A Comparison of the Security of Windows NT and UNIX†

Hans Hedbom[1,2], Stefan Lindskog[1,2],
Stefan Axelsson[1] and Erland Jonsson[1]

[1]Dept of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, SWEDEN
{sax, Erland.Jonsson}@ce.chalmers.se

[2]Dept of Computer Science
University of Karlstad
S-651 88 Karlstad, SWEDEN
{Hans.Hedbom, Stefan.Lindskog}@hks.se

Abstract

This paper presents a brief comparison of two operating systems, Windows NT and UNIX. The comparison covers two different aspects. First, we compare the main security features of the two operating systems and then we make a comparison of a selection of vulnerabilities most of which we know have been used for making real intrusions. We found that Windows NT has slightly more rigorous security features than "standard" UNIX but the two systems display similar vulnerabilities. The conclusion is that there are no significant differences in the "real" level of security between these systems.

†Presented at the Third Nordic Workshop on Secure IT Systems, NORD-SEC'98, 5-6 November, 1998, Trondheim, Norway.

1. Introduction

It has been claimed that the security of Windows NT is far better than that of previous commercial operating systems. In order to verify (or refute) this statement we have made a brief comparison of the security of Windows NT to that of UNIX. UNIX was selected as a reference since it is well-known and widely spread. Thus, the target systems were (1) a networked Windows NT 4.0 and (2) UNIX with NFS (Network File System) and NIS (Network Information System). The reason for choosing NFS and NIS is that the two operating systems constitute comparable environments, i.e. they have similar network functionality. It should, however, be stressed that UNIX comes in many different versions, so our reference is not completely unambiguous. Still, we believe that this fact does not reduce the value of the work presented.

The comparison covers two different aspects. First, we compare the main security features of the two operating systems and then we make a comparison of a selection of vulnerabilities most of which we know have been used for making real intrusions. Those have been gathered from intrusion experiments carried out at the department of Computer Engineering at Chalmers University of Technology for data collection purposes [3], [20] or from our own system analysis [7]. Some data has been taken from other publicly available sources.

For the comparison of the vulnerabilities of the two systems, we have used a taxonomy of intrusion techniques suggested by Lindquist and Jonsson [15]. The taxonomy has proven useful for classifying realistic intrusions and it covers all three security attributes: confidentiality, integrity and availability.

In the following, section 2 compares the security features, while section 4 gives a systematic comparison of weaknesses in Windows NT and UNIX. The taxonomy used for classifying the vulnerabilities in section 4 is described in section 3. Section 5 discusses the results and concludes the paper.

2. Comparison of Security Features

2.1 Background

The Windows NT operating system was developed by Microsoft Inc. and was first released in 1992. Windows NT has support for processes, threads, symmetric multiprocessing, distributed computing, and uses an object model to manage its resources. The structure of Windows NT is a hybrid between the layered model and the client/server model [5]. The former is used in the executive, which is the only part executing in kernel mode, while the latter is used to (1) provide the user with multiple operating system environments, e.g. Windows, MS-DOS, OS/2 and POSIX[1] and (2) implement parts of the operating system.

The UNIX operating system was first developed in the early seventies at AT&T Bell research laboratories. It is traditionally implemented as a single monolithic kernel, that

1. Portable Operating System Interface based on uniX

runs in kernel mode, while all user programs run in user mode. The kernel contains code for the file system, device drivers as well as code for process management [1], [16]. However, UNIX has always managed large parts of many system functions, such as networking etc, outside the kernel, in user mode processes.

In the following of this section is presented a number of security mechanisms for UNIX and Windows NT. These mechanisms have primarily been taken from the TCSEC [25] with some modifications. The mechanisms represent different aspects of security and are meant to provide a broad coverage of the area. Differences and similarities between the security mechanisms of the two operating systems are discussed in a concluding subsection.

2.2 Identification

2.2.1 Windows NT

User identification is done by a username that is mapped onto an internal Security IDentifier (SID). A SID is a numeric value that is unique within a domain (and with high probability between domains as well). When a new account is created on the system a SID is created and stored together with this account. SIDs are never reused so a newly created account can not get the SID of a previously deleted account.

2.2.2 UNIX

A user is identified by a username, which is given when the user logs on to the system. Internally a user is identified with a User IDentification number (UID), which is a numeric value selected by the system administrator at the time the account is created. In most cases, selecting unique UIDs for each user is a good idea [6], though not strictly required. The mapping of username to UID is kept in the file `/etc/passwd`, but is today often centrally managed by NIS. The super user (or root) has UID 0 (zero). Every user belongs to one or more groups. A group is identified with a group identification number (GID).

