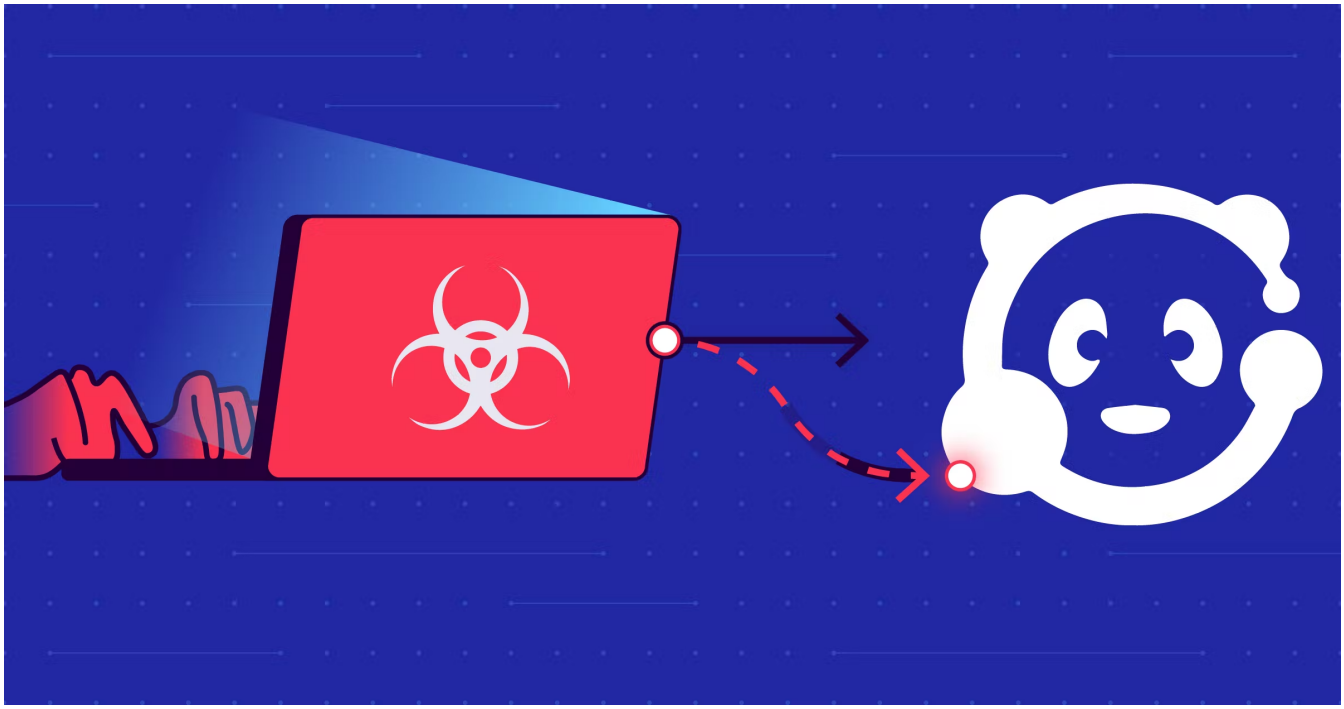
Securing Developer Tools: OneDev Remote Code Execution

BY PAUL GERSTE | SEPTEMBER 20, 2022

Security



OneDev is a self-hosted Git server that comes with a lot of development-oriented features such as CI/CD, code search, and static analysis integration. With almost 10,000 stars on GitHub, it is gaining popularity and becoming an open-source and low-maintenance alternative to GitHub, GitLab, and Bitbucket.

Source code becomes an increasingly important asset of most companies, making it crucial to protect it from being stolen or, even worse, altered by malicious actors. This is why we decided to look at the very services that are the most interesting targets of these threat actors: source code hosting platforms.

In this article, we describe the vulnerabilities we found in OneDev that could be used by attackers to take over vulnerable instances. We will first look at their impact, then dive into the technical details, and finally, we'll discuss how the maintainers fixed the issues and how you can prevent them in your own code.

