

[Products](#)[Services](#)[Publications](#)[Resources](#)[What's new](#)

Follow [@Openwall](#) on Twitter for new release announcements and other news

[\[<prev\]](#) [\[next>\]](#) [\[thread-next>\]](#) [\[day\]](#) [\[month\]](#) [\[year\]](#) [\[list\]](#)

Date: Thu, 30 Jun 2022 02:18:33 -0400

From: Demi Marie Obenour <demi@...isiblethingslab.com>

To: Open Source Software Security <oss-security@...ts.openwall.com>

Subject: GnuPG signature spoofing via status line injection

## # Background

After discovering that gpgv does not support `--exit-on-status-write-error`, I decided to check if it handles write errors on the status file descriptor properly. I ultimately found that while such errors are *not* handled properly, exploiting this flaw in practice would likely be very difficult and unreliable. However, in the course of this research (and entirely accidentally), I found that if a signature has a notation with a value of 8192 spaces, gpg will crash while writing the notation's value to the status FD. This turned out to be a far more severe flaw, with consequences including the ability to make a signature that will appear to be ultimately valid and made by a key with any fingerprint one wishes.

## # Prerequisites for exploitation

For an attack to be possible, the attacker must control the secret part of at least one key in the victim's keyring. The key does *not* need to be trusted, however. Depending on the calling code, the attack may work even if the key is revoked or the signature is expired. However, if the program requires that *all* signatures be valid (instead of merely *any* signature being valid), then a revoked or expired key cannot be used.

Additionally the code calling GnuPG must either not read status data until end of file, or satisfy both of the following:

- It uses a lax parser that is tolerant of invalid status lines.
- It does not treat a non-zero exit code from GnuPG as an error.

It turns out that gpgme satisfies both requirements, so programs using gpgme are vulnerable. Since gpgme is the recommended way to use GnuPG from a program, I believe that the number of applications that are vulnerable is very large.

## # Impact

If the attacker controls the secret part of any signing-capable key or subkey in the victim's keyring, they can provide a correctly-formed signature that some software, including gpgme, will believe to have a validity and signer fingerprint of the attacker's choosing. The consequences of this are highly application-dependent, but are likely to be serious. In an email client, this could allow spoofing emails, while in a system using key fingerprints for access control, this could allow for an access control bypass.

## # Solution

I recommend cherry-picking upstream commit [34c649b3601383cd11dbc76221747ec16fd68e1b](https://dev.gnupg.org/rG34c649b3601383cd11dbc76221747ec16fd68e1b), which can be found at <https://dev.gnupg.org/rG34c649b3601383cd11dbc76221747ec16fd68e1b>. Afterwards, it will be necessary to rebuild and reinstall GnuPG. No security advisory has been issued by upstream, no patch release is planned, and no CVE has (to my knowledge) been requested. Distributions will need to carry this as an out-of-tree patch until the next upstream release is made. For those using GnuPG on Windows, the only solution will be to build from source.

This does not fix the handling of write errors on the status file descriptor. However, I believe that exploiting the mishandling of such errors is not feasible in general. On the other hand, the out of bounds

read can be reliably exploited.

# Proof of concept

I have attached a public key, a revoked version of that key, and two signatures made by the key. Both signatures are of the empty string; you can pass /dev/null if the program takes a file instead. simple-exploit-sig.asc will not work if the key is revoked or expired, while revoked-exploit-sig.asc \*may\* work even if the key is revoked or expired.

# Details

## The bug

GnuPG does not provide an OpenPGP or S/MIME library. Instead, gpg, gpgv, and gpgsm all support writing machine-readable text to a user-provided file descriptor, which is set via the --status-fd command-line argument. Other programs and libraries then parse this output to extract information about what GnuPG has done.

In the case of gpg and gpgv, all status output goes through one of the functions in g10/cpr.c. The one of interest here is write\_status\_text\_and\_buffer(), of which the relevant part is reproduced below.

```
356 do
357 {
358     if (dowrap)
359     {
360         es_fprintf (statusfp, "[GNUPG:] %s ", text);
361         count = dowrap = 0;
362         if (first && string)
363         {
364             es_fputs (string, statusfp);
365             count += strlen (string);
366             /* Make sure that there is a space after the string. */
367             if (*string && string[strlen (string)-1] != ' ')
368             {
369                 es_putc (' ', statusfp);
370                 count++;
371             }
372         }
373         first = 0;
374     }
375     for (esc=0, s=buffer, n=len; n && !esc; s++, n--)
376     {
377         if (*s == '%' || *(const byte*)s <= lower_limit
378             || *(const byte*)s == 127 )
379             esc = 1;
380         if (wrap && ++count > wrap)
381         {
382             dowrap=1;
383             break;
384         }
385     }
386     if (esc)
387     {
388         s--; n++;
389     }
390     if (s != buffer)
391         es_fwrite (buffer, s-buffer, 1, statusfp);
392     if ( esc )
393     {
394         es_fprintf (statusfp, "%%02X", *(const byte*)s );
395         s++; n--;
396     }
397     buffer = s;
398     len = n;
399     if (dowrap && len)
400         es_putc ('\n', statusfp);
401 }
402 while (len);
```

When writing the data of a notation subpacket, GnuPG requests that



- 2022-06-15: I followed up stating that it may be possible to control the contents of the out-of-bounds memory and that this would make the bug much more severe.
- 2022-06-17: Werner responds stating that he has doubts as to whether this can be done easily, and noting that GPGME still needs to accept the injected data.
- 2022-06-17: I state that I am able to inject arbitrary data into the status output, and that the only reason Git is not vulnerable is because GnuPG eventually segfaults.
- 2022-06-18: I state that I can make GPGME mark a signature as "valid green" (the highest trust level) with whatever fingerprint I wish.
- 2022-06-19: Werner replies stating that he is not able to reproduce the injection of arbitrary data into the status output, though he can reproduce improper escaping.
- 2022-06-19: I state that the flaw is indeed less severe in git master.
- 2022-06-19: Via `git bisect`, I discover that 34c649b3601383cd11dbc76221747ec16fd68e1b is in fact the commit that fixed the vulnerability, and that arbitrary injection into the status line is possible on the immediately preceding commit 4dbef2addca8c76fb4953fd507bd800d2a19d3ec. I provide a reproducer.
- 2022-06-22: I request that this be marked as a security vulnerability and have a CVE assigned, and that an immediate security release be made. I note exactly what an attacker who exploits this vulnerability can do to a program relying on gpgme.
- 2022-06-29: As Werner Koch has stopped replying to my emails, and since there is still no public indication that GnuPG has a security vulnerability (despite the patch already being public), I am publicly disclosing the issue.

--  
Sincerely,  
Demi Marie Obenour (she/her/hers)  
Invisible Things Lab

**Download attachment "[key-without-revocation.asc](#)" of type "application/octet-stream" (1209 bytes)**

**View attachment "[revocation-certificate.asc](#)" of type "text/plain" (1936 bytes)**

**View attachment "[simple-exploit-sig.asc](#)" of type "text/plain" (659 bytes)**

**View attachment "[revoked-exploit-sig.asc](#)" of type "text/plain" (704 bytes)**

**Download attachment "[signature.asc](#)" of type "application/pgp-signature" (834 bytes)**

Powered by [blists](#) - [more mailing lists](#)

Please check out the [Open Source Software Security Wiki](#), which is counterpart to this [mailing list](#).

Confused about [mailing lists](#) and their use? [Read about mailing lists on Wikipedia](#) and check out these [guidelines on proper formatting of your messages](#).

