From: Spitzer, Andras ([redacted])
Sent: Monday, March 26, 2012 1:37 PM
To: [redacted] ([redacted])
Cc: [redacted] ([redacted]); [redacted] ([redacted]); [redacted] ([redacted])
Subject: pam ldap & crond tests

Bob,

I ran a lot of tests, please **let me share what I found so far.**

The tests were based on your idea, I worked on gislogbprep01 with the LDAP version file 7400 with having the two LDAP servers :

```
uri ldap://dsproxy6alpha.corporate.[redacted]  ldap://dsproxy6cinci.corporate.[redacted]
```

Alpha as the primary, and Cinci as secondary. The goal was to figure out exactly how the crond works in regards PAM (and via PAM in regards LDAP), also to figure out what happens if we fail to get response from one or both of the configured LDAP servers.

Let's start with the second one, what happens if we fail to communicate with one or both LDAP servers. First, we used the command 'iptables -A INPUT -s 3.34.169.233 -j DROP' to filter out any responding packets from the LDAP server for testing purposes, but then I decided to go with a more realistic approach. This command is blocking any packets received from the IP address 3.34.169.233 (dsproxy6alpha.corporate.[redacted] but in real life it's very rare that we don't receive any packets from the server. This happens if the 1. machine is turned off, or 2. if a firewall drops all the packets.

The more realistic approach is that even though the system is up, the LDAP server is not available for some reason. Although when this is the case, we do receive a TCP RST packet back from the system (this is when we see "connection refused" from the telnet). The difference is important, because if we just drop the packets, the timeout values (hence the reaction time from the LDAP module) are different, as TCP has to wait for its timers to expire, and this is independent from LDAP. This was the iptables command I used to simulate a TCP RST :

```
iptables -D OUTPUT -d 3.34.169.233 -p tcp -j REJECT --reject-with tcp-reset
```

And I also measured the difference between packet drops and TCP reset cases, for how long does it take to get answer if the first LDAP server fails and pam_ldap has to fail over to the second LDAP :

WITH RETURN-RST (iptables -D OUTPUT -d 3.34.169.233 -p tcp -j REJECT --reject-with tcp-reset)

```
-bash-3.00# while true ; do (time getent passwd alma )  2>&1 | sed
's/^[^m]*m//g;s/[.s]//g' | awk '{i = i + $0} END { print i / 1000 "s" }' ; sleep 5 ;
done
1.447s
1.454s
1.448s
1.44s
1.449s
1.448s
1.44s
1.443s
1.454s
```

As you can see here I blocked the first LDAP server using TCP RST, and I received answer from pam_ldap in about 1,5sec. Now, let's see what happens if I block the same LDAP, but by dropping packets :

WITH DROP (iptables -A INPUT -s 3.34.169.233 -j DROP)

```
-bash-3.00# while true ; do (time getent passwd alma )  2>&1 | sed
's/^[^m]*m//g;s/[.s]//g' | awk '{i = i + $0} END { print i / 1000 "s" }' ; sleep 5 ;
done
6.444s
6.443s
6.439s
6.448s
6.451s
6.449s
6.436s
6.444s
6.446s
```

This way the pam_ldap failover takes way longer, around 6,5sec. And finally, just out of curiosity let's see how long does it take to pam_ldap to timeout when both LDAP servers are blocked :

WITH DROP BOTH SERVERS (iptables -A INPUT -s 3.32.197.157 -j DROP, iptables -A INPUT -s 3.34.169.233 -j DROP)

```
-bash-3.00# while true ; do (time getent passwd alma )  2>&1 | sed
's/^[^m]*m//g;s/[.s]//g' | awk '{i = i + $0} END { print i / 1000 "s" }' ; sleep 5 ;
done
10.777s
10.657s
10.649s
10.65s
10.652s
10.65s
10.654s
10.651s
10.652s
10.65s
10.649s
10.65s
```

Around ~10sec. So the lesson is what you already realized Bob, is that even if we have two LDAP servers defined in /etc/ldap.conf and the first one is failing (TCP reset or DROP, either way), pam_ldap will try to reconnect to it each and every time still. In short, pam_ldap does not remember if an LDAP server fails, it will try to connect to it each and every time it's called.

Now, **about the pam configuration**.

It's important to differentiate nss_ldap and pam_ldap. nss_ldap is used for name services and it is used if we have the ldap keyword in /etc/nsswitch.conf. pam_ldap module is used if we have the ldap library defined in any of our pam configuration files under /etc/pam.d. During in our cron issue, we only have to deal with pam_ldap.True though that if we can't connect to any of LDAP servers, even nss_ldap will take a long time to respond (~30 sec), but nss_ldap failing (can't connect to any of the LDAP servers) will not skip any jobs in cron. Only pam_ldap failing will skip cron jobs.

