GIAC Certified Windows Security Administrator (GCWN)
Practical Assignment
Version 3.2, Option 2

Does Windows 2000 security model get along with my Linux?

Linux-Windows interoperability

Jorge D. Ortiz-Fuentes

December 7, 2003

Abstract

We are living in an heterogeneous world —even Microsoft claims that they don't have the monopoly of the desktop market— and with the fast growth of Linux systems in the enterprise environments almost everybody cares about inter-operation of Linux and Windows 2000.

Most of the security features of Windows 2000 domains and the more recent Windows 2003 are designed around the Active Directory. To benefit from this model Linux systems should be able to inter-operate with the Active Directory. The aim of this paper will be to explore which of the main aspects of security related to the Active Directory (at least in the Windows platform) can be successfully used from a Linux environment. Some of this aspects have already been covered by other authors[10], and some others remained, to the best of my knowledge, unexplored.

Obviously, not all the functionality of the Windows 2000 security model inter-operates well with Linux. Regarding the five aspects that from my point of view are most relevant —Active Directory authentication, file sharing, Dynamic DNS updates, PKI and IPSec— I have written an explanation of how they work, what is needed for inter-operation, and which are the available solutions, if any. For the ones that I could get them to work with Linux, I give some hints about the implementation, and for those that I couldn't get them to work, I have tried to explain why they won't work (commenting on whether this is due to the fact that there is a lack of software or there is an additional problem, like for example, a legacy protocol). Some other aspects have remained uncommented here because the information available about them was insufficient or even inexistent.

All the tests will be performed with Windows 2000 Server and Red Hat Enterprise Linux WS 3.0. However, most of the results can be safely extrapolated to the current versions of other distributions.

Contents

1 Introduction

Windows 2000 and its Active Directory are present in many enterprise environments, and as Linux systems get more relevant roles in these environments, more and more people want to know how to integrate them. This paper will talk about inter-operation of the main security features of Windows 2000 when working with Linux systems.

Also, I decided to study Linux and Windows 2000 inter-operation instead of writing a pure Windows 2000 paper, because I think that by having to make these two different operating systems work together, I get a better understanding of what happens under the hood in the Windows security model.

This paper explains things related with Microsoft's Active Directory, the Lightweight Directory Access Protocol (LDAP), Kerberos 5, the Domain Naming Service (DNS), Public Key Infrastructures (PKI), and the IP Security Protocol (IPSec) and some others. However, I haven't tried to explain how they work because there are very good introductory papers and books that do that. I have included the references to these papers and books so the reader that isn't so familiar with any of them can still benefit from the contents of this paper and enjoy the others. Also, a good knowledge on system administration tasks, both in Windows 2000 and Red Hat Enterprise Linux WS 3.0, is required.

I haven't tried to write a "*do this, do that, and click here*" guide to implement the inter-operation. I have preferred instead to explain the way the things work, the things needed for the integration, and the possible (and available) solutions and their security implications. I hope that you too consider this more helpful.

1.1 Acknowledgments

The unquestionable help that Rosa and Lidia have provided me with, through their patience and unconditional support, has been the decisive factor for writing this paper.

I also thank Raul and David. Working together as a group is making us all improve and extend our knowledge about security.

Finally, I sincerely thank all the people that have put their knowledge and work in developing the free tools that I use every day and that are both an important part of what I explain in this paper and the applications used for writing it.

2 Integrating Linux with the Windows security model

The Windows security model covers every aspect of the operating system: authentication, access control to the system objects, auditing, secure communication, usage policies, and many others. Some of them refer to each system, the majority apply to all the computers of the domain, and some others cross the domain boundaries.

I have chosen to study the inter-operation of five of those security aspects with Linux systems: user authentication, dynamic DNS updates, file and printer sharing, PKI, and IPSec.

2.1 User authentication

Directory services have been long used for distributed computing environments, mainly as a way to authenticate users and deal with computers. Some examples of directory services are Sun's Network Information Service (NIS), also known as Yellow Pages, Sun's NIS+, Novell Directory Services (NDS) and, X.500 although it is a more ambitious one and has not been used, to the best of my knowledge, for authentication.

With Active Directory Microsoft tried to take a step beyond in directory services. They designed the Active Directory to act as the centralized configuration of the network using standard protocols and keeping security in mind. Users, machines, applications and other objects keep their configurations as attributes stored in the Active Directory, which becomes an essential component of the Windows 2000 domain architecture.

When the Active Directory is used to act as the network user and password database, Windows clients use the Kerberos protocol to validate the user or machine account[1]. The Kerberos KDC consults the active directory directly and obtains the authorization of the account to use the network services and its access level, expressed as a set of Security Identifiers (SIDs). After that, the WinLogon process can query the LDAP to obtain her attributes, such as the Group Policy Objects (GPOs) that may apply[2]. I will not explain how the Kerberos and LDAP protocols work in the Windows platform, since other documents[3][1][4] have already done so very well. But I will comment on the details related to the problem I would like to solve here: Linux-Windows interoperability.

Although Kerberos was designed with interoperability in mind[5], when Microsoft included the protocol as the preferred one to perform authentication in a Windows 2000 domain, they also made some modifications. These modifications were directed to include the authorization data of the user —his SID and the SIDs of the groups he belongs to— in the application-specific data field of the ticket. Doing this, they got the side effect of disallowing the use of a KDC other than a Windows based one to be used to authenticate the users for any windows domain member if a real domain integration is required. This is the same reason that restricts Kerberos credential delegation[6] for Windows services to Windows systems only (i.e. both the system with the original ticket and the system that trusts the first one.)

In this section I will show how to use a Linux machine as client of the Active Directory. By being able to do so, we can centralize user administration and make our lives easier. As I said above, the opposite —Windows 2000 domain members authenticating to a Linux KDC— could be implemented only with some loss of functionality, since the SIDs would not be included in the returned ticket.

This is a basic step in the Linux-Windows interoperability road. Some people[7][10] have already published papers on this matter, but this will try to be a comprehensive description of the solution.

2.1.1 Description

In this section I will explain how a windows client authenticates a user to the Active Directory and what pieces are needed to get this capability from Linux.

