Overcoming Active Directory Woes with Plain Text Caches and Replacing Passwords

Jason St. John
Guardant Health
Palo Alto, California, USA
jstjohn@guardanthealth.com

Alex Younts
Guardant Health
Palo Alto, California, USA
ayounts@guardanthealth.com

ABSTRACT

Reliable authentication is a key component of all HPC systems. This paper discusses an approach that bypasses systemic authentication problems experienced by the authors to provide a simple and reliable manner of managing service accounts and user groups for HPC centers using plain text caches and alternatives to passwords.

KEYWORDS

authentication, high performance computing, reliability

ACM Reference Format:
Jason St. John and Alex Younts. 2023. Overcoming Active Directory Woes with Plain Text Caches and Replacing Passwords. In *Proceedings of HPC-SYSPROS23: HPC System Professionals Workshop (SC '23).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Authentication and authorization systems are fundamental components of IT systems. Centralized authentication systems must support a comprehensive list of downstream clients and use cases, which increases their complexity and configuration. This complexity makes designing, building, and maintaining authentication systems a regular pain point, especially when dealing with legacy systems. In this paper, we discuss an alternative strategy to leverage centralized authentication with new approaches to simplify this complex infrastructure for both system administrators and end users based on our experience deploying such at Guardant Health.

2 THE PROBLEM

We have experienced a long list of persistent authentication problems on HPC systems at Guardant Health. These authentication problems caused many job start failures every day because compute nodes were unable to resolve user and group names when jobs launched. The site scheduler, Sun Grid Engine, also set nodes offline and impacted system throughput requiring daily attention.

We found many underlying issues, all of which were causes of these authentication query failures for users, groups, and passwords.

SC '23, November 12–17, 2023, Denver, CO

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

The notable issues we discovered included the following:

- Compute nodes were connecting to Active Directory (AD) behind a NAT gateway, and AD was rate limiting queries based on source IP address.
- Some hosts shared Kerberos keytab files while others did not.
- Organizational Unit (OU) assignments for compute nodes and HPC infrastructure hosts had large variation.
- Users in AD did not have Linux UID/GIDs, home directories, and default shells assigned.
- Remote AD servers were unreliable, would routinely fail to answer queries, and would make servers' sssd daemons go "offline".
- sssd misconfigurations, configuration variability, and cruft.

3 THE SOLUTION

`libnss-extrausers` is a Debian package that provides extra passwd, shadow, and group files[3]; however, we did not implement this for shadow. It can be enabled by adding "`extrausers`" into `/etc/nsswitch.conf` as an additional data source. The relevant section of `/etc/nsswitch.conf` is found below:

```
passwd:     files extrausers
group:      files extrausers
```

This tells Name Service Switch (NSS)[2] first to check the standard passwd and group files under `/etc` and then to check `/var/lib/extrausers/`. The extra files under `/var/lib/extrausers` are in the same format as those found under `/etc`, and these function as a node-local NSS cache for our upstream AD databases. The `libnss-extrausers` library is essentially a copy of the code used to access the standard system files. The Debian project maintains a number of useful patches to the latest version of the library. We also produced an internal patch to allow an `extrausers` group below its hard-coded GID limit of 500 because of a legacy system group with a GID of 420. This code was compiled and distributed to our infrastructure and compute nodes as an RPM package.

3.1 Alternative Libraries

The authors had familiarity with `libnss-extrausers`, but there are other libraries including ones distributed with Red Hat Enterprise Linux. These include nss-altfiles[4], nss_db[1], and pam_extrausers[5]. We would like to note that nss_db is particularly difficult to make function properly between multiple glibc versions and with groups containing spaces in their names.

3.2 Generating the extrausers files

We host a private repository on GitHub from which our backend infrastructure machines pull updates on an hourly basis via cron. This repository contains plain text flat files of supplemental account and group information, primarily for service accounts, custom groups, and group memberships for various system authorization policies.

We use a Bash script on backend infrastructure servers to dump the upstream, enterprise AD user and group databases into Linux-compatible, plain text files. The Bash script sanitizes the AD databases to remove junk data and concatenates these files with our version-controlled supplemental authentication databases from GitHub. This provides a union of the AD database with our supplemental system accounts and groups in plain text flat files that gets written to a shared, parallel file system. All HPC infrastructure, login nodes, and compute nodes are scheduled via cron to update their local authentication databases every hour.

Because these plain text files are human-readable, special tools are not necessary to interact with them—the administrator needs only a text editor and a Git client. The files are cached locally on compute nodes which prevents network or service outages from impacting running jobs.

The infrastructure that pulls together the final caches has been an excellent place to implement filters and transforms. We are able to override parameters like shell location and home directory for service accounts, as well as filter out user and group names with incompatible characters like "!", "#", "$", and spaces.