Also a difference between nss_ldap and pam_ldap is that if we modify the /etc/nsswitch.conf (removing all the ldap keywords for example), it will take effect only by restarting the cron daemon. pam_ldap is different, if we modify any of the pam configuration files under /etc/pam.d, that will be immediate, and we don't have to restart the cron daemon. But both modules will complain in /var/log/messages if there is an error with any of the LDAP servers! But in regards crond, we just have to worry about the pam_ldap messages.

The way I tested crond whether it's using nss_ldap and pam_ldap is simple, I've set up a job to run every minute /bin/true, then I used strace : 'strace -p <crond pid>-e trace=open -f -t -q -o cron.out' to see which libs are loaded every minute. The reason I used /bin/true because I wanted to have the simplest program possible so it won't cause

much noise in the strace output (as –f follows the forks, we'll strace the scheduled command as well), and /bin/true is made out of a single command, which is exit(0).

Using this strace command you'll see all the libs which are used crond. Again, true though that even if we create a pam_ldap free pam configuration file for cron, crond will still use nss_ldap because of /etc/nsswitch.conf, but we don't worry about that as the cron jobs does not depend on that. cron jobs only depend on pam_ldap, so that's what I will focus on. Now, let's the variations on crond pam files :

the default /etc/pam.d/crond :

```
#
# The PAM configuration file for the cron daemon
#
#
auth       sufficient pam_rootok.so
auth       required   pam_stack.so service=system-auth
auth       required   pam_env.so
account    required   pam_stack.so service=system-auth
account    required   pam_access.so
session    required   pam_limits.so
session    required   pam_loginuid.so
```

This one will use LDAP if we have ldap configured in /etc/pam.d/system-auth, and we have. The reason is simple, pam_stack.so works as an include directive, it will read /etc/pam.d/system-auth and process any pam modules declared there. As we have pam_ldap configured there, it will use LDAP. That's our standard and current crond pam file we use today.

Proposal #1 (file can be found at gislogbprep01:/etc/pam.d/crond_without_LDAP) :

```
#
# The PAM configuration file for the cron daemon
#
#
auth       sufficient pam_rootok.so
auth       required   pam_env.so
account    required   pam_access.so
session    required   pam_limits.so
session    required   pam_loginuid.so
```

What I did here is very simple. I removed the pam_stack.so lines, so it won't include any other pam config files. As this file does not include pam_ldap, this version won't use pam_ldap, so cron jobs will not depend on ldap anymore.

Proposal #2 (file can be found at gislogbprep01:/etc/pam.d/crond_dummy) :

```
#
# The PAM configuration file for the cron daemon
#
#

auth       required   pam_permit.so
account    required   pam_permit.so
session    required   pam_permit.so
```

Even though we can't turn off PAM in crond, the closest we can get to disabling it is to create a dummy crond pam file, like this. Using only the simplest pam module, pam_permit.so will accept any PAM requests from cron, hence the jobs will not depend on LDAP again.

Tested, both proposed versions are working. Bob, please test them too using the strace command wrote above, and if you find it not working as I described, please let me know. Please don't forget, we care only about pam_ldap, nss_ldap depends on nsswitch.conf, and job execution does not depend on that. Even if nss_ldap can't connect to

LDAP, jobs will still run. Only pam_ldap can stop jobs running in case of connectivity error to the LDAP servers.

Why I had two proposals? Even though Proposal #1 looks good and looks the closest to what RedHat release as the default crond pam file, we might say it's the closest version of RedHat without the dependency of any other pam file. The reason why I would still prefer proposal #2 is that if we want to deploy this crond pam file across our Linux farm, we have to make sure the modules listed in the pam file does exist in all the different versions of Linux we use.. For example if an older OEL version does not have pam_loginuid.so, pam will fail again and even though we fixed a problem in the same time we created a new one. With proposal #2, we only use pam_permit.so, which is one of the most basic pam modules ever and it's part of the pam package since the beginning. So the chances are that a Linux version does not have pam_permit.so is very, very low.

So from an operational point of view, I would vote for proposal #2, as it's simple hence it's more robust. (btw we also can run an Opsware audit before a company-wide deployment to make sure all our Linuxes have pam_permit.so installed, or we also can include a pre-check during the deployment to use the dummy cron file only if there is a pam_permit.so in the system)

Bob, please let me know what you think.

Regards,
sendai