Miniban, Group 1,

We developed miniban, a utility for automatically or manually banning and unbanning IP addresses from interacting with a computer. It supports a whitelist of addresses never to be banned and keeps a tidy database with timestamps on every ban. To access the iptables, the scripts must be executed by the superuser.

We have tried to implement the utility in a modular fashion, such that each script and subroutine can be read, understood, maintained, and updated with ease. The process of banning and unbanning a specific address can be done manually by executing "./ban.sh" and "./unban.sh" respectively. Inside the main miniban.sh-script, the procedures for fetching logs, parsing logs, and reading the database are separated into clean, independent functions.

A short loop at the very end of miniban.sh checks for the last line of the journald-entry for the ssh daemon. If it contains information related to a password login over SSH, the funtion banCheck is called. It double checks the age of the log entry, verifies that it contains an IPv4 address and updates the attempts-counter according to the specific message.

The reverse process, looking through the database of banned addresses and unbanning them at the right time, is done by a loop that runs in the background with the default interval of one minute. When an address is checked, its timestamp is compared to the expiry time, and is unbanned if enough time has passed. If it successfully unbanned in iptables, it is also removed from the database.

We have developed and tested our miniban scripts on a stock install of Debian with PAM installed. This means that our auth logs look like this.

Nov 26 16:12:20 miniban sshd[778947]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=222.187.232.39 user=root

Nov 26 16:12:17 miniban sshd[778945]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=222.187.232.39 user=root

To correctly count the number of login attempts, we have written a test for each type of log entry. This could have been done more cleanly by using a custom logging module in PAM, but by doing everything via the ssh daemon logs, we have written a simple, and most importantly portable, solution to the problem that don't require any additional installation.

As both the ban.sh and unban.sh scripts can be run as background processes, it is important to avoid write conflicts. To avoid this, we use flock with a lock file, so that only a single process can edit the main database at one time.

To achieve our goals and make a robust script, we have used a wide range of UNIX tools, both built-in methods in the bash scripting language and external tools like awk and grep.

The detail we are most unhappy about is the method of detecting a change in the log. This could have been done by using a "following" function like journalctl's "-f"-option to save on processor overhead. By instead constantly polling the latest line of the log, we ensure that we don't repeat lines that have already been handled, and that the software is easily ported to other systems. It is written in such a way that little must be updated in order to work on a distribution without systemd or PAM.