The authentication of a user in a Windows 2000 client to a domain using Kerberos consists of the following basic steps[1]:

1. Get a Kerberos Ticket Grating Ticket (TGT) for the user through the Kerberos authentication service (AS).
2. Get a ticket for the host the user is logging on using the Kerberos ticket granting service (TGS) and the previously obtained TGT.
3. Make a query to the LDAP server to obtain all the user attributes, such as GPOs that apply.

This behavior can be observed in the `tcpdump` trace of figure 1

```
14:33:14.282223 192.168.10.11.1056 > 192.168.10.2.kerberos: v5
14:33:14.286199 192.168.10.2.kerberos > 192.168.10.11.1056: v5
14:33:14.291537 192.168.10.11.1057 > 192.168.10.2.kerberos:
14:33:14.293588 192.168.10.2.kerberos > 192.168.10.11.1057:
14:33:14.616205 192.168.10.11.1058 > 192.168.10.2.ldap: udp 207
14:33:14.618620 192.168.10.2.ldap > 192.168.10.11.1058: udp 185
```

Figure 1: `tcpdump` trace of the user logon process

Linux will use Kerberos to obtain a TGT, which confirms the identity of the user, and a ticket for the host the user is login to, which is this same Linux system. After that, it will use LDAP to download the attributes of the account. These attributes will include the same data contained in the user line of the `/etc/passwd` in a *traditional*,

[1]Other steps are taken, which I won't mention for the sake of clarity, since they relate to other tasks different from authentication itself.

so to say, system (home directory, shell. . .), as well as data from other files, like the additional user groups. However, other attributes, like the GPOs that are assigned to the user, will not be used.

In the Linux/UNIX platforms a simple mechanism was invented to be able to change the authentication method and even combine some of them easily. The idea is quite simple, and thus a beautiful one. Instead of adding every authentication method to every program and recompiling it in case a new method is required, authentication methods are implemented as modules (Pluggable Authentication Modules, i.e. PAM) in dynamically loadable libraries which are loaded at run time and the desired method for each service is selected in a configuration file. This mechanism has made easier to integrate UNIX systems into different authentication paradigms available for distributed environments, like NIS, NIS+, LDAP, and of course Kerberos, among others.

The designers of PAM identified four different and independent groups of authentication tasks, as explained in the Linux-PAM documentation[8]:

account management enables the program to verify things like access or validity of a password.

authentication management, as its name indicates, it establishes whether the user is the same who claims to be.

password management takes care of updating the information required for the authentication.

session management performs any of the maintenance tasks before the user has access to the service and right after finishing using it.

2.1.2 Integration

In order to carry out integration successfully, it is important to understand what the dependencies are.

Kerberos uses two pieces of data: user name and password. The user name will be mapped to its UID using whatever is configured in the nsswitch.conf to query. Consequently, LDAP should already be working and configured or else, a local account, i.e. in /etc/passwd, where every user name must exist. The UID is needed, among other things, to cache the tickets so only the right user can read them and, most important, to set the UID of the service to run as the right user.

Also, Kerberos authentication validates the timestamps of every ticket request. Thus, time should be synchronized before trying to use Kerberos. Additionally, it helps to troubleshoot problems using logs written in different machines.

The order to follow when configuring the systems to implement Linux authentication to the Active Directory are:

1. Time synchronization.
2. LDAP queries.
3. Kerberos authentication.

Time Synchronization All the systems that are going to authenticate their users in the Active Directory must synchronize their times to the domain controller. This is very important because Kerberos checks that the timestamp included in each ticket request isn't older than, by default, 5 minutes compared to the clock of the KDC.

Windows 2000 `Win32Time` service is used to do time synchronization for the Windows hosts, but, by default, does not allow other systems to use it as an NTP source attending UDP port 123.

I have found information[9] indicating that this capability can be enabled by setting to 1 the value name `LocalNTP` that can be found under
`[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters]`
to 1 (one) and restart the `Win32Time` service. However, I have been unable to make Red Hat Enterprise Linux WS 3.0 synchronize to it.

Instead, I did synchronize both the Windows 2000 domain controller and the Red Hat Enterprise Linux WS 3.0 to the same external time source. This capability can be configured for a Windows 2000 system by issuing the command `NET TIME /SETSNTP: ntpsource`, that writes the data to the corresponding registry key. The contents of the required `ntp.conf` file for the Red Hat Enterprise Linux WS 3.0 are:

```
drift /var/lib/ntp/drift

restrict ntpsource mask 255.255.255.255 nomodify
server ntpsource

restrict 127.0.0.1
restrict default ignore
```

For both systems *ntpsource* has to be changed to whatever the NTP source is.

LDAP Queries To be able to authenticate against the LDAP server the schema must be modified. This is a very important and irreversible change of the Active Directory schema. At least three alternative solutions exist:

- Modify the schema manually as indicated by the RFC 2307. Windows 2000 provides a way to enable a new snap-in —running the command `regsvr32 schmmgmt.dll`— to administer the Active Directory. From there the schema can be edited by hand or, wiser, using a LDIF file (downloaded or your own). *Do not ever try this in a production environment.*

Adding data to the newly defined attributes must be done manually as well.

I consider this a complicated solution. Good to experiment with, but a bad one for the production systems.

- Use AD4UNIX which modifies the schema and adds a plug-in for user administration. Unfortunately enough, the domain that hosted the AD4UNIX package has very recently been freed and the page is not hosted anywhere else that I could find. The package is still available from many ftp sites, though.

  AD4UNIX provides a DLL that integrates itself with the management console and lets the administrator to add the info for the additional attributes when configuring each of the accounts.

  AD4UNIX provides a friendly interface, but it is still an unsupported product.

- Use the Microsoft Services For UNIX (SFU). This is probably the best for a production environment since it is supported by Microsoft and they take care of providing their users with the required patches when new vulnerabilities are found. It also requires an additional investment in software.

  Of course, Microsoft's product offers integration with the management console also.

