Incomplete Fix of Uninitialized Memory Use in kcm

Gabe Kirkpatrick - g@be-k.biz

Summary

macOS 11.3 Beta 2 contains an apparent fix for a previously reported bug in kcm ("Uninitialized Memory Use in kcm", Follow-up: 758397982). This fix is unfortunately incomplete due to insufficient checking of a return value, and can still be used to leak uninitialized heap memory.

Tested on the following software version:

```
ProductName: macOS
ProductVersion:  11.3
BuildVersion:    20E5186d
```

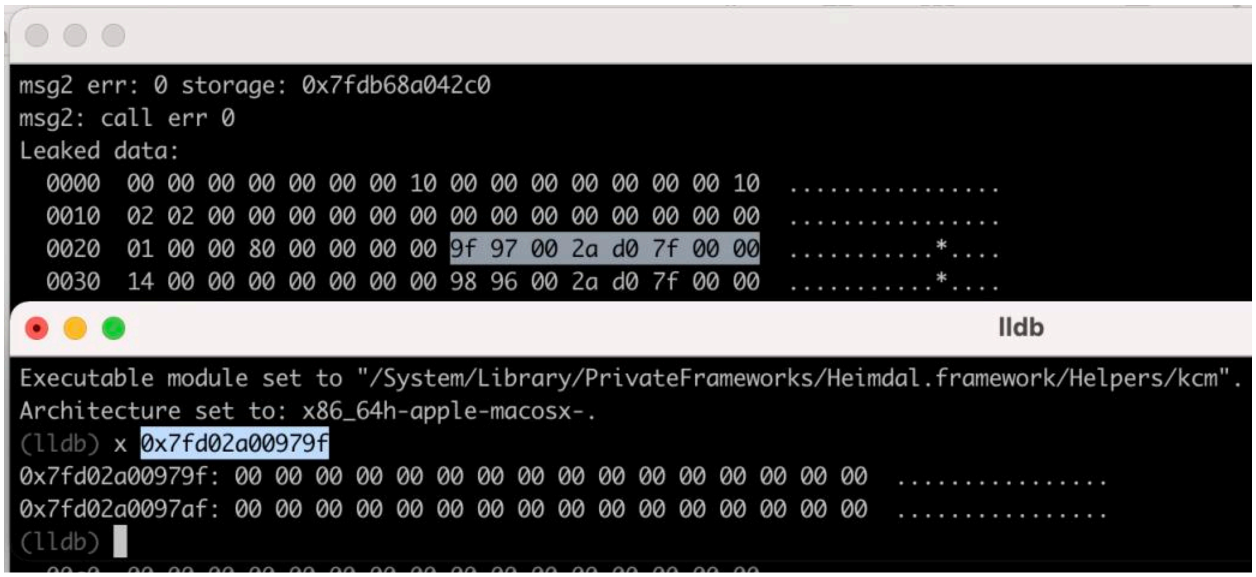
PoC Details

This PoC works by sending 2 messages to kcm: first a malformed `KCM_OP_CRED_LABEL_SET` message that causes a heap allocation to be created but not initialized, and then a `KCM_OP_CRED_LABEL_GET` message that retrieves the still uninitialized contents of the heap allocation.

Repro Steps:
Build and run the PoC by running the following from the `heap_leak` folder:

```
make
./heap_leak
```

The PoC will print hex dumps of buffers of various sizes leaked from the server. Below you can see a buffer leaked with a pointer in it validated via lldb:



```
msg2 err: 0 storage: 0x7fdb68a042c0
msg2: call err 0
Leaked data:
  0000  00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 10  ................
  0010  02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  0020  01 00 00 80 00 00 00 00 9f 97 00 2a d0 7f 00 00  ...........*....
  0030  14 00 00 00 00 00 00 00 98 96 00 2a d0 7f 00 00  ...........*....
```

lldb

```
Executable module set to "/System/Library/PrivateFrameworks/Heimdal.framework/Helpers/kcm".
Architecture set to: x86_64h-apple-macosx-.
(lldb) x 0x7fd02a00979f
0x7fd02a00979f: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x7fd02a0097af: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
(lldb)
```

Bug Details

The apparent fix for the previously reported issue was to add a call to `krb5_data_zero` on the `data` object, which succeeds in stopping the use of uninitialized stack memory and prevents the crash triggered by the previously submitted PoC. However, when the data object is initialized by calling `krb5_ret_data` to read data from the request, the return value is still not checked.

By providing a request that specifies a size for the data, but ends abruptly after without a corresponding buffer, the heap buffer allocated will remain uninitialized.

```
static krb5_error_code
kcm_op_cred_label_set(krb5_context context,
              kcm_client *client,
              kcm_operation opcode,
              krb5_storage *request,
              krb5_storage *response)
{
    struct kcm_ntlm_cred *c;
    kcmuuid_t uuid;
    krb5_data data;
    char *label = NULL;
    ssize_t sret;

    krb5_data_zero(&data); <-- Added initialization of data, prevents
crashing on free

    KCM_LOG_REQUEST(context, client, opcode);

    sret = krb5_storage_read(request, &uuid, sizeof(uuid));
    if (sret != sizeof(uuid)) {
    krb5_clear_error_message(context);
    return KRB5_CC_IO;
    }

    krb5_ret_stringz(request, &label);
    krb5_ret_data(request, &data); <-- Return value not checked,
failing mid-function results in uninitialized buffer.

…

}

KRB5_LIB_FUNCTION krb5_error_code KRB5_LIB_CALL
krb5_ret_data(krb5_storage *sp,
           krb5_data *data)
{
    int ret;
    krb5_ssize_t sret;
    int32_t size;

    ret = krb5_ret_int32(sp, &size); <— Read size
    if(ret)
        return ret;
    ret = size_too_large(sp, size);
    if (ret)
        return ret;
    ret = krb5_data_alloc (data, size); <— Buffer is allocated
with size, not initialized
    if (ret)
        return ret;
    if (size) {
        sret = sp->fetch(sp, data->data, size); <— This call can
fail if size is greater than remaining bytes, leaving buffer
uninitialized
        if(sret != size)
            return (sret < 0)? errno : sp->eof_code;
    }
    return 0;
}
```

In the PoC provided, the call to `sp->fetch` is made to fail, which results in returning from `krb5_ret_data` without initializing the buffer. Since the return value of `krb5_ret_data` is not checked, that uninitialized buffer will be used in the function to set a cred label, which can then be returned to the client via the `KCM_OP_CRED_LABEL_GET` message.

As mentioned in my previous report, this bug should be relatively easy to fix, by adding checks for the return values of `krb5_ret_stringz` and `krb5_ret_data` in the.