

Forum:[Hack Forums](#)

Thread: [Study on Command Injection in GitLab Community and Enterprise Edition 11.11](#)

Bri4n0ps - August 11, 2020, 3:13 pm

Study on Gitlab's SSRF vulnerability allowing RCE by authenticated users (CVE-2019-12430)

Server Side Request Forgery And Cross Application Scripting

SSRF is Server Side Request Forgery. This issue occurs when an application sends requests over the internet to an attacker-controlled location. Attackers have many uses for such functionality, including:

Denial of Service attacks

A reflected DoS vulnerability is when an attacker can send a small request that results in a large request being sent to an attacker-controlled endpoint.

Access to internal networks

The attacker can send web requests to other servers running within the same network. This is especially dangerous if the result of the request is returned to the attacker. This could expose sensitive data

Access to internal networks includes the machine the attack is being run on - this could allow the attacker to perform port scans, or even talk to other services running on the local machine that aren't internet accessible

Reflected exploitation

If the attacker knows of an exploit for another service, and has control over most of the request, he may be able to send an exploit through the SSRF, preventing his IP from appearing in logs on the affected service.

CAS

CAS stands for Cross Application Scripting. Cross Application scripting allows one application to communicate with another.

Cross Application Scripting is when an attacker tricks an application into talking to another application. For example, you can talk HTTP to a memcached server. The memcached server will discard all invalid commands, and valid commands will be run. For example, a request using the custom HTTP method "stats sizes" will look like the following:

```
stats sizes / HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost:11212
Accept: /
```

The first line is a valid memcached command. The following lines will be ignored. Even if the following lines weren't ignored, it makes no difference since the attack has already completed. This request can be sent to a memcached server with a curl request: ``curl -X "stats sizes" localhost:11211``.

Gitlab vulnerable to SSRF & CAS

A vulnerability in gitlab was publicised earlier this year (2019) by "@nyangawa of Chaitin Tech". This vulnerability affects the "import project by repo URL" functionality of gitlab. The import project by repo URL is by design, vulnerable to SSRF. Gitlab attempts to mitigate this by blacklisting certain URL patterns including ``127.0.0.1``, `localhost`, and others. An IPv6 format was not blacklisted though, and requests to ``[0:0:0:0:ffff:127.0.0.1]`` are permitted. Gitlab uses redis rather than memcached, so an attacker would specify the port ``6379``.

The developers of redis are aware of attacks like this. Therefore, if redis sees the ``Host:`` header, it

recognises that the request is an HTTP request and drops the connection. Fortunately for attackers, gitlab supports the `git://` protocol, which is not detected.

Now that attackers are capable of inserting arbitrary content into the redis server, it is trivial for attackers to run any code on the gitlab server. One such method is to queue a `GitlabShellWorker` task with the attacker-controlled commands.

NotesExploitation walkthrough: <https://github.com/jas502n/gitlab-SSRF-redis-RCE>

Gitlab issue report: <https://gitlab.com/gitlab-org/gitlab-ce/issues/41293>

Gitlab public release communication: <https://about.gitlab.com/2019/06/03/security-release-gitlab-11-dot-11-dot-1-released/>

This vulnerability was introduced in GitLab 8.8 via https://gitlab.com/gitlab-org/gitlab-ce/merge_requests/3940

This vulnerability has been assigned CVE ID: CVE-2019-12430