2.3 Authentication

2.3.1 Windows NT

User authentication is done using passwords. Windows NT stores passwords together with SIDs and other information on the user in a database handled by the Security Accounts Manger (SAM) subsystem. Two hashed versions of the password are stored, LM-hash and NT-native, unless the system is told to just use one. The NT-native variant is stored using MD4 and the LM-hash using a variant of DES.

2.3.2 UNIX

After identification, UNIX will also request a password to authenticate the user's identity. When the user has entered the password, it is encrypted using a modified DES algorithm described in [6], and compared against the encrypted password stored in `/etc/passwd` (or the NIS database). If the two match, the user has proven to be a

legitimate user in the system. The file `/etc/passwd` is readable for everyone in the system, which makes it sensitive for password attacks. A solution to this problem is to use what is known as a "shadow" file (`/etc/shadow`). The whole idea is then to move the encrypted passwords from `/etc/passwd` to `/etc/shadow`, and make the latter not readable by normal users.

2.4 Access Control

2.4.1 Windows NT

Every object in the system has an Access Control List (ACL) associated with it. This list consists of a number of Access Control Entries (ACE). Every ACE is associated with a user (or a group) SID and holds the actions that this user is allowed or disallowed to perform on this object. ACEs that disallow are put before ACEs that allow in the ACL. A user that does not have an ACE in the ACL has no access at all to that object.

An object can also have a NULL ACL or an empty ACL. If the object has a NULL ACL this object has no restrictions. An empty ACL on the other hand means that nobody can access this object in any way. A newly created object is usually given the ACL of its creator by default.

When a user is authenticated to the system a token is created for this user. This token is called the primary token. It contains, among other things, the SID for the user and the SIDs of the groups that this user is a member of. This token is compared with an object’s ACL to grant (or deny) the user access to this object.

2.4.2 UNIX

UNIX's access control is implemented through the file system. Each file (or directory) has a number of attributes, including a filename, permission bits, a UID and a GID. The UID of a file specifies its owner.

The permission bits are used to specify permissions to read (r), write (w) and execute (x) the file for the user, for the members of the user's group, and for all other users in the system. The permissions: `rwxr-x--x` specify that the owner may read, write and execute the file, while the group members are allowed to read and execute it, while all others only may execute the file. A dash ("-") in the permission set indicates that the access rights are disallowed. Most systems today also support some form of ACL schemes.

Furthermore, each process in UNIX has an effective and a real UID as well as an effective and a real GID associated with it. Whenever a process attempts to access a file, the kernel will use the process's effective UID and GID to compare them with the UID and the GID associated with the file to decide whether or not to grant the request

2.5 Auditing

2.5.1 Windows NT

The Security Reference Monitor (SRM) and the Local Security Authority (LSA) together with the Event Logger handle the auditing in Windows NT. 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. For details on event groups see [18]. If auditing applies and what is to be audited is determined by the Audit Policy that 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 subsystem or server by SRM (in some cases by LSA). Requests from the executive is always carried out. Servers, on the other hand, need the Audit privilege for SRM to honor their 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 has expanded some fields and compressed others. The Event Logger commits the audit record to permanent storage.

2.5.2 UNIX

Traditionally the UNIX kernel, and system processes, store pertinent information in log files, either locally, or centraly on an network server, via the the flexible and configurable *syslog* facility [13]. In addition, many modern UNIX systems supports a more comprehensive type of auditing known as C2 audit. This is so-name because it fulfills the audit requirements for the TCSEC C2 security level [25].

2.6 Networking

2.6.1 Windows NT

The distributed parts of Windows NT rely heavily on Server Message Block (SMB). This is an application level protocol used by Microsoft for a number of thinks. Among those are authentication, RPC and the Common Internet File System protocol (CIFS) [8], [11], [12]. In the Windows NT environment an SMB is carried on top of a NetBIOS over TCP/IP (NBT) session [21], [22], including UDP as a carrier for NetBIOS as well. There are a number of things to be said about CIFS/SMB, regarding security. First, it is possible to establish so-called NULL sessions, i.e. sessions with no username. Machines that exchange RPCs through named pipes frequently do this. Second, all logging in Windows 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 IP address forging, but it is even simpler to forge a computer name. Third, the protocol is very outspoken, and will freely give away much information about what is going on. Last, all in all, there is too much trust in client machines behaving in a non-malicious manner. For a more in-depth description on some of the weaknesses in CIFS/SMB, see for example [10].

