Cybersecurity

Subject: Controlling access to network services

Most of the traditional network services and application protocols had not been designed with security in mind. Some of them, continuously in use today, have undergone considerable improvements, nevertheless many of them still represent a vulnerable entry point to the underlying operating system. Proper protection against misuse of those services requires significant amount of configuration, which is often a tedious and error-prone task. The goal of the current laboratory is to get accustomed to access control policy mechanisms for managing network access to services in Unix/Linux family of operating systems.

1. Remote access control to network services

1.1 xinetd – internet service daemon

The task of the xinetd daemon (originally inetd) is to make basic network services available through the network ports of the local operating system. The xinetd receives client requests from the network and forwards them to the appropriate server processes that handle these requests. Some of the simplest services are handled internally by xinetd itself, and to handle the rest, xinetd simply starts external programs (as the servers) according to its configuration.

The configuration of the xinetd daemon is split into the main configuration file /etc/xinetd.conf and a set of separate configuration files for each service, usually stored in the /etc/xinetd.d/ directory. A sample configuration for the legacy `rlogin` service (*remote login*) might look like this:

file /etc/xinetd.d/rlogin:

```
service login
{
     socket_type       = stream
     protocol          = tcp
     flags             = NAMEINARGS
     wait              = no
     server            = /usr/sbin/in.rlogind
     server_args       = -a
     user              = root
     group             = root
     disable           = no
}
```

According to this configuration xinetd would provide a service called login (in fact it is the actual `rlogin` network service) available through the TCP protocol, and implemented by an external daemon, run (for each connection separately) from /usr/sbin/in.rlogind program with root privileges.

The `disable` option can be used to deactivate a service (xinetd will then not process network packets incoming to the port of this service). Deactivation of services that are unnecessary or known to have security issues is considered crucial for increasing the security of the entire operating system.

You should keep in mind, that despite the existence of xinetd, *users can run perfectly well their own processes communicating through the network (that is, servers for their own private services), which will then not fall under the control of* xinetd *and, if not properly supervised, may endanger security of the system.*

1.2 Access control mechanism

If, after disabling all those services which have been judged dangerous or unwanted for any other reason, hosting some network services is still required, it is absolutely essential to use some system-wise control mechanism for communication with these services. One of its goals would be to allow accessing services only from selected remote systems that are considered trustworthy according to the current security policy. To some extent, access control can be handled by the server process of a particular service itself. However, first, it requires having a properly adapted server for each service separately, and second—it is extremely difficult to maintain a consistent security policy for a large number of services. In order to make this operation more efficient an intermediary process can be introduced between the xinetd daemon and the server of the considered service. This process—TCP wrapper (the tcpd daemon)—manages the access control to services supported by xinetd. It operates on two lists describing the access control policy—the list of explicit permissions (file /etc/hosts.allow) and explicit access denials (file /etc/hosts.deny). Each new connection request is then confronted with policy rules in the first list, and if this does not bring a decision—on the second list.

The TCP wrapper can be intertwined between xinetd and the final network service process, which requires a fairly simple modification of the corresponding configuration file for that service, such as the following one:

```
service login
{
     socket_type       = stream
     protocol          = tcp
     flags             = NAMEINARGS
     wait              = no
     server            = /usr/sbin/tcpd
     server_args       = /usr/sbin/in.rlogind -a
     user              = root
     group             = root
     disable           = no
}
```

In case of applying configuration changes to an already running xinetd daemon, the configuration can be simply reloaded with the command:

```
service xinetd reload
```

An sample service access control policy (i.e. the tcpd configuration) may look like this:

file /etc/hosts.allow:

```
ftpd:       ALL
in.rlogind: hq.mi6.net   150.254.27.45 \
            jbond@agents.mi6.net \
            150.254.32.0/255.255.254.0
ALL:        LOCAL
```

file /etc/hosts.deny:

```
ALL: PARANOID
ALL: ALL : twist /bin/echo -e "\nAccess denied to %d for %u@%h.\n\r"
```

This configuration provides access to the ftp service (or more precisely, the ftpd daemon) from any remote system. The rlogin service (in.rlogind daemon) can be accessed from hq.mi6.net and 150.254.27.45 computers, and, only by a particular user, jbond, from agents.mi6.net, as well as by any computer from the 150.254.32.0 and 150.254.33.0 networks (!). Any other services (ALL) are available only from local-scope hostnames (not FQDN). Other access attempts will be silently rejected, regardless of the service in question, if the domain name provided in the request does not match the IP address of the reporting client (PARANOID), and in any other case—with an access denial message sent back to the client.

1.2.1 Newer syntax of the TCP wrapper configuration

In newer versions of the TCP wrapper, the access control policy can be configured within a single configuration file, i.e. /etc/hosts.allow itself (both policy files, however, can still be used together). A new optional keyword ALLOW or DENY may then be explicitly used (after an additional colon), as in the following example:

```
in.rlogind: ALL EXCEPT foo.mi6.net : ALLOW
in.rlogind: foo.mi6.net : spawn /bin/echo "rlogin z %h" | mail -s "alert %H" root
ALL : ALL : DENY
```

1.3 xinetd build-in access control policy

The recent versions of xinetd daemon, along with some other network servers (including sshd, for instance), have the access control support built-in using the libwrap library. Since the xinetd daemon itself uses libwrap to read the /etc/hosts.allow and /etc/hosts.deny policy configuration files, there is no further need to use the tcpd daemon the way it was described in section 1.2.

Moreover, it is possible to define elements of the service access policy directly in the xinetd's configuration files from the /etc/xinet.d/ directory. Three configuration parameters can be used for this purpose:

- no_access,
- only_from,
- access_times.

Build-in control example:

```
service login
{
  socket_type       = stream
  protocol          = tcp
  flags             = NAMEINARGS
  wait              = no
  server            = /usr/sbin/in.rlogind
  server_args       = -a
  user              = root
  group             = root
  only_from         = 192.168.10.0/8
  access_times      = 08:00-12:15
  log_on_success    += USERID
  log_on_failure    += USERID
  disable           = no
}
```