  If this is the solution chosen for implementation, the NIS service must be stopped —the two necessary SFU to do this are the modifications to the schema and the extra functionality of the **AD Users and Computers** snap-in— with the command `"NET STOP NisSvc"` and disable it from the services snap-in of the Management Console.

Windows 2000 systems query the Active Directory both directly from the KDC service, implemented in the `kdcsvc.dll` that invokes `Ntdsa.dll`, and by using a Kerberos ticket that allows access to the Active Directory through the LDAP interface. On the Linux side, the PAM module for Kerberos authentication needs to map the user name to a UID by querying the Active Directory. But, the user has not been authenticated yet and, by default, Active Directory does not allow anonymous queries. So a solution is needed that allows the PAM module to do a LDAP query to the Active Directory. The following three alternatives can be used to overcome this limitation:

- Use the administrator password in the `ldap.conf` file. Although this solution would work, it would mean that each Linux system will have a world readable file containing the password of the Active Directory administrator. Clearly a very bad idea.

- Allow anonymous queries to the Active Directory. While this solution would not expose to unnecessary risks any administrative password, it would allow any user/system with access to TCP port 389 (LDAP) to enumerate the domain. It would be giving away lots of information with zero effort for any potential intruder. This is also a bad idea.

- Create a restricted user with permission to query the Active Directory. This solution implies to have the password of the user in the same world readable file. However, the access of this user can be tightly restricted making an overall lower risk. This is explained in detail in Poulin's paper[10]

After all these steps, the new schema is ready. Now, the attributes that have been added to the schema must contain valid data for each user that we would allow to log on from the Linux systems. The same applies to the groups being used for those users.

There are things that work differently to what Linux administrators are used to. The most important one is User Private Groups (UPG).

Most modern Linux distributions use User Private Groups. A UPG is a group that includes just one user as a member and that has the same name than that user. Usually its GID is the same number as the UID of the user. The user has his UPG set as his primary group. It makes easier to group users together —by making them belong to a group for each task they are involved into and using a umask that allows him to share his work with the other users of the group— while keeping his private data protected —the owner group of the data is his UPG so the umask only needs to take care of the *others* permissions. Windows 2000 doesn't use this mechanism because it has more granularity for access control with the use of Access Control Lists (ACLs).

Active Directory does not allow to create a group with the same name as an existing user. Thus a different approach for naming UPGs must be taken. A possibility is to prefix or postfix the user name with `upg`. The output of the `ls` command will look different, but it will do the trick.

With the right data written to the new attributes of the Active Directory, the next step is to be able to access the data from the Linux system. The Active Directory can be manually queried to check that the information about the users can be recovered. This is done by using the `ldapsearch` utility that is included with Red Hat Enterprise Linux WS 3.0, and the following queries to retrieve the information:

The first one is to recover all the attributes of a user whose UID is 10000 and the second one does the same thing for a group. Obviously, `ldapproxy` should be replaced by the previously configured LDAP proxy user, `myrealm` and `net` by the realm, `s3cr3t_p4sswd` by the password of the LDAP proxy user, and the UID/GID for the user to test. If anything fails, remember Ethereal[11] is your friend. Ethereal

```
ldapsearch -x -D "ldapproxy@myrealm.net" -w s3cr3t_p4sswd -b \
"cn=Users,dc=myrealm,dc=net" \
"(&(objectclass=User)(msSFU30UidNumber=10000))"

ldapsearch -x -D "ldapproxy@myrealm.net" -w s3cr3t_p4sswd -b \
"cn=Users,dc=myrealm,dc=net" \
"(&(objectclass=Group)(msSFU30GidNumber=10000))"
```

Figure 2: LDAP test queries

is a network sniffer that allows to capture and analyze network traffic —particularly LDAP— helping to reduce the troubleshooting time.

Once the right attributes are set in the Active Directory and it has been checked that they can be queried, the settings in `nsswitch.conf` will be used to decide where to get the information about users and groups from. This is used, for example, when `ls` has to print the user and group that own each file. The configuration must contain this three lines:

```
passwd:  files ldap
shadow:  files ldap
group:   files ldap
```

You can edit the file manually or use Red Hat's configuration command: `authconfig`.

If you try to use the nss_ldap package that comes with Red Hat Enterprise Linux WS 3.0 (`nss_ldap-207-2.rpm`) you will notice that it is compiled with schema mapping disabled. This feature enables the system administrator to configure the name of the attributes that will be queried (instead of the ones specified in the RFC2307[12]) and is required if you decided to use Microsoft's SFU since they use their own naming system. To add this capability to package, it has to be recompiled from its source adding the parameter `--enable-schema-mapping` to the line that contains the `configure` command in the `spec` file.

With the schema mapping capability enabled in nss_ldap, its LDAP queries will honor the mapping contained in the `ldap.conf` file. The following lines define the right mapping that is needed for the authentication process.

```
nss_base_passwd      cn=Users,dc=myrealm,dc=net?one
nss_base_shadow      cn=Users,dc=myrealm,dc=net?one
nss_map_objectclass  posixAccount   User
nss_map_attribute    uidNumber      msSFU30UidNumber
nss_map_attribute    uid            msSFU30Name
nss_map_attribute    gidNumber      msSFU30GidNumber
nss_map_attribute    uniqueMember   posixMember
nss_map_attribute    userPassword   msSFU30Password
nss_map_attribute    homeDirectory  msSFU30HomeDirectory
nss_map_attribute    loginShell     msSFU30LoginShell
nss_map_attribute    gecos          msSFU30Gecos
nss_map_objectclass  posixGroup     Group
nss_map_attribute    gid            msSFU30Name
nss_map_attribute    memberUid      msSFU30MemberUid
```

If this is planned for real usage, not only testing, performance should also be taken into account. The `getXXent()` functions controlled by the settings in `nsswitch.conf` can be called very often. Let me put an example. If there is a directory with one hundred files, for each of the files two queries to the Active Directory are done —one for the user and another one for the group. There is a solution to this problem already implemented in Red Hat Enterprise Linux WS 3.0: the name services cache daemon (`nscd`). This daemon can be enabled using `authconfig` also.