2.6.2 UNIX

Most modern UNIX machines are networked. The default networking support is based on TCP/IP. Remote terminal access, remote file transfer and remote command execution are provided through a set of programs (rlogin, rcp, rsh, etc). These are collectively known as the 'r' commands. The Network File System (NFS) [23] adds support for several hosts to share files over the network, while the Network Information System (NIS) [9], formally known as the Sun Yellow Pages, allow hosts to share system databases containg data concerning user account information, group membership, mail aliases etc. via the network, to facilitate centralised administration of the system.

The 'r' commands are not secure, due to a number of facts. First, they assume that all hosts in the network are trusted to play by the rules, e.g. any request coming from a TCP/IP port below 1024 is considered to be trusted. Second, they use address-based authentication, i.e. the source address of a request is used to decide whether or not to grant a service. Third, they send clear text passwords over the network.

Before an NFS client can access files on a file system exported by an NFS server, it needs to mount the file system. If a mount operation succeeds, the server will respond with a ”file handle,” which is later used in all accesses to that file system in order to verify that the request is coming from a legitimate client. Only clients that are trusted by the server are allowed to mount a file system. The primary problem with NFS is the weak authentication of the mount request [4], which is based on IP addresses, that may be faked.

NIS can be configured to perform (1) no authentication, (2) traditional UNIX authentication based on machine identification and UID, or (3) DES authentication [9]. (3) provides quite strong security, while (2) is used by default by NFS. According to [6], a forth authentication method based on Kerberos [24] is also supported by NIS. Both servers and clients are sensitive to attacks, though Hess et al. [9] are of the opinion that the real security problem with NIS resides on the client side. It is easy for an intruder to fake a reply from the NIS server. Such an intrusion is further described in subsection 4.1.3.2.

2.7 Impersonation

The user of the system must be able to perform certain security critical functions on the system normally exclusive to the system administrator, whithout having access to the same security permissions as her/him. One way of giving users controlled access to a limited set of system privileges is for the system to allow the execution of a specified process by an ordinary user, with the same permissions as another user, i.e. system privileges. This specified process can then perform application level checks to insure that the process does not perform actions that user was not intended to be able to perform. This of course places stringent requirements on the process in terms of correctness of execution, lest the user be able to circumvent the security checks, and perform arbitrary actions, with system privileges.

2.7.1 Windows NT

Every thread that executes in the system has the possibility to have two tokens. One of them is the primary token and the other is a so called impersonation token. This is a token given to the thread by another subject which allows the thread to act on that subjects behalf. Impersonation token is a full or restricted variant of that subject's primary token.

2.7.2 UNIX

Two separate but similar mechanisms handle impersonation in UNIX, the so called set-UID, (SUID), and set-GID (SGID) mechanisms. Every executable file on a filesystem so configured, can be marked for SUID/SGID execution. Such a file is executed with the permissions of the *owner*/*group* of the file, instead of the current user. Typically certain services that require super user privileges are wrapped in a SUID-super user program, and the users of the system are given permission to execute this program. If the program can be subverted into performing some action that it was not originally intended to perform, serious breaches of security can result.

2.8 Discussion of Security Features

There are similarities in the security features of the two systems. Our opinion is, however, that the Windows NT systems mechanisms are more ambitious than the standard UNIX mechanisms. This is of course partly due to the fact that Windows NT is of a later date than UNIX. However, most Windows NT security features are available for modern UNIX systems.

It is interesting to note that both systems contain mechanisms for impersonation, while this has been an almost endless source of security problems with UNIX, it will be interesting to see if Windows NT will come to suffer from the same problems.

Finally, it is worth mentioning that a system need not be more secure than another just because it has more and better security features. Unfortunately both Windows NT and UNIX have most of these features disabled by default for convenience reasons. Thus, it takes an active decision to introduce security during installation.

3. Classification of Computer Security Weaknesses

A taxonomy is used to categorize phenomena, which, in general, make system studies possible, or at least easier. By using an established classification scheme of intrusion techniques, different systems can be compared based on intrusion data. One such scheme was suggested by Neumann and Parker in [19], and was refined by Lindquist and Jonsson [15]. We chose the latter in our comparison.

The selected taxonomy consists of three categories: bypass of intended controls (NP5), active misuse of resources (NP6) and passive misuse of resources (NP7), see Table 1.