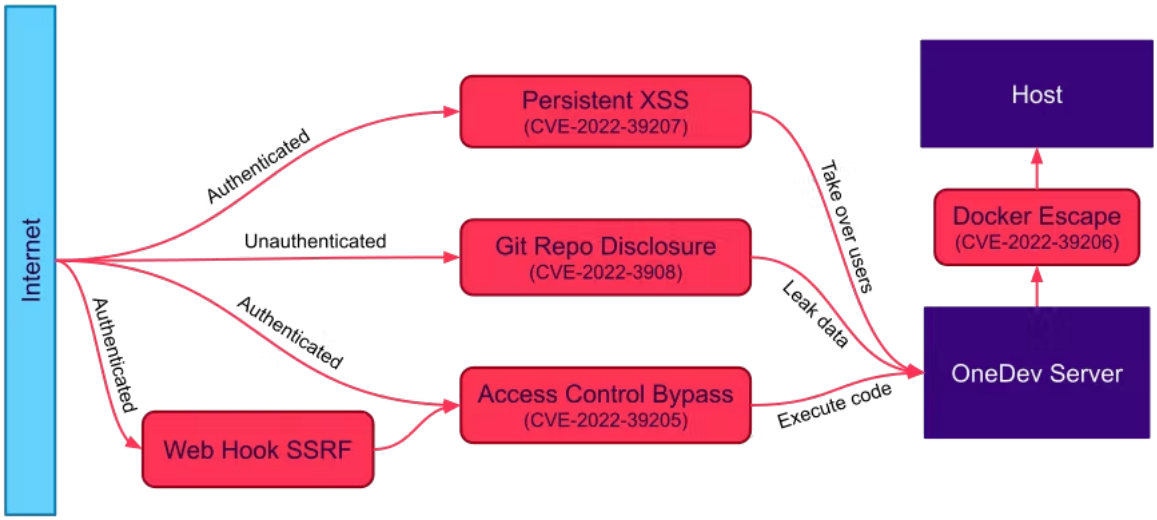
Impact

We discovered the following vulnerabilities in OneDev 7.2.2

:05)

- Git Repository Disclosure (CVE-2022-39208)

Some of them can be combined by attackers to execute arbitrary commands on vulnerable OneDev instances. This would allow them to steal or manipulate source code, build artifacts, and launch further attacks against internal infrastructure. Most of the vulnerabilities require authentication as a regular user, except for CVE-2022-39208 which can be exploited without any authentication at all. The following is an overview of how the vulnerabilities can be exploited:



We strongly recommend updating to at least version 7.3.0 to benefit from the maintainer's fixes.

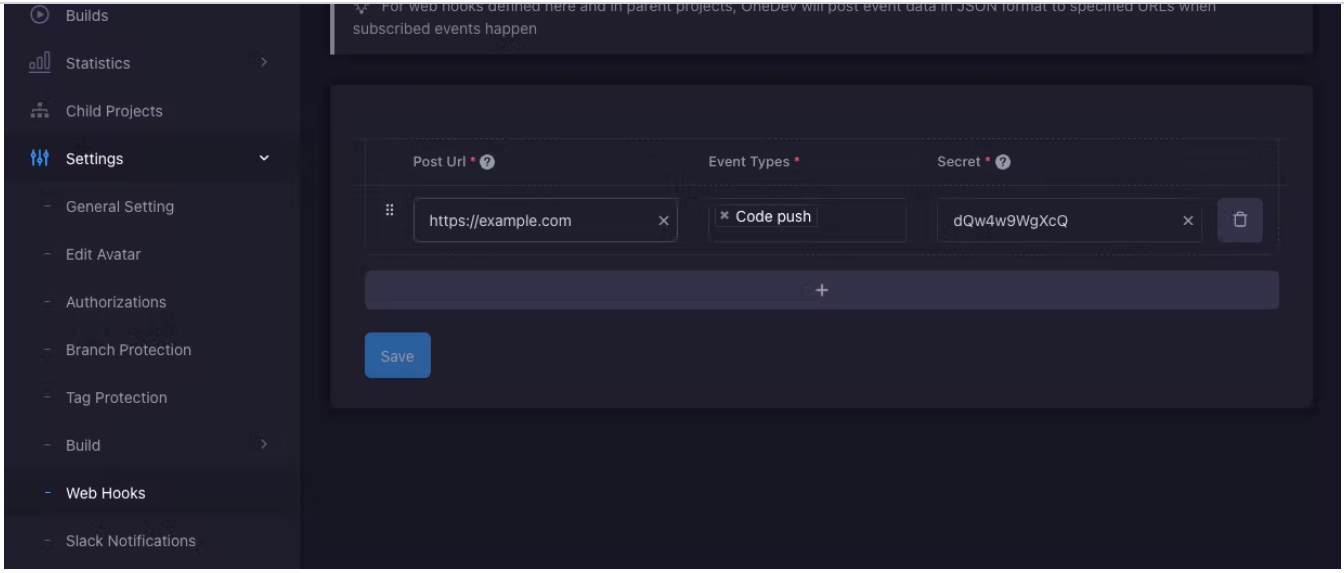
Since the OneDev maintainers are using the OneDev instance at code.onedev.io to develop the project, attackers could have hijacked the project's infrastructure to plant malicious code into OneDev itself. This would have allowed them to compromise users of OneDev without having to directly attack them. Such a software supply chain attack is very stealthy and has happened before, for example in the Codecov and SolarWinds incidents, and could have also happened in the open-source world.