Everything should work by now, but secure communication aren't already in place. Active Directory by default is only attending LDAP in clear text, but it can easily configured to also accept LDAPS, as M. Poulin explains in his paper[10]. This clearly makes sense because:

- The DES hash of the keys of the users can be obtained by sniffing from the LDAP traffic —they are stored in `msSFU30Password`.
- The proxy user to query the LDAP can be compromised. Although it can be argued that its key is in clear in a world readable file, it is important to note that to read this file is necessary to have access to one of the Linux workstations.

Kerberos Authentication The users of the Active Directory that are shared with the Linux systems will validate their identities using the Kerberos protocol. Kerberos is based on symmetric cryptography, which means that every user shares a secret key with the KDC —the Active Directory domain controller— and so it does every host that offers services.

When the user tries to logon the Active Directory domain, he types the password and the KDC can retrieve from the Active Directory its copy of the password. For the services, it is somehow different. In Linux, their passwords are stored locally in a file called `keytab`. The `keytab` file can be created following the steps described by M. Poulin in his paper[10].

The task of getting the tickets from the KDC is performed by the PAM module pam_krb5, but this module can not work alone. Since Kerberos authentication uses only two data —the user name and the password— the module needs help to somehow check whether the account exists and to get its corresponding UID. This help comes from the name service that maps user names and UIDs and gets other related information, and that was configured in the previous section (page 8) to use the LDAP service offered by the Active Directory.

The easiest way to configure the use of pam_krb5 for authentication is to use the `authconfig` command, fill in the data, and accept.

With all this settings the user is able to login into the Linux system using his Active Directory user. He can even change his password —using the `kpasswd` command— from his Linux workstation. But be aware that there is a bug here. If a user changes his password, either from a Linux or a Windows system, the object of the Active Directory that describes his account changes the value of the attribute `msSFU30Password` from the default value (`ABCD!efgh12345$67890`) to the DES hash of the new password. Due to an unknown reason to me, if the value of this attribute is a DES hash, `pam_krb5` fails to cache the credentials and klist will complain:

```
$ klist -5
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_10000)
```

This problem has two possible workarounds, none of them completely satisfactory:

- The user can execute `kinit`, type his password again and get his credentials cached. This means typing the password twice.

- The administrator can reset the value of the attribute to the default one. This can be done from Linux using the `ldapmodify` command as follows:

  ```
  ldapmodify -x -D "Administrator@myrealm.net" -w s3cr3t_p4sswd \
   -f mod_pass
  ```

  where `s3cr3t_p4sswd` must be changed for the administrator password, `myrealm.net` for your Kerberos realm, and the contents of the `mod_pass` file are:

  ```
  dn: cn=No Cred User,cn=Users,dc=myrealm,dc=net
  changetype: modify
  replace: msSFU30Password
  msSFU30Password: ABCD!efgh12345$67890
  ```

  where the common name *No Cred User* has to be the name of the user that has changed his password.

  Doing this, the authentication is done against the *new* password, the one that user chose when using `kpasswd`. This is because the attribute used by Kerberos when authenticating a user is a different one.

2.2 Dynamic DNS updates

Windows 2000 has moved away from NetBIOS to work with TCP/IP —although it still implements NetBIOS over TCP/IP— and has a very much closer relationship with DNS (Domain Name Services) than the preceding versions of Windows.

2.2.1 Description

In this section I will explain the Dynamic DNS (DDNS) mechanism used to get the Active Directory to update a DNS entry as well as the security considerations[13].

One advantage of Windows 2000 implementation of DNS is the ability to make Dynamic DNS updates in a secure fashion. Once a system gets an IP address via Dynamic Host Configuration Protocol (DHCP), the DHCP server updates the DNS resource records (RRs) —stored in the Active Directory— to include the direct (A record) and reverse (PTR record) resolution of this address. This is, of course, a great idea to deal with users of laptops and other portable computers in your domain and has been successfully extended to office desktops to reduce deployment time.

This need for dynamically updating the contents of the DNS has been around for some time now. The developers of BIND —the Berkeley Internet Name Domain,— which is considered by many as the reference implementation of DNS, started to include different functionalities through the different releases of BIND 8 that were necessary to implement this feature in a secure way.

The need of security is quite clear. Dynamic updates allow to (un)bind a name of the domain with an IP address. This update must be from an authenticated user with authorization to do it. However, DNS is a non authenticated protocol, so anybody can query the DNS server as long as the IP is allowed to do so —and even forging a UDP packet with a spoofed IP source address is a trivial task.

The Internet Engineering Task Force (IETF) proposed an extension to the DNS protocol, known as DNSSEC[14], that implemented data integrity and authentication based on public cryptography. These extensions were implemented in BIND, but didn't get the wide adoption expected, because of the computational resources required by the public cryptography algorithms and the latency associated to each query.

They soon offered an alternative solution to the authentication problem with the introduction of Transaction Signatures (TSIG)[15]. The basic idea behind TSIG is to create a query packet with a timestamp and obtain the cryptographic hash of the packet plus a shared secret key, and then send the packet and the hash —no key, obviously— to the server, that can generate the hash with its copy of the shared key and check if the query comes from a trusted user.

Microsoft implemented the same mechanism in their operating systems Windows 2000 and later, but they choose a different kind of signatures instead of the ones proposed in the TSIG RFC, which were HMAC-MD5. They implemented GSS-TSIG instead[2], which was proposed as a draft at that moment and has just become a RFC standard[16]. This standard uses the same ideas of the RFC2845 about TSIG, extending it with a new algorithm based Generic Security Service API (GSSAPI), which, as its name implies, provides security services in a generic way.

The main problem to implement the original TSIG mechanism is the distribution of the secret keys that must be known only by both the client and the server. In the Windows domain model, a computer that belongs to a domain, shares the key of the computer account with the KDC. This key belongs to the computer account; the users have no access to this key. The key is established when a computer joins the domain and is changed periodically through a secure channel. Every time a Windows 2000 system is started it obtains a Kerberos ticket for the computer. GSS-TSIG uses information from this Kerberos ticket as the shared secret key to sign the transaction.