TABLE 1. Classification of Intrusion Techniques

Category		
Bypassing Intended Controls (NP5)	Password Attacks	Capture
		Guessing
	Spoofing Privileged Programs	
	Utilizing Weak Authentication	
Active Misuse of Resources (NP6)	Exploiting Inadvertent Write Permissions	
	Resource Exhaustion	
Passive Misuse of Resources (NP7)	Manual Browsing	
	Automated Searching	Using a Personal Tool
		Using a Publicly Available Tool

Bypass of Intended Controls (NP5). Is the act of circumventing mechanisms in the system that are put there to stop users from performing unauthorized actions.

Active Misuse of Resources (NP6). Is the act of maliciously using permissions or resources for which the user accidentally have authorization to.

Passive Misuse of Resources (NP7). Is the act of searching for weak or erroneous configurations in the system without having authorization to do so.

Descriptions of the applicable subclasses of each category are further presented in section 4. Other taxonomies exist as well, though none, the two mention above included, is perfect. A survey of previous work in the field is presented in [15].

4. Systematic Comparison of Weaknesses

In this section we will give examples of security weaknesses in both the Windows NT system and the UNIX system. The different attacks are categorized according to the taxonomy presented in section 3. Vulnerabilities in the same class are compared with each other. In the cases where the attacks are identical in both systems only one example is given and discussed.

4.1 Bypassing Intended Controls (NP5)

4.1.1 Password Attacks

To learn the password for a certian account, the attacker can either capture a plain text password, or a cryptographically hashed password. In the later case the attacker can then attempt to either brute force the password, i.e. try all possible combinations, or with greater chance of quick success, try likely passwords, e.g. the last name of the user, backwards, in all capital letters.

4.1.1.1 Password Capturing Attacks in Windows NT

Windows 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 older of these variants sends plain-text passwords over the network. The server can always try and suggest a more secure variant but it is the client that, unless there is a share-level versus user-level incompatibility, sets the level of security. The server as default will always accept plain-text passwords as valid authentication [10]. An 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, 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.

4.1.1.2 Password Capturing Attacks in UNIX

Some of the protocols, e.g. telnet and ftp, and the ‘r’ commands used in the UNIX environment sends plain-text password information. Thus all an attacker has to do is sniffing the network for these activities and copy the password off the wire.

4.1.1.3 Comparison of Password Capturing Attacks

The weaknesses described here could all be categorized as legacy problems. Windows NT has this problem because it tries to be backward compatible and the same could be said about the UNIX environments that uses the protocols described above. If compatibility is not a requirement, a Windows NT Server or Client could be configured not to accept older variants of the SMB protocol and in the UNIX environment more secure protocols that has the same functionality, e.g. SSH [26], could be used.

4.1.1.4 Password Guessing Attacks in Windows NT

In Windows NT, the passwords are stored in Lan-Manager-hash, or LM-hash, format as well as in NT-native format. The LM format has a much weaker encryption algorithm than the native format, e.g. in Lan Manager only upper-case characters are allowed and the 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. The public domain program *L0phtCrack* can be used for brute force attacks as well as for dictionary attacks. To gather encrypted passwords for later cracking one can either try to get hold of a copy of the SAM database or sniff the network. Programs exists for both purposes.

4.1.1.5 Password Guessing Attacks in UNIX

In UNIX, encrypted passwords are traditionally stored in the /etc/passwd file. This file is readable for every legitimate user on the system. In some UNIX versions the encrypted passwords are instead stored in a shadow file, which is only readable by the super user. Since the introduction of NIS it is also possible to obtain passwords from the network 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 with for example *Crack* [17].

4.1.1.6 Comparison of Password Guessing Attacks

Both Windows NT and UNIX are susceptible to password attacks. Not really because of weak encryption methods but due to the fact that they allow too short or easily guessed passwords. An alternative way of looking at this is to say that they allow passwords to have a longer lifetime than the time it takes to guess or break them. Another question is or course, if users can be trusted to choose more secure passwords, i.e. that are more difficult to guess. The authors are not all that convinced that this is the case. Neither are we convinced that automatically generated passwords is a solution to this dilemma.

4.1.2 Spoofing Privileged Programs

If a program, that is trusted by more than a user, can be lured to perform actions on behalf of that user, s/he has access to more privileges than intended. The program can be fooled into giving away information, changing information or cause denial of service attacks. There are a number of examples of exploits in this category, including *GetAdmin* [7] and the X terminal emulator logging vulnerability [14]. Below we present a vulnerability that is identical for both systems.