Technical Details

In this section, we will look at the technical details of three vulnerabilities that can be combined into a chain. First, we will look at a Server-Side Request Forgery issue, then we will see how it can be used to achieve Remote Code Execution, and finally, we will show how attackers could have used a Docker misconfiguration to escape from a container to the host system.

Server-Side Request Forgery

A common feature among source code hosting platforms is the ability to add webhooks that will be triggered by certain events. An example use case for this is to trigger a deployment on every push to a repository's main branch. This functionality requires the user to set a URL and an event type, telling the server to send an HTTP request to that URL when events of that type occur. The request usually contains the event data in its body. This is what the feature looks like for OneDev:



In the case of OneDev, the target URL of a webhook is not restricted at all (besides having to be a syntactically valid URL). This allows users to force the server to make HTTP requests to internal targets, including the server itself, by setting the webhook URL to an internal IP address and triggering the defined event.

This is known as Server-Side Request Forgery (SSRF), which can be used to talk to internal systems and trigger further vulnerabilities in them. In this case, it is a limited SSRF: the user does not control the HTTP method or any HTTP headers, they only control some parts of the body, and they can't see the HTTP response. This limits the user-controlled inputs that can be passed to an internal server to the URL, i.e. its path and query parameters.

The OneDev maintainers did not request a CVE for this vulnerability because they see it as intended behavior. They expect OneDev setups to be so diverse that blocking certain IP ranges could lead to issues for some users. The next section will cover how it was possible to use this SSRF to reach an internal endpoint, bypass its protection, and execute arbitrary code on the server.

Access Control Bypass leading to Remote Code Execution (CVE-2022-39205)

To validate push events, OneDev uses Git commands under the hood. This is implemented with a callback endpoint that will be called whenever a push event happens. This endpoint is only supposed to be called from localhost, so the following check was introduced:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) {
    String clientIp = request.getHeader("X-Forwarded-For");
    if (clientIp == null) clientIp = request.getRemoteAddr();

    if (!InetAddress.getByName(clientIp).isLoopbackAddress()) {
        response.sendError(HttpServletResponse.SC_FORBIDDEN,
                "Git hook callbacks can only be accessed from localhost.");
        return;
    }
    // ...
}
```

The endpoint checks if the client's IP address is the loopback address, e.g. `127.0.0.1`, but it blindly trusts the `X-Forwarded-` e deployed in front of the OneDev instance.

After that initial check, the callback endpoint will read some parameters from the request. This includes a project ID, a user ID, and commit hashes. These values are then used to check certain conditions based on the project's settings and the user's permissions. Finally, a Git command will be executed based on the previous checks. OneDev invokes this command in a safe way that prevents Command Injection and Argument Injection vulnerabilities.

However, the callback also takes environment variables from the request's query parameters. They are then passed to the Git command invocation, which is very unsafe because the behavior of programs can be influenced by specifying certain environment variables.

Making Git write files

Modern versions of Git support setting any config value via environment variables, which would be enough to execute arbitrary commands. This can be achieved by setting the `GIT_CONFIG_COUNT`, `GIT_CONFIG_KEY_*`, and `GIT_CONFIG_VALUE_*` variables. However, the Git version that was shipped in the default Docker container of OneDev was older, so this option was not feasible.

Another technique to control program behavior via environment variables is to use `LD_PRELOAD` which allows specifying a shared library that can override certain functions used by a binary. This would require first uploading such a shared library to the filesystem, making it also not feasible in this scenario without an additional vulnerability.

We noticed that OneDev uses Git hooks to validate some Git events and that the hooks were simple shell scripts. These files were also writable by the user that the OneDev server runs as, so it would be possible to overwrite them with malicious commands. We then checked the Git documentation for any environment variable that would enable writing to an arbitrary file. We found `GIT_TRACE_SETUP`, which is used to activate the output of verbose debugging information, either to `STDERR` or to a file that can be specified in the variable's value. Its output looks like this:

