundation
OSI	Open Systems Interconnection
PAP	Password Authentication Protocol
PDC	Primary Domain Controler
POSIX	Portable Operating System Interface based on uniX
POTS	Plain-Old Telephone Service
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PSTN	Public Switching Telephone Network
RM	Reference Model
RAS	Remote Access Service
SAM	Security Accounts Manager
SAS	Security Attention Sequence
SID	Security ID
SLIP	Serial Line Internet Protocol
SMS	System Management Server
SMB	Server Message Block
SP	Service Pack
SPAP	Shiva PAP
SRM	Security Reference Monitor
SUID	Set User ID
TAPI	Telephone Application Programming Interface
TCB	Trusted Computer Base
TCP	Transmission Control Protocol
TDI	Transport Driver Interface
UDP	User Datagram Protocol
UID	User ID
URL	Uniform Resource Locator
VDM	Virtual Dos Machine
VFAT	Virtual FAT
VPN	Virtual Private Network
WAN	Wide Area Network
Win32	Windows 32-bit user interface

WinLogon	Default logon process in NT
WWW	World Wide Web
X.25	A standard for packet switching networks