4.1.2.1 Spoofing Privileged Programs in Windows NT and UNIX

When an IP packet handler gets a message that is to large to fit into a packet it normally breaks this message into fragments and puts every fragment in a separate IP packet. All but the last of these packets has the More Fragments (MF) bit set in the header. The packets also have an offset counter that tells the other side where in the original message this fragment fits.

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. The function that reads data from the packet buffer calculates the number of bytes to read by taking the end counter and subtract the offset-counter. In this case it will end-up with a request to read a negative number of bytes and therefore read too much data, and by doing this crash the system. All in all this is a

traditional buffer overrun error. Windows NT and many UNIX versions, as well as routers, are sensitive to a *Teardrop* attack. The reason for this can be that the same reference implementation of the TCP/IP stack has been used in all systems.

4.1.3 Utilizing Weak Authentication

Utilizing Weak Authentication means taking advantage of the fact that the system does not properly authenticate the originator of a certain request. In the subsections below we will exemplify this class by presenting two man-in-the-middle attacks.

4.1.3.1 Utilizing Weak Authentication in Windows NT

In Windows NT a normal remote logon occurs as follows:

1. The client tries to setup a SMB connection to the exported service on the remote computer.
2. The server will send the client a challenge
3. The client will calculate two 24-byte strings using the challenge and the LM and NT passwords, and send those in an SMB message to the remote computer. After that the user is considered authenticated.

The protocol can, however, be fooled as the following attack shows. It is described by Dominique Brezinski in [2] and relies on the fact that there is nothing that prevents an attacker from masquerading as the server. Brezinski 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

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’s credentials

4.1.3.2 Utilizing Weak Authentication in UNIX

When NIS is added to the basic operating system similar attacks are possible in UNIX as well. One described by David K. Hess et al. in [9] goes as follows.

1. An intruder is watching on the connection between the NIS client and the NIS server for the NIS/RPC/UDP `yp_match` query for a particular user. This command is used to get authentication information from a NIS server. The reply is a string identical to a user entry in the `/etc/passwd` file.
2. When this query passes by the intruder it quickly generates a fake reply and sends this to the client before the server sends its reply. Since UDP is used and the servers reply is later than the intruders the latter message is just discarded.

The result of this attack is that the user is authenticated by the intruder and not by the proper NIS server.

4.1.3.3 Comparison

These two attacks succeed because of misplaced trust. In both cases the client trusts the server blindly and because of that it can be fooled. The results differs slightly. In the Windows NT case the intruder gets access to the server and only as the user that s/he manages to intercept. In the UNIX case on the other hand the intruder gets access to the client but as a user of his/her own choice. One could, however, easily think of variants were the tables are turned in both cases.

4.2 Active Misuse of Resources (NP6)

4.2.1 Resource Exhaustion

Resource exhaustion is usually employed as a mean of causing denial of service. The idea is to allocate as many instances of one (or more) type(s) of resources as possible. Normally this will either slow down or crash the system.

4.2.1.1 Resource Exhaustion in Windows NT

Normally, a thread in Windows 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, Windows NT has an aging mechanism but will only age threads up to priority 14. *CpuHog* is a small program, which uses the priority mechanism to hang the system. What *CpuHog* actually does is to set priority 15 on itself and then enters an infinite while loop. This will cause the system 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 addressed this problem in a service pack by allowing aging up to priority level 15 which means that *CpuHog* will only slow down the system considerably.

4.2.1.2 Resource Exhaustion in UNIX

Probably the most known denial of service attack in the UNIX environment is the `while (1) fork();` program. This line of C code starts a new process for every iteration in the while loop. The result will be that all entries in the system wide process table are consumed, which implies that no new processes can start. Many vendors have nowadays fixed this problem by limiting the number of processes a user can start.

4.2.1.3 Comparison

In both systems resource exhaustion attacks are possible, due to the fact that they allow users to allocate an unreasonable number of resources. It is interesting to note that the two attacks uses different mechanisms, but that the result is the same.

4.3 Passive Misuse of Resources (NP7)

Passive misuse of resources is the idea of an unauthorized user looking for weaknesses without using them. This knowledge can later be used in an active attack. The methods used are either manual browsing or automated browsing using specialized tools.