```
$ env GIT_TRACE_SETUP=/tmp/trace.txt git status
On branch main
No commits yet
nothing to commit (create/copy files and use "git add" to track)
$ cat /tmp/trace.txt
15:40:18.109099 trace.c:311             setup: git_dir: .git
15:40:18.109933 trace.c:312             setup: git_common_dir: .git
15:40:18.109940 trace.c:313             setup: worktree: /tmp/git-trace
15:40:18.109947 trace.c:314             setup: cwd: /tmp/git-trace
15:40:18.109953 trace.c:315             setup: prefix: (null)
```

This approach looked promising, but it had some drawbacks:

- First, only some parts of the output can be controlled via other environment variables;
- Second, Git would only append to existing files, not overwrite them. This is especially problematic here, because the Git hook scripts all ended with a call to `exit` on their last line, ending the execution of the script regardless of what comes in the next lines;
- The last straw that remained here was that the scripts had no linebreak at the very end, which meant that we could execute commands in a subshell if we could inject them in the first line of the trace output.

Controlling the output

`_DIR` environment variable. It has to point to a
ill be written. It is also not possible to insert
his means the attacker needs to create a

Since the attacker in our scenario already has access to a project on the OneDev server, they can do this by creating and pushing a new branch. Branches are stored in a Git repository as `.git/refs/heads/<branchname>`. Branch names can include slashes, so this can result in multiple nested directories, but the last "segment" of a Git branch name will always be a file that holds the branch's most recent commit hash. So to create a branch that suits the needs of the exploit, the attacker has to include at least one slash after the subshell command to make sure the command will be in a directory name, not in a file name.

Putting it all together, the attacker would create and push such a branch and then use `GIT_DIR` and `GIT_TRACE_SETUP` to append to a Git hook. This is what such a hook script will look like afterward:

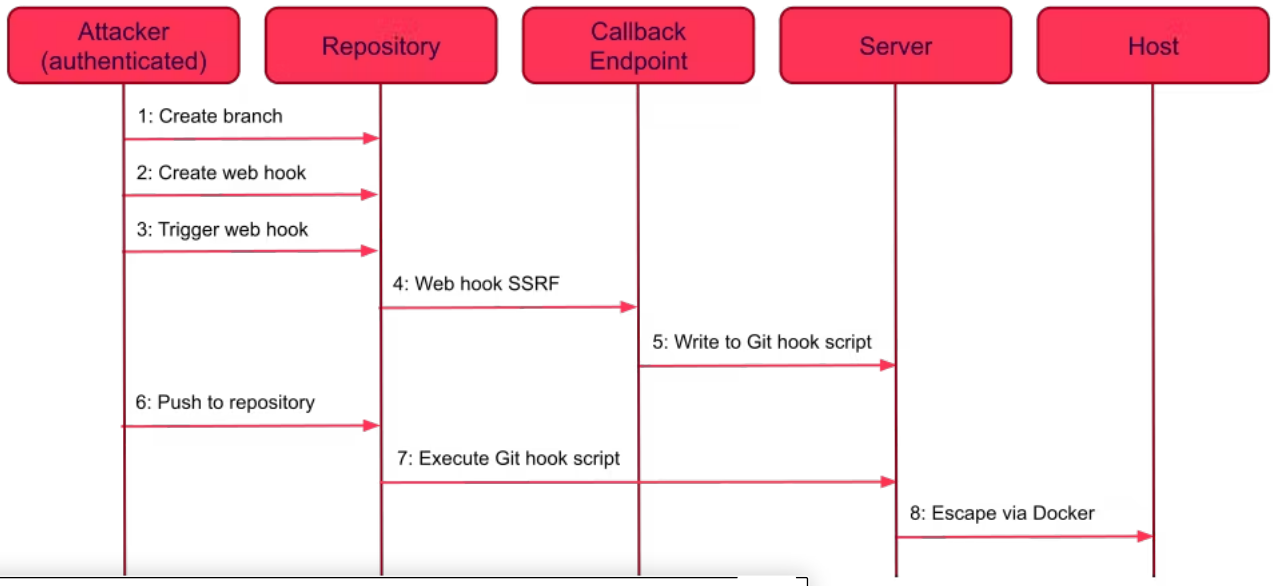
```
[...]
exit $returnCode15:50:06.391600 trace.c:311             setup: git_dir: .git/refs/heads/$(id>/tmp/pwned)/../..
15:50:06.393284 trace.c:312             setup: git_common_dir: .git/refs/heads/$(id>/tmp/pwned)/../../../../..
15:50:06.393318 trace.c:313             setup: worktree: /private/tmp/git-trace
15:50:06.393336 trace.c:314             setup: cwd: /private/tmp/git-trace
15:50:06.393352 trace.c:315             setup: prefix: (null)
```

Container Escape (CVE-2022-39206)

OneDev recommends starting the server with access to a Docker socket to enable its Docker-based CI/CD pipelines. If the OneDev server is hosted in a Docker container itself, this is recommended to be done by mounting the host's Docker socket into the OneDev container.

This is dangerous because it allows breaking out of the container and executing commands as root on the host system unless Docker runs in rootless mode. OneDev also mounts the host's Docker socket into certain CI/CD pipeline containers, making it even possible for users with the appropriate permissions to perform a Docker escape by simply defining a malicious CI configuration.