2.2.2 Integration

This restriction introduced by the use of GSS-TSIG applies at least to the DHCP server that should take care of updating the RRs —at least the PTR— in the DNS server[17]. If the DHCP server is configured so it it does not enable updates for DHCP clients that do not support Microsoft's DDNS —the default configuration— it also affects to those clients, namely the Linux ones. If this option is enabled, the only requirement on the DHCP client side is that it must be able to send the fully qualified domain name (FQDN) of the system encoded in the DHCP as option 81.

Red Hat Enterprise Linux WS 3.0 comes with `dhclient` that is the client part of the freely distributable reference implementation provided by the Internet Software Consortium (ISC)[18]. `dhclient` is able to send the FQDN encoded in the option 81. The required statements are:

```
send fqdn.server-update true;
send fqdn.fqdn "system.domain.net";
```

and must be included in the configuration file `/etc/dhclient-`*`if`*`.conf`, where *if* is the interface name —usually `eth0`— and `system.domain.net` must be replaced by the FQDN of the Linux system.

As for the server side, the ISC also distributes in their free reference implementation a DHCP server. ISC's DHCP server implements DDNS updates based on TSIG. GSS-TSIG based updates aren't supported yet, however, all the parts seem to be separately available. Red Hat Enterprise Linux WS 3.0 comes with this software bundled in:

[2]In perfect harmony with Microsoft's old *Embrace & Extend* strategy.

dhcp-server is the program that implements the DHCP server.

libgssapi_krb5 is the dynamic library that implements the GSSAPI. It is part of the Kerberos 5 distribution.

pam_krb5 is the PAM module required to use Kerberos for authentication purposes.

If you need this feature working now and the IP addresses assigned to the Linux systems being updated through DDNS, you can either allow unauthenticated updates, or implement your DHCP server in a Windows 2000 system.

The only sensible way to allow unauthenticated updates is in a different DNS zone than the one used for your servers and restricted to systems that don't depend on their name being resolved correctly by the DNS server. You should also keep in mind that you won't be able to trust the logs that write the name of those systems obtained with reverse resolution.

If your want to serve DHCP from a Windows 2000 system it must be a different one than the system used as Active Directory domain controller. You should configure the DNS properties of the DHCP server to update the client information in DNS and do it always, and to enable updates for clients that do not support dynamic updates. The DHCP clients must still be configured to send the FQDN.

Using the second alternative —DHCP server updating the DNS— has one advantage. Because the DNS fully integrated in the Windows 2000 security model the DNS administrator has the ability to set discretionary access controls to each of the resource records. This model provides more granularity to control the read and write access to the DNS, but in some circumstances it might be more troublesome than beneficial and make administration complicated in excess.

Microsoft's suggests[17] to use the built-in group `DnsUpdateProxy` and make the DDNS updates with this group from the DHCP server. This means that the resource records that have been created by this group can be overwritten by other authorized users that will then become the owner of the resource record. They also warn their users that using the `DnsUpdateProxy` means that you can not use the same system as domain controller and DCHP server, because any DNS names registered by this system that belongs to the group are non secure.

2.3 File and printer sharing

Perhaps the most commonly used networking feature of Windows systems is the capability to share files and printers. The feature appeared with the Windows for Workgroups release and became widely accepted among the Windows users. It rooted so deeply rooted in the hearts and minds of the Windows user community that has been proclaimed as a de facto standard and other operating system vendors offer software that can communicate with Windows shares.

If your computer isn't in my network neighborhood it doesn't exist.

2.3.1 Description

Server Message Block (SMB) is a network information sharing protocol. It allows the users to share files, printers and other information. It was designed to run over NetBIOS, that was a transport protocol —as opposed to an application protocol as SMB.

Windows versions prior to Windows 2000 required support of NetBIOS. When Microsoft realized of the success of TCP/IP and the inherent limitations of NetBIOS they started to evolve SMB to run over TCP/IP. They first added the capability to run NetBIOS over TCP/IP, which was called NBT, and later they modified SMB so it could run directly over TCP/IP and renamed the SMB protocol as Common Internet File System (CIFS).

2.3.2 Integration

The wide usage of SMB/CIFS incited many people to reverse engineer the SMB protocol and CIFS later and all of their dialects to be able to make the Linux/UNIX and the Windows worlds communicate. The most successful implementation of SMB/CIFS for Linux is Samba and they have just released their long awaited version 3, which integrates better in an Active Directory domain.

Samba has several features that are related to security and inter-operation between Linux and Windows. The most important ones are:

- Computers can be integrated with an Active Directory domain as a member server. Computer accounts can be automatically created following a process which is very similar to the one used for native Windows clients. It also implements the required mechanisms to deal with this account —like the establishment of secure channels through RPC.

  This is a three step process[19]:

  1. Edit the smb.conf file and check/modify the following options, so they contain the right values:

     ```
     realm = MYREALM.NET
     security = ADS
     encrypt passwords = yes
     password server = dc.myrealm.net
     ```

     where `MYREALM.NET` must be replaced by the realm of the domain, and `dc.myrealm.net` by the fully qualified domain name of the domain controller.

2. Edit the Kerberos configuration `krb5.conf`. If your computer is able to authenticate Active Directory users already, you should be OK.
3. Make the computer join the domain using the administrator's account:

   ```
   net ads join -U Administrator%s3cr3t_p4sswd
   ```

   This step will create a computer account —the ones used to authenticate the computers and make sure that they belong to the domain— in your Active Directory domain. You can browse the properties of this account in the Computers folder of the **Active Directory Users and Accounts** snap-in.

- Maybe the most important feature of the newest version of Samba is that it can join to an Active Directory using LDAP and Kerberos authentication. Not only the computer account previously commented, but every access will be authorized or denied based on its Kerberos ticket.
- Samba clients can independently be configured to use the Kerberos credentials acquired as explained in Sec. 2.1 to get access to objects (files, printers, ...) available in other computers of the domain.

