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. System services remote access control mechanisms

1.1 xinetd – internet service daemon

Basic web services are made available through the xinetd daemon (originally inetd), whose task is to receive service client requests from the network and direct them to the appropriate processes that handle these requests (service servers). Some of the simplest services are handled by xinetd itself, and to handle the rest, xinetd runs processes assigned to the appropriate services according to the configuration. Originally (for the older version of inetd daemon) the configuration was stored in a single /etc/inetd.conf file. For example:

```
#<service> tli <proto> <flags> <user>  <server_pathname>    <args>

echo      stream  tcp   nowait  root    internal
echo      dgram   udp   wait    root    internal
daytime   stream  tcp   nowait  root    internal
daytime   dgram   udp   wait    root    internal

ftp       stream  tcp   nowait  root    /usr/sbin/in.ftpd    in.ftpd
shell     stream  tcp   nowait  root    /usr/sbin/in.rshd    in.rshd
login     stream  tcp   nowait  root    /usr/sbin/in.rlogind in.rlogind -a
talk      dgram   udp   wait    root    /usr/sbin/in.talkd   in.talkd
finger    stream  tcp   nowait  nobody  /usr/sbin/in.fingerd in.fingerd
```

According to this configuration xinetd would provide, among others, a daytime service on both TCP and UDP transport protocols, internally supported by the internet daemon itself, and a service on the port called login (in fact it is the `rlogin` network service, already known to us) available only through the TCP protocol. It is implemented by an external daemon, run (for each connection separately) from /usr/sbin/in.rlogind with root privileges.

Compared to the older version, the newer xinetd daemon includes a large number of functional enhancements, also in terms of security (such as event logging, to name one only). The configuration of the xinetd daemon is diveded into the main configuration file /etc/xinetd.conf and separate configuration files for each service, usually stored in the /etc/xinetd.d directory (loaded by the main configuration file). A sample configuration for the `rlogin` service 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
}
```

The `disable` option deactivates the service (xinetd will not receive requests directed to the port of this service). Deactivation of services that are unnecessary or threaten to maintain the security of the operating system is an crucial element of increasing the security level of the entire network.

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

1.2 Service access control

If, after disabling all those services which have been judged unnecessary, hosting some network services is still required, it is absolutely essential to use some communication control mechanism with these services. One of its goals would be to allow accessing services only from selected remote systems that are considered trustworthy according to your 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—tcpd, or TCP wrapper—deals with the access control to services supported by xinetd. It operates on two lists describing the access control policy—the list of explicit permissions of access to services (file /etc/hosts.allow) and explicit access prohibitions (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 settlement—on the second list.

The TCP wrapper can be invoked by xinetd, so that it will be woven between it and the ultimately executed network service process, which requires a fairly obvious modification of the corresponding configuration file of that service, e.g. /etc/xinetd.d/rlogin:

```
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 the already launched internet daemon, the configuration should be reloaded with the command:

```
service xinetd reload
```

An exemplary network services 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 (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 by the jbond user from agents.mi6.net. and by each computer from the 150.254.32.0 and 150.254.33.0 network (!). Any other service (ALL) is available only from local 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 TCP wrapper daemon configuration newer version

In newer versions of the TCP wrapper daemon, the access control policy can be configured with a single configuration file, i.e. /etc/hosts.allow itself. An optional additional definition line item (another colon) may contain the keyword ALLOW or DENY, 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. When the xinetd daemon uses libwrap to read itself the /etc/hosts.allow and /etc/hosts.deny policy configuration files, there is no further need to use the TCP wrapper 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_time.

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_time       = 08:00-12:15
  log_on_success    += USERID
  log_on_failure    += USERID
  disable           = no
}
```