4.3.1 Manual Browsing

In manual browsing the attacker looks for weaknesses without using any tool designed for this purpose. The knowledge gained by manual browsing can later be incorporated into an automatic tool. It is difficult to say whether one of the two systems is more susceptible to manual browsing attacks than the other. The key thing here is a good understanding of the system i.e. to know what to look for. Today it is easier to find in-depth publicly available descriptions of the UNIX systems.

4.3.2 Automated Searching

Automated searching is subdivided into using a personal tool or using a publicly available tool. The only reason for this division is that it is easier to automatically detect the use of a publicly available tool.

A lot of different tools that looks for weaknesses or other information exists for both environments. In some cases these tools have the same name and looks for general as well as operating system specific vulnerabilities. A program in this category is the Internet Scanner (IS), which is a commercial version of the well known program Internet Security Scanner (ISS). Other general programs that have for a long time been used in the UNIX community are either going to be ported or are now in the process of being ported to Windows NT. SATAN is an example of such a program. All in all, both systems are definitely sensitive to these kinds of attacks.

5. Discussion and Conclusion

This paper demonstrates that the security mechanisms of Windows NT are slightly better than those of UNIX. Despite this fact the two systems display a similar set of vulnerabilities. This implies that Windows NT has the theoretical capacity of being more secure than “standard” UNIX. However, with the present way of installing and using the systems there seems to be no significant difference between their security level. 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 should hold when comparing the systems at the same state of development and market penetration. Thus, the only reason for more UNIX penetrations is that the system is older and more well-known and we should anticipate an increasing number of intrusions into Windows NT, a tendency that has already started.

It is clear that the Achilles heel of both systems is networking. Since both systems utilize the same low level protocols, i.e. IP, TCP and UDP, and comparable high level protocols. This could to some extent explain that the security behavior of both systems is similar, but it does not provide the full explanation. However, as long as the networking is such a weak point, the usefulness of other security mechanisms is diminished.

Acknowledgments

The work described in this paper was partly supported by Telia Research, Farsta, Sweden.

References

[1] Maurice J. Bach, The Design of The UNIX Operating System. Prentice-Hall Inc, 1986.

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

[3] Sarah Brocklehurst, Bev Littlewood, Tomas Olovsson and Erland Jonsson, On Measurement of Operational Security. In COMPASS 94, 9th Annual IEEE Conference on Computer Assurance, Gaithersburg, pp.257-66, IEEE Computer Society, 1994.

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

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

[6] Simson Garfinkel and Gene Spafford, Practical UNIX and Internet Security, 2nd edition, O'Reilly & Associates, Inc., 1996.

[7] Hans Hedbom, Stefan Lindskog, Stefan Axelsson and Erland Jonsson, Analysis of the Security of Windows NT. Chalmers University of Technology, June, 1998.

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

[9] David K. Hess, David R. Safford and Udo W. Pooch, A UNIX Network Protocol Security Study: Network Information Service. Texas A&M University.

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

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

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

[13] LeFebvre-W, Simply syslog. Unix-Review, vol. 15, no. 12, November 1997.

[14] Ulf Lindqvist, Ulf Gustafson and Erland Jonsson, Analysis of Selected Computer Security Intrusions: In Search of the Vulnerability. Technical Report 275, Department of Computer Engineering, Chalmers University of Technology, Göteborg,

Sweden, 1996.

[15] Ulf Lindqvist and Erland Jonsson, How to Systematically Classify Computer Security Intrusions. In Proceedings of the 1997 IEEE Symposium on Security & Privacy, pp. 154-163, Oakland, California, USA, May, 1997.

[16] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels and John S. Quarterman, The Design and Implementation of the 4.4BSD Operating System. Addison-Wesley, 1996.

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

[18] 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.

[19] 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.

[20] Tomas Olovsson, Erland Jonsson, Sarah Brocklehurst and Bev Littlewood, Towards Operational Measures of Computer Security: Experimentation and Modelling. In Predictably Dependable Computing Systems, editor B. Randell et al., Springer Verlag, 1995.

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

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

[23] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh and Bob Lyon, Design and Implementation of the Sun Network Filesystem. Summer USENIX Conference Proceedings, Portland, 1985.

[24] Jennifer G. Steiner, Clifford Neumann and Jeffery I. Schiller, Kerberos: An Authentication Service for Open Network Systems. USENIX Winter Conference, Dallas, Texas, USA, February, 1988.

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

[26] Tatu Ylönen, SSH - Secure Login Connections over the Internet. SSH Communications Security Ltd, June 7, 1996.