  I couldn't make this feature to work properly with the version of `smbclient` shipped with Red Hat Enterprise Linux WS 3.0. The TGT was correctly used to get a ticket for the system that hold the share that it was trying to connect to. It seems to me that this release (`samba-client-3.0.0-14.3E.rpm`) has a bug, since user/password authentication works fine. I have tried every possibility —SMB signing, SPNEGO, Unicode...— in the `smb.conf` to modify the behavior of the client, but couldn't get it to run successfully with the Kerberos option.
- The capability to map the Linux permissions of each of the files into the access control list that gets displayed in their security tabs.

2.4 PKI

A *Public Key Infrastructure* is set of elements and services that are need be able to use digital certificates. Windows 2000 has many of these elements and services available to deploy a PKI. I will talk about how well can it be used in heterogeneous environments.

2.4.1 Description

In this section I will explain which of the certificate models represented by the certificate templates can be used in Linux systems working together with Windows

2000.

An excellent introduction to PKI can be found in chapter 15 of Bruce Schneier's "Secrets & Lies"[20], but let me state the main assumptions behind any PKI:

- Asymmetric encryption means that a message encrypted with a public key can only be decrypted using its corresponding private key. If it has been encrypted using the private key, it can only be decrypted with the public key.
- It is unfeasible to obtain the private key from the public key.
- The only one who has access to the private key of a user is himself.
- The certification authority is to be trusted when it certifies that a user, i.e. it is dutiful for all aspects of the certification process.

PKIs provide the means for strong authentication and encryption. Some of the most common uses of certificates that make sense also in the Linux platform are:

Email signature and encryption Email messages differ from their non-electronic counterparts in that they can be easily forged or modified without the recipient being able to realize of the problem. Also, there is a lack of privacy, since they are transmitted in clear through public communication channels and untrusted machines.

The utilization of email in a corporate environment must be accompanied with the appropriate mechanisms to ensure security.

The signature of an email —or any other file— is done calculating its cryptographic hash and encrypting it with the user's private key. Anybody will be able to verify that the mail has been sent by the author and that it hasn't been modified using his public key, decrypting the hash and comparing it with one calculated independently.

The process for maintaining the privacy in a secure channel is similar. The public of the recipient is used to encrypt the message —more accurately a session key. Thus, only the intended recipient —the only one who is supposed to have the private key that matches with this key— can decrypt the message.

User authentication A user can be authenticated by encrypting an authentication challenge or an authenticator with his private key. The private key can be stored in the computer —cryptographically protected— or, even better in a device especially designed for this purpose (smart card or token).

Encrypted filesystem Private data should be protected even from users with privileged access —legitimate or not— in the system. Files can be encrypted using pseudo-random session keys and these keys can be stored encrypted with the public key of the individuals that have authorization to decrypt the file —if it is only the user and the key is lost, there is no way to recover the contents of the files.

2.4.2 Integration

The first step to inter-operate with Windows 2000 PKI is to get the certificates in the Linux systems. Although, certificates can be requested through a web page (`http://dc.realm.net/certsrv/`), the web site is configured to support Integrated Windows Authentication and uses an ActiveX control to calculate the key pair and make the request.

Kerberos authentication can be used with Mozilla —the web browser that comes with Red Hat Enterprise Linux WS 3.0,— but comes as a plug-in[21] that is installed separately. Mozilla has built in functionality to generate key pairs and certificate requests —in SPKAC format. Unfortunately the requests are not compatible with the ones used by Windows 2000 CA —that are PKCS#10— obtained in Windows 2000 by running the ActiveX control. This means that certificates can't be requested from Linux.

The certificates can be read from the active directory using LDAP. They are stored as an attribute of the user and computer objects and can be obtained with the following queries:

```
ldapsearch -x -D "ldapproxy@myrealm.net" -w s3cr3t_p4sswd -b \
 "cn=Computers,dc=myrealm,dc=net" "cn=systemname" \
 userCertificate

ldapsearch -x -D "ldapproxy@myrealm.net" -w s3cr3t_p4sswd -b \
 "cn=Users,dc=myrealm,dc=net" "sAMAccountName=username" \
 userCertificate
```

The first one is to obtain the certificates of a computer account called `systemname` and the second one does the same thing for a user called `username`. Obviously, `ldapproxy` should be replaced by the username of the previously configured LDAP proxy user, `myrealm` and `net` by the realm, `s3cr3t_p4sswd` by the password of the LDAP proxy user.

Obtaining the certificates from LDAP solves the problem of sending encrypted messages without having the certificates of the recipients previously. This is meant for mail user agents, but can not be used get the certificates from the Linux platform. Nonetheless, they can be imported.

The certificates stored in a standard format can be imported from many Linux mail applications. Red Hat Enterprise Linux WS 3.0 comes with Mozilla Mail, which will do the job perfectly.

The method for authenticating users in Windows 2000 when using smart cards or tokens is PKINIT, that is still a draft[22] for extending Kerberos. PKINIT takes advantage of the fact that, when using Kerberos authentication, the user password is used only once, at the beginning, to get the TGT. The draft proposes the replacement of the initial authentication based on symmetric keys —as the rest of the Kerberos standard— by an alternative based on public key cryptography. This alternative consists in sending in the initial authentication request the public key data and a authenticator encrypted with the user's private key. Once the user has the TGT, Kerberos authentication proceeds as usual.

Although the authors of the draft of PKINIT claim that using public key cryptography for the initial authentication has the additional benefit of allowing authentication without prior registration of the principal, I think that this is not a real advantage, since the only way a principal can get a smart card with a certificate is by registering.

Authentication in Linux using smart cards is still under development, and a beta version of a PAM module called pkcs11_login[23] is available for testing. This module uses a X.509 certificate accessed in accordance to PKCS#11[24], the Cryptographic Token Interface Standard, which is a standard sponsored by RSA Security. I have also found a patch[25] for Heimdal Kerberos[26] that implements PKINIT for the Schlumberger Cryptoflex Win2k smart card[3].

Finally, Linux is unable to inter-operate transparently with the Encrypted Files System. Of course this inter-operation only makes sense for shared folders, since for files that are local system other alternatives exist[30][31].

