4

CVE-2022-32206: HTTP compression denial of service

Share:

TIMELINE

nyymi submitted a report to curl. May 14th (7 months ago)

Summary:

Curl does not prevent resource consumption when processing certain header types, but keeps on allocating more and more resources until the application terminates (or the system crashes, see below).

The attack vectors include (at least):

- Sending many `Transfer-Encoding` with repeated encodings such as "gzip,gzip,gzip,..."
- if `CURLOPT_ACCEPT_ENCODING` is set sending many `Content-Encoding` with repeated encodings such as "gzip,gzip,gzip,..."
- Sending many `Set-Cookie` with unique cookie names and about 4kbyte value

Steps To Reproduce:

1.Run the following HTTP server:
`perl -e 'print "HTTP/1.1 200 OK\r\n";for (my $i=0; $i < 10000000; $i++) { printf "Transfer-Encoding: " . "gzip," x 20000 . "\r\n"; }' | nc -v -l -p 9999`
2. `curl http://localhost:9999`

The application will terminate when it runs out of memory.

On macOS the app dies due to OOM:

Code 23 Bytes Wrap lines Copy Download

```
Killed: 9
$ echo $?
137
```

On linux it's the same:

```
3  137
```

When targeting Windows 11 system the system would stop responding. Once the attack script was terminated the system would not recover after 10 minutes of waiting. While it was possible to log on to the system the display would remain black. Rebooting the system was necessary to recover the system to a working state. This of course is likely due to bugs in the Windows operating system or drivers.

On other platforms nasty effects may also occur, such as causing extreme swapping or a system crash. Depending on how the system handles the application gobbling all memory it may result in collateral damage, for example when kernel attempts to release system resources by killing processes.

Impact

- Uncontrolled resource consumption
- Uncontrolled application termination
- System crash (on some platforms)

nyymi posted a comment. May 14th (7 months ago)

Naturally to be exploitable the application needs to connect to attacker provided URL or the attacker must be able to perform man in the middle attack on HTTP connection to inject the headers.

The attack can be mitigated by using `--max-time` with sufficiently short timeout.

nyymi posted a comment. May 14th (7 months ago)

Another application level mitigation would be to use `curl_global_init_mem` and keep track of total memory used and refuse allocation if extreme amounts of memory are consumed.

nyymi posted a comment. May 14th (7 months ago)

Suggested fix would be:

- Restrict `Transfer-Encoding` / `Content-Encoding` stack generation. Add some sensible maximum stack depth limit and error out if it is exceeded.
- Limit to total amount of memory that can be consumed by cookies or use some other means to control the memory consumption

Other:

Encoding based DoS seems to have been added about 5 years ago by https://github.com/curl/curl/commit/dbcced8e32b50c068ac297106f0502ee200a1ebd

Cookie based DoS is probably much older.It seems to be at least 23 years old (as long as git history).

nyymi posted a comment. May 15th (6 months ago)

I did some testing on the 64bit ARM platform:

- `Transfer-Encoding` "gzip" x 20470 results in ~140 MiB of memory consumption per header line
- `Transfer-Encoding` "zstd" x 20470 results in ~634 MiB consumption per header line

Thus If the target libcurl has been compiled with `zstd` support this is far more efficient in causing the DoS.

nyymi posted a comment. May 15th (6 months ago)

libcurl nghttp2 `HTTP/2` implementation is only affected in limited manner due to the header buffer being limited to `DYN_H2_HEADERS` (128KB) bytes maximum. This limits the amount of memory the attacker can consume. `HTTP/2` also doesn't accept `Transfer-Encoding` from the server.

In my testing on a 48GB Debian GNU/linux box with 100mbit connection (no swap) the victim application dies in about 15 seconds.

Code 60 Bytes Wrap lines Copy Download

```
Killed

real	0m14.269s
user	0m1.952s
sys	0m10.694s
```

A system with some swap will stand for longer. 32GB macOS Monterey system:

Code 53 Bytes Wrap lines Copy Download

```
Killed: 9

real 0m35.237s
```

This means that if this issue would be protected against with `CURLOPT_TIMEOUT` you'd need to set really tight timeout value.

nyymi posted a comment. May 15th (6 months ago)

While I normally would rate this issue medium, there are factors that make it worse than medium in my opinion:

- It's in the most used core functionality of libcurl
- Exploitation is quite easy
- Depending on application and/or platform it can cause a persistent denial of service requiring application restart or system power cycle to recover
- OOM Killer may kill other high memory use processes leading to collateral damage
- Secrets (decrypted private keys, AES key schedule, passwords) persistent in application memory may get dumped to core dump files. If the device doesn't have full disk encryption this may lead to leak of these secrets.

bagder curl staff posted a comment. May 16th (6 months ago)

Thank you for your report!