A simplified version of the generation script:

```
# Get copies of identity data from AD through sssd
getent -s sss passwd | sort > passwd.tmp
getent -s sss group  | sort > group.tmp

# Copy static users into files
cat /var/db/passwd.static >> passwd.tmp
cat /var/db/group.static  >> group.tmp

mv passwd.tmp /var/lib/extrausers/passwd
mv group.tmp  /var/lib/extrausers/group
```

3.3 Group-based Authorization

The version-controlled custom files for user groups allow for trivial management of user authorization among federated HPC clusters, where system administrators need to grant access for teams or research groups to dedicated clusters.

For example, the following two lines to a supplemental extrausers group file would add users to a privileged group depending on which cluster the users and administrators are assigned:

```
hpc-cluster1-users:*:12345:user1,user2,user3,user4
hpc-cluster1-admins:*:67890:admin1,admin2
hpc-cluster2-users:*:12346:user5,user6,user7
hpc-cluster2-admins:*:67891:admin3,admin4
```

To authorize only the proper users to log into the HPC system named "cluster1", add the following line to `/etc/ssh/sshd_config`:

```
AllowGroups hpc-cluster1-users hpc-cluster1-admins
```

Using plain text group files and a Git client makes this workflow easier and quicker than sending in tickets to a central IT team or having to modify authentication databases with convoluted syntax like LDAP.

3.3.1 Users with Many Secondary Groups. From previous experience, the authors find that users with many secondary groups (more than 16) have a tendency to break NFS solutions and require server-side configuration changes to function. Currently, all of our storage is mounted using the native IBM Spectrum Scale client.

3.3.2 User Offboarding. Authorization to use system resources is revoked for many reasons and, in most circumstances, this action is sensitive in nature. We have ensured that users who needed to lose access to HPC systems would automatically be removed without our manual intervention and recommend that other engineers consider this condition as well.

3.4 Passwords

Given the widely known security risks of password authentication, and that we still relied on AD to authenticate passwords, we also wanted a way to disable password authentication as broadly as possible. Passwords are not sufficiently supported via libnss-extrausers, however.

3.4.1 SSH Keys. SSH keys are more secure than passwords, even when created with a null password, and they are widely adopted in the HPC community. Users cannot write them down on sticky notes; in fact, they do not even need to because SSH clients will automatically log them in if configured properly with `authorized_keys` and `ssh-agent`. User SSH keys also provide additional auditing and the ability to revoke compromised keys. Additionally, users prefer SSH keys because they just work without any notable effort.

Once SSH keys and libnss-extrausers are in place, administrators can disable nscd, nslcd, sssd, slapd, or whatever compatibility layer is being used.

3.4.2 Google Authenticator. However, users that must sudo still need a way to authenticate their session when invoking sudo, so we must provide a secure alternative to entering sudo passwords. Google Authenticator provides a PAM module that can be inserted into `/etc/pam.d/sudo`. Google Authenticator only requires two pieces: (a) synchronized time between the authenticating device (e.g. smartphone) and the backend authentication server and (b) some sort of shared storage so the backend authentication servers can read authentication secrets. This provides a single factor of authentication that is a suitable replacement for a password.

4 FUTURE WORK

Something must bootstrap users' SSH keys and Google Authenticator secrets. The user's public SSH key must be placed into their `/home/$USER/.ssh/authorized_keys` file, and the secret key for Google Authenticator must be installed on a file system readable by the backend authentication servers. These tasks could be done during user onboarding by system administrators, but this is a ticket generator for IT teams. Ideally, a web portal that is gated by an organization's single sign-on (SSO) portal would provide forms for users to enter their SSH public key and Google Authentication secrets, which subsequently get written to a shared home file system. Such a web portal would be offloading password authentication

to the enterprise IT department while enabling users to self-serve revocation and updates to their SSH key and Google Authenticator secret key for lost or stolen devices. Today, we use a bastion host that sits in our Access Zone that authenticates using AD passwords and allows users to edit their authorized_keys file.

5 CONCLUSION

In this paper, we discuss a simple alternative to complex authentication schemes like AD and LDAP that removes all requirements for additional authentication server daemons and allows critical HPC infrastructure to remove the dependency on upstream authentication servers entirely. Our method gets rid of passwords for users entirely, relying on SSH keys to harden infrastructure.

REFERENCES

[1] Ulrich Drepper and Mark Kettenis. 2011. *nssdb.* Retrieved August 3, 2023 from https://sourceforge.net/projects/nssdb
[2] Free Software Foundation Inc. 2023. *The GNU C Library Reference Manual.* Retrieved August 3, 2023 from https://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html
[3] Bernhard R. Link. 2001. *libnss-extrausers.* Retrieved August 1, 2023 from https://packages.debian.org/en/source/sid/libnss-extrausers
[4] Adrian Perez. 2014. *nss-altfiles.* Retrieved August 3, 2023 from https://github.com/aperezdc/nss-altfiles
[5] Michael Terry, Elliot Lee, and Jan Rękorajski. 2019. *pam_extrausers.* Retrieved August 3, 2023 from https://manpages.ubuntu.com/manpages/impish/man8/pam_extrausers.8.html

Received 04 August 2023; revised unknown; accepted unknown