In Windows 2000, the encryption of a file or folder, which is inside of a shared folder, takes place in the computer where the files are stored. It is the job of the EFS service to retrieve the keys needed for the encryption/decryption of the file by impersonating the owner of the file. This keys are retrieved from the users profile, and, I haven't found any implementation of this process in Linux. Also, I haven't been able to fully test it personally because it requires the Kerberos authentication of the `smbclient` utility.

2.5 IPSec

IPSec[32] is a security architecture designed on top of the IP protocol —for both versions 4 and 6—that provides encryption and authentication services. It is a public standard and was designed for inter-operability. However, its implementations are relatively new and some issues can be found when establishing IPSec communications between different platforms.

[3]I haven't tried this patch because I don't own a smart card reader of this model

2.5.1 Description

All the processes of IPSec take place at the IP (host-to-host) level (layer 3).

Administrators of IPSec enabled systems can decide the level of security required, which can go from clear text unauthenticated traffic, authenticated traffic —using the Authenticated Headers (AH) protocol[33],— or encrypted and authenticated traffic —using the Encapsulating Security Payload (ESP) protocol[34]. If AH or ESP are used then the session key is negotiated using the Internet Key Exchange (IKE) protocol[35].

Phase 1 of the IKE protocol consists in establishing a Security Association (SA). In this phase the systems are mutually authenticated. Windows 2000 supports three methods for peer authentication:

- preshared key,
- x509 cert, and
- Kerberos ticket, which is the default method.

2.5.2 Integration

FreeS/WAN[36] goal is to provide a free implementation of IPSec with source code available.

I could not install any of the RPMs available from FreeS/WAN website, because they only work with the Red Hat Linux version, and not with the Enterprise one. Thus, I installed it from the source code (version 2.04).

Since it is adds two new protocols to the IP stack the kernel must be ready to deal with them. So the kernel must be patched. The makefile —`make menugo` takes care of everything and takes the user through the process of rebuilding the Linux kernel and installing it with very few interactions. You only need a previously configured and compiled kernel present. It also provides a way to install the kernel if your system uses `make install` as part of the kernel generation process —that is the case of Red Hat Enterprise Linux WS 3.0—- that does all the required steps.

The configuration is quite straight forward. For the Linux system:

1. Edit `/etc/ipsec.conf` adding a connection. The parameters for the *left* side, should be those of your Linux system. The parameters for the *right* side are those of your windows system.
2. Add the preshared key in `/etc/ipsec.secrets`.
3. Start IPSec with `service ipsec start`

In Windows you should put MD5-3DES at the top of the list of the security methods for the filter action.

The connection can be started from the Linux side with
`ipsec auto --up` *`connection-name`*.

There is a patch for FreeS/WAN that enables the use of x.509 certificates for establishing the security associations. This patch distributed by Strongsec[37] and included in Super FreeS/Wan[38]. The newest version comes with support for smart cards and USB tokens.

3 Conclusions

I have made several tests using a Windows 2000 domain controller and several clients, Windows 2000, Windows XP and Red Hat Enterprise Linux WS 3.0. From all the tests I have extracted the following conclusions about how well do Windows 2000 and Linux work together:

- Windows 2000 Active Directory can be used in a secure way to centralize the administration of the users in an environment with Windows and Linux desktop clients. For the Linux clients, all the functionality that is needed is already present —although some packets may need to be recompiled— in the current versions of the most popular Linux distributions, particularly in Red Hat Enterprise Linux WS 3.0. For the Windows 2000 Server that acts as a domain controller, the only additional software (SFU) that is needed can be bought from Microsoft, although other free alternatives exist.

- Microsoft's implementation of Kerberos 5 makes very difficult to use a non Windows system acting as an Active Directory domain controller. Other features, like ticket forwarding have inter-operation issues.

- Windows 2000 DHCP server can be configured to take care of the DDNS updates for the IP addresses that it leases to its clients. This is the easiest way have the IP addresses assigned to the Linux clients registered in the DNS to get the right direct and reverse resolution from it.

- Using a non Windows DHCP server means that the clients won't benefit from the DDNS capabilities of the Windows 2000 DNS server.

- Samba 3.0 allows to use a Linux system as a member server of an Active Directory domain taking advantage of Kerberos and LDAP and making file and printer sharing an straight forward task.

- Windows 2000 PKI certificates can not be directly requested from a Linux system, but they can be exported from a Windows system and imported into the Linux systems later.
- The mechanisms to authenticate users to the Active Directory from Linux using certificates stored in smart cards or tokens are still under development. There is no implementation of PKINIT shipped with Red Hat Enterprise Linux WS 3.0.
- EFS cannot even be used when accessing contents stored in a Windows 2000 share folder. No implementation of the private key retrieval done by the EFS service is available for Linux clients.
- IPSec can be used to protect the communications between Windows 2000 and Linux installing additional software in Red Hat Enterprise Linux WS 3.0 (FreeS/WAN) .

References

[1] "Windows 2000 Kerberos Authentication." Microsoft White Paper 9 Jul. 1999. URL: http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerberos.asp (25 Nov. 2003)

[2] Molnar. G et al. "Windows 2000 Client Startup and Logon Traffic Analysis." Microsoft Enterprise Services White Paper, Aug. 2000. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/windows2000serv/deploy/w2kstart.asp (25 Nov. 2003)

[3] Bryant, B. "Designing an Authentication System: a Dialogue in Four Scenes." Feb. 1988. URL:http://web.mit.edu/kerberos/www/dialogue.html (25 Nov. 2003)

[4] "Introduction to Lightweight Directory Access Protocol (LDAP)." Microsoft Knowledge Base (Q196455) 28 Oct. 2002 URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;196455 (25 Nov. 2003)

[5] Kohl, J. & Neuman, C. "RFC1510: The Kerberos Network Authentication Service (V5)." Internet Engineering Task Force, Sep. 1993. URL: http://www.ietf.org/rfc/rfc1510.txt (22 Nov. 2003)

[6] Brown, K. "Understanding Kerberos Credential Delegation in Windows 2000 Using the TktView Utility." Microsoft Developers Network Magazine: Security Briefs, May 2000. URL: http://msdn.microsoft.com/msdnmag/issues/0500/Security/default.aspx (25 Nov. 2003)

