Introduction

CloudFoundry is a VMWare owned cloud that makes it easier for developers to lower the cost and complexity of configuring infrastructure for their web apps. It allows them to build web apps with APIs that abstract the underlying infrastructure (such as AWS or GCP). A vulnerability in the gorouter package - the component responsible for directing web requests to specific CloudFoundry app instances, allowed attackers to push an undesirable response that ended up in a front end cache and was served to other users causing denial of service. The patch for this vulnerability was released on February 24, 2020 and it was given the CVE ID CVE-2020-5401 and its severity was classified as 'Medium'. I hereby allow University of Tartu to exhibit this paper publicly until 2025.
Vulnerability description

Cache poisoning DoS or CPDoS (short for cache poisoning denial-of-service) is a type of attack where malicious clients send invalid headers, causing caching layers to reject subsequent legitimate clients trying to access the app. This attack affected Cloud Foundry gorouter - “a routing tool which routes the traffic that is coming into Cloud Foundry to the appropriate component, whether the request comes from an operator addressing the Cloud Controller or from an application user accessing an app running on a Diego Cell”[5] (a “self-healing” container management system that is designed to keep the correct number of instances running in Diego cells, thus avoid network failures and crashes).

Nathan Davidson, a senior developer at digital transformation agency, who was the first to report the issue said in his blog that a cache would typically not allow this to happen, provided that the technique used to cause the undesirable response relies on some input given in a standard or anticipated way (query string, common HTTP headers). Such input would instead be used by the cache to generate the key for the cached copy, and other requests would receive the cached response back only if they had the same input. Althought this works in theory, cloud systems often provide advanced users and systemadministrators with extra abilities via custom HTTP headers, for example added diagnostic information or control over routing, which is exactly where the CPDoS vulnerability for CloudFoundry could be found - in its custom X-CF-APP-INSTANCE request header. This specific header allowed the requester(s) to target a specific app and instance with the format of APP-GUID:INSTANCE-NUMBER, such as X-CF-APP-INSTANCE: aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1 (when a valid value guid held by a running app is given, and the instance number is pointing to a valid running instance, then the request will be served by that instance).
Proof of concept

The author of the article published on Nathan Davison`s blog shows an example where he uses App Guid as bad value for the header to show what kind of a response it would give (Fig 1).

![Figure 1 A response from a bad header value](image)

The author pointed out that the problem with the response in Fig 1 is that we are given back a HTTP 404 error, which is cacheable by default according to the RFC, and in fact will be cached by default when using common web cache frontends like Cloudflare and CloudFront. The author pointed out that Cache-Control: public, max-age=3600 only seemed to compound the issue, although his testing on CloudFront seemed to indicate that it was not critical for the 404 error to be cached.

In this next example (Fig 2) the author demonstrates how he could cause widespread DoS with only a handful of requests. He pointed out that this asset was behind CloudFront, which means that a resource like https://www.nexcess.net/web-tools/dns-checker/ could be used to first collect the IP addresses of the target's edge caches, and then for each edge cache, run a script like the following:
This script would poison the given cache for the resource https://TARGET_HOST/?cb=xxx, where the edge cache IP and the target hostname would then be placed in the raw HTTP request and openssl command arguments where relevant. What makes this handful of requests powerful is that they issue the poisoning every 10 seconds, to ensure that the DoS is maintained.
“In this particular instance, a specific header with a specific type of value created an error response when targeting a specific cloud platform, which was cacheable and likely would have been cached by most conforming web caching layers” [2]. The error raised can and will be served to other users, effectively blocking their access to the target causing the so called CPDoS. In the case of CVE-2020-5401, there are several fixes that would work - the caching layer could be configured to no longer cache 404 errors, or the X-CF-APP-INSTANCE header could be made to determine the cache key. Also, the cloud provider could instead return a non-cacheable response - the exact thing Pivotal (a software and services company based that provided cloud platform hosting and consulting services that is now apart of VMware) did in fixing this vulnerability in the gorouter component, choosing a HTTP 400 response instead (which, according to the CPDoS.org research may still be cached, but only with a poorly configured caching layer) (Fig 3).
Figure 3 Cloud providers non-cacheable response to error code [4]
References

1. CVE - CVE-2020-5401 [Internet]. [cited 2020 May 01]. Available from: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5401


5. Gorouter discription [Internet]. [cited 2020 May 01]. Available from: https://docs.cloudfoundry.org/concepts/architecture/router.html

6. CPDoS.org research [Internet]. [cited 2020 May 01]. Available from: CPDoS.org