The following summarizes the whole exploit chain:



We think it would be a good middle ground to have an admin setting that can control such a blocklist and give it a default value that blocks requests to all local and private IP ranges. If you have code in your own code base that makes requests based on user-controlled values, make sure to properly validate each request. The most important thing here is to validate the protocol, hostname, and port. Modifying them can allow attackers to talk to internal services that are otherwise not exposed. Such services are usually less secured and attackers could make requests to them in order to hijack them. Keep in mind that the validation has to be done at the right moment to be effective, as explained in our recent blog post about a similar issue in WordPress.

Access Control Bypass

For the Access Control Bypass vulnerability, the maintainers introduced a new system of authentication tokens that have to be sent with each request and are validated before any actions are performed. This is a very solid solution because even SSRF vulnerabilities can't be used to trigger the endpoint as the attacker would have to know a valid token first.

As a rule of thumb, never blindly trust HTTP headers like `X-Forwarded-For`, unless you made really sure that they cannot be set by an attacker. A good recommendation is to make trusting these headers opt-in instead of opt-out, reducing the likelihood of accidentally trusting them.

Container Escape

The Container Escape vulnerability was only possible because the host's Docker socket was exposed in environments where users with low privileges could use it. This is an unsafe pattern and should be avoided. Make sure that only trusted users (e.g. admins) can access the socket and that all parameters that are included in requests to it are properly validated. There are also third-party proxies, such as docker-socket-proxy, that can restrict the Docker interface to a safer subset of possible requests.

Timeline

Date	Action
2022-05-30	We report all issues to the OneDev maintainers
2022-05-30	The OneDev maintainers confirm the issues
2022-05-31	The OneDev maintainers fix the issues (except the SSRF), release a new version (7.3.0) and rebuild the cod e.onedev.io instance from scratch to ensure that it is not backdoored
2022-09-13	The CVE IDs are assigned

Summary

In this article, we presented several high-impact vulnerabilities we found in OneDev, an open-source Git server. The issues have a critical impact but require a low-privileged user account to be exploited. While this reduces the impact for many instances, it was possible for attackers to take over the infrastructure of the OneDev itself, which could have resulted in a widespread impact for many users of the project.

the maintainers chose to tackle them, as well

Related Blog Posts

- Agent 008: Chaining Vulnerabilities to Compromise GoCD
- WordPress Core - Unauthenticated Blind SSRF
- Securing Developer Tools: Argument Injection in Visual Studio Code
- PHP Supply Chain Attack on Composer

PAUL GERSTE
Vulnerability Researcher