Chapter 18. Securing Samba

Andrew Samba Team Tridgell

Samba Team

<tridge@samba.org>

John H. Samba Team Terpstra

Samba Team

<jht@samba.org>

May 26, 2003

Table of Contents

Introduction

The information contained in this chapter applies in general to all Samba installations. Security is everyone's concern in the information technology world. A surprising number of Samba servers are being installed on machines that have direct internet access, thus security is made more critical than it would have been had the server been located behind a firewall and on a private network. Paranoia regarding server security is causing some network administrators to insist on the installation of robust firewalls even on servers that are located inside secured networks. This chapter provides information to assist the administrator who understands how to create the needed barriers and deterents against “the enemy”, no matter where [s]he may come from.

> A new apprentice reported for duty to the chief engineer of a boiler house. He said, “Here I am, if you will show me the boiler I'll start working on it.” Then engineer replied, “You're leaning on it!”

Security concerns are just like that. You need to know a little about the subject to appreciate how obvious most of it really is. The challenge for most of us is to discover that first morsel of knowledge with which we may unlock the secrets of the masters.

Features and Benefits

There are three levels at which security principles must be observed in order to render a site at least moderately secure. They are the perimeter firewall, the configuration of the host server that is running Samba, and Samba itself.

Samba permits a most flexible approach to network security. As far as possible Samba implements the latest protocols to permit more secure MS Windows file and print operations.

Samba can be secured from connections that originate from outside the local network. This can be done using *host-based protection*, using Samba's implementation of a technology known as "tcpwrappers," or it may be done be using *interface-based exclusion* so smbd will bind only to specifically permitted interfaces. It is also possible to set specific share- or resource-based exclusions, for example, on the *[IPC$]* autoshare. The *[IPC$]* share is used for browsing purposes as well as to establish TCP/IP connections.

Another method by which Samba may be secured is by setting Access Control Entries (ACEs) in an Access Control List (ACL) on the shares themselves. This is discussed in File, Directory, and Share Access Controls.

Technical Discussion of Protective Measures and Issues

The key challenge of security is that protective measures suffice at best only to close the door on known exploits and breach techniques. Never assume that because you have followed these few measures, the Samba server is now an impenetrable fortress! Given the history of information systems so far, it is only a matter of time before someone will find yet another vulnerability.

Using Host-Based Protection

In many installations of Samba, the greatest threat comes from outside your immediate network. By default, Samba accepts connections from any host, which means that if you run an insecure version of Samba on a host that is directly connected to the Internet, you can be especially vulnerable.

One of the simplest fixes in this case is to use the hosts allow and hosts deny options in the Samba `smb.conf` configuration file to allow access to your server only from a specific range of hosts. An example might be:

```
hosts allow = 127.0.0.1 192.168.2.0/24 192.168.3.0/24
hosts deny = 0.0.0.0/0
```

The above will allow SMB connections only from `localhost` (your own computer) and from the two private networks 192.168.2 and 192.168.3. All other connections will be refused as soon as the client sends its first packet. The refusal will be marked as `not listening on called name` error.

User-Based Protection

If you want to restrict access to your server to valid users only, then the following method may be of use. In the `smb.conf` *[global]* section put:

```
valid users = @smbusers, jacko
```

This restricts all server access either to the user *jacko* or to members of the system group *smbusers*.

Using Interface Protection

By default, Samba accepts connections on any network interface that it finds on your system. That means if you have an ISDN line or a PPP connection to the Internet then Samba will accept connections on those links. This may not be what you want.

You can change this behavior using options like this:

```
interfaces = eth* lo
bind interfaces only = yes
```

This tells Samba to listen for connections only on interfaces with a name starting with `eth` such as `eth0` or `eth1`, plus on the loopback interface called `lo`. The name you will need to use depends on what OS you are using. In the above, I used the common name for Ethernet adapters on Linux.

If you use the above and someone tries to make an SMB connection to your host over a PPP interface called `ppp0`, then [s]he will get a TCP connection refused reply. In that case, no Samba code is run at all, because the operating system has been told not to pass connections from that interface to any Samba process. However, the refusal helps a would-be cracker by confirming that the IP address provides valid active services.

A better response would be to ignore the connection (from, for example, ppp0) altogether. The advantage of ignoring the connection attempt, as compared with refusing it, is that it foils those who probe an interface with the sole intention of finding valid IP addresses for later use in exploitation or denial of service attacks. This method of dealing with potential malicious activity demands the use of appropriate firewall mechanisms.