[7] Elson, David. "Active Directory and Linux." Infocus (Security Focus), 3 Apr. 2002. URL: http://www.securityfocus.com/infocus/1563 (17 Sep. 2003)

[8] Morgan, Andrew G. "The Module Writer's Manual." Linux-PAM online documentation v.0.76. 9 May. 2002. URL: http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules.html

[9] Anonymous. "Enable NTP Time Server (Windows 2000/XP)." Registry Guide for Windows. 19 Jul. 2002. URL: http://www.winguides.com/registry/display.php/1117/ (4 Dec. 2003)

[10] Poulin, Martin. “Windows and UNIX interoperability.” Sans InfoSec Reading Room, 2003.
URL: http://www.giac.org/practical/GCWN/Martin_Poulin_GCWN.pdf (5 Aug. 2003)

[11] Combs, G. “Ethereal” URL: http://www.ethereal.com/ (2 Dec. 2003)

[12] Howard, L. “RFC2307: An Approach for Using LDAP as a Network Information Service.” Internet Engineering Task Force, Mar. 1998.
URL: http://www.ietf.org/rfc/rfc2307.txt (29 Nov. 2003)

[13] Larson, M. & Liu, C. “DNS on Windows 2000. 2nd Edition” Madrid: O’Reilly & Associates, Inc., September 2001

[14] Eastlake, D. “RFC2535: Domain Name System Security Extensions” Internet Engineering Task Force, Mar. 1999.
URL: http://www.ietf.org/rfc/rfc2535.txt (1 Dec. 2003)

[15] Vixie, P. et al. “RFC2845: Secret Key Transaction Authentication for DNS (TSIG)” Internet Engineering Task Force, May. 2000.
URL: http://www.ietf.org/rfc/rfc2845.txt (1 Dec. 2003)

[16] Kwan, S. et al. “RFC3645: Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)” Internet Engineering Task Force, Oct. 2003.
URL: http://www.ietf.org/rfc/rfc3645.txt (1 Dec. 2003)

[17] Microsoft Corp. “Using DNS servers with DHCP” Microsoft Windows 2000 Server Documentation.
URL: http://www.microsoft.com/windows2000/en/server/help/
sag_DHCP_imp_InteroperabilityDNS.htm (4 Dec. 2003)

[18] Various. “ISC Dynamic Host Configuration Protocol (DHCP)” ISC website.
URL: http://www.isc.org/products/DHCP/ (4 Dec. 2003)

[19] Various. “SAMBA Project Documentation.” Samba website. 19 Aug. 2003.
URL: http://us4.samba.org/samba/devel/docs/
Samba-HOWTO-Collection.pdf (17 Sept. 2003)

[20] Schneier, Bruce. Secrets & Lies. New York: John Wiley & Sons, 2000.

[21] Kouril, Daniel. “NegotiateAuth.” Mozdev. 6 Dec. 2003.
URL: http://negotiateauth.mozdev.org/index.html (6 Dec. 2003)

[22] Tung, Brian et al. "Public Key Cryptography for Initial Authentication in Kerberos." IETF Internet Drafts 24 Nov. 2003.
URL:http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-17.txt
(5 Dec. 2003)

[23] Strasser, Mario. "PKCS #11 PAM Login Module." 2003.
URL: http://n.ethz.ch/student/mariost/pkcs11_login/ (6 Dec. 2003)

[24] RSA. "PKCS #11 - Cryptographic Token Interface Standard", RSA website.
URL: http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html (17 Sept. 2003)

[25] Holub, P. & Kouril, D. "PKINIT implementation for Heimdal", Meta Center. 27 Oct. 2002.
URL:http://meta.cesnet.cz/software/heimdal/pkinit.en.html (5 Dec. 2003)

[26] Various. "Heimdal Kerberos 5"
URL: http://www.pdc.kth.se/heimdal/heimdal.html (5 Dec. 2003)

[27] Dubrawsky, Ido."Cryptographic Filesystems, Part One: Design and Implementation." Infocus (Security Focus), 7 Mar. 2003.
URL: http://www.securityfocus.com/infocus/1673 (17 Sept. 2003)

[28] Dubrawsky, Ido."Cryptographic File Systems, Part Two: Implementation." Infocus (Security Focus), 14 Apr. 2003.
URL: http://www.securityfocus.com/infocus/1685 (17 Sept. 2003)

[29] Various. "Public Key Cryptography for Initial Authentication in Kerberos." IETF Drafts. 12 Mar. 2002.
URL: http://www.ietf.org/proceedings/03mar/I-D/
draft-ietf-cat-kerberos-pk-init-16.txt (17 Sept. 2003)

[30] Various. "The GNU/Linux CryptoAPI site" 12 May 2003.
URL:http://www.kerneli.org/ (6 Dec. 2003)

[31] Zadok, Erez. "File System Translator." FiST Home Page. 19 Apr. 2003.
URL: http://www1.cs.columbia.edu/∼ezk/research/fist/ (6 Dec. 2003)

[32] Kent, S. & Atkinson, R. "RFC2401: Security Architecture for the Internet Protocol." Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2401.txt (6 Dec. 2003)

[33] Kent, S. & Atkinson, R. "RFC2402: IP Authentication Header." Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2402.txt (6 Dec. 2003)

[34] Kent, S. & Atkinson, R. “RFC2406: IP Encapsulating Security Payload (ESP).” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2406.txt (6 Dec. 2003)

[35] Harkins, D. & Carrel, D. “RFC2409: The Internet Key Exchange (IKE).” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2409.txt (6 Dec. 2003)

[36] Various. “FreeS/WAN Documentation” FreeS/WAN website. 15 Apr. 2003.
URL: http://www.freeswan.org/freeswan_trees/freeswan-2.02/doc/index.html (17 Sept. 2003)

[37] Steffen, Andreas. “X.509 Certificate Support for the Linux FreeS/WAN IPsec Stack.” 14 Nov. 2003. http://www.strongsec.com/freeswan/

[38] “Contributed Patches” Super FreeS/WAN website.
URL: http://www.freeswan.ca/patches/ (17 Sept. 2003)