From: Alex Malin <amalin@lanl.gov>
To: Pam Hamilton <pgh@pop.llnl.gov>
Cc: jallen@pop.llnl.gov, Jon Bringhurst <jonb@lanl.gov>, Cory Lueninghoener <cluening@lanl.gov>, Graham Van Heule <grahamvh@lanl.gov>, kyle_lamb@lanl.gov
Subject: Munge Vulnerabiltiy (CVE-2013-4086)
Date: Mon, 10 Jun 2013 13:28:24 -0600 *(12:28 PDT)*

Hi Pam,

I'd appreciate if you would please pass this information along to Chris
Dunlap and/or others at LLNL, and have them contact Jon Bringhurst at
LANL. Jon discovered what appears to be a bug in munge. He reserved a
CVE name however we have not disclosed the flaw beyond LANL HPC staff.

Our initial assessment is that the difficulty level to exploit this bug
is high; the potential damage is high --root on the cluster and possibly
other clusters if the munge key is the same on multiple clusters. It
likely effects most clusters at LLNL / SNL / LANL and other sites.

Here's Jon's description:

```
> """
> Topic: Munge non-constant time HMAC validationtiming attack
> CVE Name: CVE-2013-4086
> Affects: All known versions of Munge (including the master repository
> source).
>
> WARNING: This vulnerability is under DOE/LANL disclosure embargo until
> further notice.Do not disclose!
>
> Summary: HMACs in Munge are compared using the standard library
> memcmp(3) function. memcmp(3) uses several optimizations which are not
> suited for cryptographic manipulation of HMAC primitives. As a result,
> it is likely possible to utilize the non-constant time property of
> memcmp(3) to reduce the search complexity of a brute-force attack on the
> Munge HMAC. This can be accomplished by recording the time to rejection
> of a forged HMAC by a server process. The time to rejection can be input
> to an application to statistically determine where the HMAC was
> rejected. This will significantly reduce the search space of a
> brute-force attack on a Munge HMAC.
>
> Mitigation: Replace the use of memcmp(3) on HMACs with the constant time
> CRYPTO_memcmp(3ssl). Please note that autotools should be updated to
> check for the existence of CRYPTO_memcmp for a release version of Munge
> (CRYPTO_memcmp is a recent addition to OpenSSL). A patch for Munge HEAD
> follows.
>
> diff --git a/src/munged/dec.c b/src/munged/dec.c
> index 98b1fa8..04a3557 100644
> --- a/src/munged/dec.c
> +++ b/src/munged/dec.c
> @@ -668,7 +668,7 @@ dec_validate_mac (munge_cred_t c)
>
>        /*  Validate new computed MAC against old received MAC.
>         */
> -     if ((n  != c->mac_len) || (memcmp (mac, c->mac, c->mac_len)  != 0)) {
> +     if ((n  != c->mac_len) || (CRYPTO_memcmp (mac, c->mac, c->mac_len)
>  != 0)) {
>            return (m_msg_set_err (m, EMUNGE_CRED_INVALID, NULL));
>       }
>       /*  Ensure an invalid cred error from before is caught
> diff --git a/src/munged/replay.c b/src/munged/replay.c
> index acc04d4..eb405a9 100644
> --- a/src/munged/replay.c
> +++ b/src/munged/replay.c
> @@ -266,7 +266,7 @@ replay_cmp_f (const replay_t r1, const replay_t r2)
>       if (r1->data.t_expired  != r2->data.t_expired) {
>           return (-1);
>       }
> -     if (memcmp (r1->data.mac, r2->data.mac, sizeof (r2->data.mac))) {
> +     if (CRYPTO_memcmp (r1->data.mac, r2->data.mac, sizeof
> (r2->data.mac))) {
>           return (-1);
>       }
>       return (0);
> """
>
> -Jon
> Just a few examples:
>
> * Trust between the slurmctld on the master and the slurmds on the
> computes.
> * Trust between salloc/sbatch/srun on a login node and the slurmd on
> the computes.
> * Trust between slurmd on a compute and a slurmd on another compute.
>
> Basically every message that slurm sends to another slurm process is
> validated by Munge.
```