Using a Firewall

Many people use a firewall to deny access to services they do not want exposed outside their network. This can be a good idea, although I recommend using it in conjunction with the above methods so you are protected even if your firewall is not active for some reason.

If you are setting up a firewall, you need to know what TCP and UDP ports to allow and block. Samba uses the following:

Port 135/TCP - used by smbd
Port 137/UDP - used by nmbd
Port 138/UDP - used by nmbd
Port 139/TCP - used by smbd
Port 445/TCP - used by smbd

The last one is important because many older firewall setups may not be aware of it, given that this port was only added to the protocol in recent years.

When configuring a firewall, the high order ports (1024-65535) are often used for outgoing connections and therefore should be permitted through the firewall. It is prudent to block incoming packets on the high order ports except for established connections.

Using IPC$ Share-Based Denials

If the above methods are not suitable, then you could also place a more specific deny on the IPC$ share that is used in the recently discovered security hole. This allows you to offer access to other shares while denying access to IPC$ from potentially untrustworthy hosts.

To do this you could use:

```
[IPC$]
hosts allow = 192.168.115.0/24 127.0.0.1
hosts deny = 0.0.0.0/0
```

This instructs Samba that IPC$ connections are not allowed from anywhere except the two listed network addresses (localhost and the 192.168.115 subnet). Connections to other shares are still allowed. Because the IPC$ share is the only share that is always accessible anonymously, this provides some level of protection against attackers who do not know a valid username/password for your host.

If you use this method, then clients will be given an `access denied' reply when they try to access the IPC$ share. Those clients will not be able to browse shares and may also be unable to access some other resources. This is not recommended unless for some reason you cannot use one of the other methods just discussed.

NTLMv2 Security

To configure NTLMv2 authentication, the following registry keys are worth knowing about:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"lmcompatibilitylevel"=dword:00000003
```

The value 0x00000003 means to send NTLMv2 response only. Clients will use NTLMv2 authentication; use NTLMv2 session security if the server supports it. Domain controllers accept LM, NTLM, and NTLMv2 authentication.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0]
"NtlmMinClientSec"=dword:00080000
```

The value 0x00080000 means permit only NTLMv2 session security. If either NtlmMinClientSec or NtlmMinServerSec is set to 0x00080000, the connection will fail if NTLMv2 session security is negotiated.

Upgrading Samba

Please check regularly on http://www.samba.org/ for updates and important announcements. Occasionally security releases are made, and it is highly recommended to upgrade Samba promptly when a security vulnerability is discovered. Check with your OS vendor for OS-specific upgrades.

Common Errors

If all Samba and host platform configurations were really as intuitive as one might like them to be, this chapter would not be necessary. Security issues are often vexing for a support person to resolve, not because of the complexity of the problem, but because most administrators who

post what turns out to be a security problem request are totally convinced that the problem is with Samba.

Smbclient Works on Localhost, but the Network Is Dead

This is a common problem. Linux vendors tend to install a default firewall. With the default firewall in place, only traffic on the loopback adapter (IP address 127.0.0.1) is allowed through the firewall.

The solution is either to remove the firewall (stop it) or modify the firewall script to allow SMB networking traffic through. See the Using a Firewall section.

Why Can Users Access Other Users' Home Directories?

“ We are unable to keep individual users from mapping to any other user's home directory once they have supplied a valid password! They only need to enter their own password. I have not found any method to configure Samba so that users may map only their own home directory. ”

“ User xyzzy can map his home directory. Once mapped, user xyzzy can also map anyone else's home directory. ”

This is not a security flaw, it is by design. Samba allows users to have exactly the same access to the UNIX file system as when they were logged on to the UNIX box, except that it only allows such views onto the file system as are allowed by the defined shares.

If your UNIX home directories are set up so that one user can happily `cd` into another user's directory and execute `ls`, the UNIX security solution is to change file permissions on the user's home directories so that the `cd` and `ls` are denied.

Samba tries very hard not to second guess the UNIX administrator's security policies and trusts the UNIX admin to set the policies and permissions he or she desires.

Samba allows the behavior you require. Simply put the only user = %S option in the *`[homes]`* share definition.

The only user works in conjunction with the users = list, so to get the behavior you require, add the line:

```
users = %S
```

This is equivalent to adding

```
valid users = %S
```

to the definition of the *`[homes]`* share, as recommended in the `smb.conf` man page.