We will take some time and investigate your reports and get back to you with details and possible follow-up questions as soon as we can!

bagder curl staff posted a comment. May 16th (6 months ago)

> it can cause a persistent denial of service requiring application restart or system power cycle to recover

Sure, but this is the result of an active malicious/MITM server. Such a malicious server can already cause *a lot* of troubles just using ordinary means that are benign in normal cases but for each specific case can cause a lot of trouble. Like slowlaris, end-less content, poisoned data etc. This is "just another" version of attack you can do when you are a malicious server. Another reason why HTTPS is preferred - which in itself doesn't completely *remove* the risk for this kind of attack but certainly makes it much much less likely.

nyymi posted a comment. May 16th (6 months ago)

> Sure, but this is the result of an active malicious/MITM server. Such a malicious server can already cause a lot of troubles just using ordinary means that are benign in normal cases but for each specific case can cause a lot of trouble. Like slowlaris, end-less content,

Yes, it is a similar case, but not quite the same.

At least with those other scenarios you have a probability to defend yourself in a reasonable way:

- Check excessive amount of data arriving via `CURLOPT_WRITEFUNCTION`
- Have sensible `CURLOPT_TIMEOUT`
- Verify content signature

I think being forced to set timeout of couple of seconds or having to defend against runaway `malloc` is not something application should have to expect to defend against.

bagder curl staff posted a comment. May 16th (6 months ago)

I agree that this is silly to allow and we should restrict it, but really cannot agree that this is a "high" severity. The instant this card is played out, you've "shown your hand" so to speak. It's a one-off opportunity to do a DOS against a libcurl-based client, but in a very blunt and loud way.

nyymi posted a comment. Updated May 16th (6 months ago)

I knew suggesting high severity is pushing it and that's why I listed the points that would defend that argument.

However, after some more consideration some of those points actually are due to other vulnerabilities (such as platform not coping with OOM situation, or failing to protect the filesystem). Those are not due to this vulnerability and hence are irrelevant when considering the rating.

So yeah I agree, this is not high severity.

bagder curl staff posted a comment. May 16th (6 months ago)

Here's a first shot at a patch that (arbitrarily) limits the chain "depth" to 5 and bails out if exceeded. I have a separate test that verifies it too.

1 attachment:

F1731757: 0001-content_encoding-return-error-on-too-many-compressio.patch

bagder curl staff updated the severity from High to Medium. May 16th (6 months ago) nyymi posted a comment. May 18th (6 months ago)

Patch is looking good to me.

`Set-Cookie:` can still be used to drain memory. This attack is extremely slow to pull off in comparison, however.

bagder curl staff updated CVE reference to **CVE-2022-32206**. Jun 9th (6 months ago)

Jun 9th (6 months ago)
bagder curl staff
changed the report title from **Denial of service via HTTP headers** to **CVE-2022-32206: HTTP compression denial of service**.

bagder curl staff posted a comment. Jun 9th (6 months ago)

First advisory draft

1 attachment:
F1762920: CVE-2022-32206.md

nyymi posted a comment. Jun 9th (6 months ago)

Looking good to me.

bagder curl staff posted a comment. Jun 18th (5 months ago)

Since I accidentally but stupidly committed a draft version of the advisory for this issue to the master branch of the curl web site git repository a few days ago (only to realize it today). We have decided to speed up the release process and instead do it on June 27. I'm very sorry for this.

bagder curl staff closed the report and changed the status to ○ **Resolved**. Jun 27th (5 months ago)

Thanks for your work. The actual monetary reward part for this issue is managed by the Internet Bug Bounty so the curl project itself therefor sets the reward amount to **zero USD**. If you haven't already, please submit your reward request to them and refer back to this issue.

curl has decided that this report is not eligible for a bounty. Jun 27th (5 months ago)

This report has been disclosed. Jun 27th (5 months ago)