New issue

SIGSEGV in receive_xattr() on FreeBSD - qsort() called with wrong element count #871

Closed #875

venglin opened last week

Summary

`receive_xattr()` in `xattrs.c` calls `qsort()` with the count of xattr entries received from the wire (`count`), rather than the count of entries actually added to `temp_xattr` (`temp_xattr.count`). When any entries are skipped during receive-side filtering (via `continue`), `qsort` reads beyond the valid portion of the array, passing uninitialized or stale pointers to the comparator. The comparator calls `strcmp(xa->name, ...)` where `xa->name` is NULL, causing SIGSEGV.

Crash Details

```
Signal: SIGSEGV (Segmentation fault)
Process: rsync --server -vlHogDtpAXrxe.iLsfxCIvu --delete --numeric-ids . <dst>

Thread backtrace:
  #0  libc strcmp           — movdqu (%rdi,%rax), %xmm0  [rdi = 0x0 NULL]
  #1  libc qsort            — calls comparator
  #2  rsync receive_xattr   — qsort(temp_xattr.items, count, sizeof(rsync_xa), ...)
  #3  rsync send_file_list / process_file
```

The comparator is:

```c
static int rsync_xal_compare_names(const void *x1, const void *x2)
{
    const rsync_xa *xa1 = x1;
    const rsync_xa *xa2 = x2;
```

```
    return strcmp(xa1->name, xa2->name);   // xa1->name or xa2->name is NULL
}
```

Root Cause

In `receive_xattr()` (xattrs.c), xattr entries received from the network are processed in a loop. Several paths skip an entry without adding it to `temp_xattr`:

```
// path 1 — xattr filter
if (saw_xattr_filter) {
    if (name_is_excluded(name, NAME_IS_XATTR, ALL_FILTERS)) {
        free(ptr);
        continue;        // NOT added to temp_xattr
    }
}

// path 2 — Linux: non-root, non-user namespace
if (am_root <= 0 && !HAS_PREFIX(name, USER_PREFIX)) {
    if (!am_root && !saw_xattr_filter) {
        free(ptr);
        continue;        // NOT added to temp_xattr
    }
    ...
}

// path 3 — non-Linux (FreeBSD etc.): non-user namespace, non-root
} else {
    free(ptr);
    continue;            // NOT added to temp_xattr
}

// path 4 — rsync.%FOO without -XX
if (preserve_xattrs < 2 && ...) {
    free(ptr);
    continue;            // NOT added to temp_xattr
}
```

After the loop, the sort is unconditionally done using the wire count:

```
// BUG: uses 'count' (wire), not 'temp_xattr.count' (actual items added)
if (need_sort && count > 1)
    qsort(temp_xattr.items, count, sizeof (rsync_xa), rsync_xal_compare_names);
```

The memory for `count` entries was pre-allocated at the start of the function:

```
if ((count = read_varint(f)) != 0) {
    (void)EXPAND_ITEM_LIST(&temp_xattr, rsync_xa, count);
```

```
    temp_xattr.count = 0;   // reset — slots beyond temp_xattr.count are uninitialized
}
```

When fewer than `count` entries are added, the trailing slots in `temp_xattr.items` contain either uninitialized memory (first call, fresh malloc) or stale data from a previous invocation (temp_xattr is static). Either way, `rsync_xa.name` (at offset +8 in the struct) is NULL or garbage, causing the comparator to crash.

Trigger Condition

The crash is reliably triggered when the **sender** (e.g. Linux running as root with `-X`) transfers a file with extended attributes in a namespace other than `user.*` (e.g. `security.*`, `trusted.*`) **to** a **non-Linux receiver** (e.g. FreeBSD) running rsync as **non-root**. The non-Linux, non-root receiver filters out these namespaces (`path 3` above), making `temp_xattr.count < count`, and then `qsort` crashes.

In the reported case the exact command on the server was:

```
rsync --server -vlHogDtpAXrxe.iLsfxCIvu --delete --numeric-ids . destination
```

Fix

Replace `count` with `temp_xattr.count` in the `qsort` call:

```
--- a/xattrs.c
+++ b/xattrs.c
@@ -860,8 +860,8 @@ void receive_xattr(int f, struct file_struct *file)
        rxa->num = num;
    }

-   if (need_sort && count > 1)
-       qsort(temp_xattr.items, count, sizeof (rsync_xa), rsync_xal_compare_names);
+   if (need_sort && temp_xattr.count > 1)
+       qsort(temp_xattr.items, temp_xattr.count, sizeof (rsync_xa), rsync_xal_compare_names);

    ndx = rsync_xal_store(&temp_xattr); /* adds item to rsync_xal_l */
```

`rsync_xal_store()` already uses `xalp->count` (i.e. `temp_xattr.count`) correctly, so only the qsort call needs to change.

Analysis Environment

- FreeBSD server (x86-64), rsync 3.4.1 as non-root user
- Linux client (aarch64), rsync sender as root
- Core file analysed with lldb: crash confirmed as `strcmp(NULL, ...)` inside `rsync_xal_compare_names`, called from `qsort` in `receive_xattr`

Safari77 added a commit that references this issue last week

`SIGSEGV in receive_xattr()` ... `71b6d0a`

laffer1 added a commit that references this issue last week

`net/rsync: fix a new cve with a workaround patch` ... `91511bf`

nixpkgs-security-tracker mentioned this 5 days ago

In rsync 3.0.1 through 3.4.1, receive_xattr relies on an untrusted … NixOS/nixpkgs#510877

tridge 3 days ago — Member

@**venglin** it was incredibly irresponsible of you to post this publicly with no attempt to contact the maintainer (me) via the published security disclosure email. Why did MITRE then publish this without asking for any evidence of responsible disclosure?
I have other commitments today, but my first cut look at this is that the "disclosure" here:
https://www.openwall.com/lists/oss-security/2026/04/16/2
is way overinflated. This is mostly a way for the client to shoot down their own connection to the fork-per-connection process.
I need to look further into the potential impact of a malicious server, but right now I'm just really annoyed at you for not following responsible disclosure practices

👍 3

venglin 3 days ago — Author

@**tridge** Well, I apologize for that. I posted on oss-security after investigating possible exploitation options and realizing that code execution was highly unlikely.

solardiz 3 days ago

@**tridge** I understand the frustration and indeed it would have been great to have your assessment and official fix ready at time of public disclosure. That said, to address your question:

> Why did MITRE then publish this without asking for any evidence of responsible disclosure?

I assume that's because responsible disclosure is not among criteria for CVE publication, nor should it be.

tridge mentioned this 4 hours ago

xattrs: fixed count in qsort #875

tridge 4 hours ago Member

fixed in #875

tridge closed this as completed 4 hours ago

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

xattrs: fixed count in qsort

RsyncProject/rsync

Participants