Environment variable disclosure via Response-Derived DSL Expressions

Moderate **ehsandeep** published **GHSA-jm34-66cf-qpvr** 2 days ago

Package

github.com/projectdiscovery/nuclei (Go)

Affected versions	Patched versions
>= 3.0.0, < 3.8.0	3.8.0

Description

A vulnerability in Nuclei's expression evaluation engine makes it possible for a malicious target server to inject and execute supported DSL expressions. This happens when HTTP response data containing helper/function syntax gets reused by multi-step templates. If the `-env-vars` / `-ev` option is explicitly enabled, this can expose host environment variables. That option is off by default, so standard configurations are not affected by the information disclosure risk.

Affected Component

The issue lives in `expressions.Evaluate()` at `pkg/protocols/common/expressions/` and in the unresolved-variable validation path (`hasLiteralsOnly()`).

Description

`expressions.Evaluate()` replaces placeholders first, then scans the substituted output for expressions. Because of this two-pass approach, response-derived values (including extractor output and response body content) can be reinterpreted as DSL/helper syntax on the second pass.

When `-env-vars` (`-ev`) is enabled, environment variables get merged into the template variable map. A malicious target can return response data containing expressions like `{{env_var_name}}` which, when reused in a subsequent template request, resolve to actual environment variable values. This can expose sensitive host data like API keys, credentials, and tokens.

Without `-ev` enabled (the default), injected DSL expressions may still trigger helper functions such as `{{md5("test")}}` , but this has no meaningful security impact beyond unexpected behavior.

There is also a separate issue in `hasLiteralsOnly()`: it was evaluating helper expressions while deciding whether `{{...}}` contained unresolved variables, which caused validation logic to run side-effectful helpers even when the final request kept the value as a literal.

> **Note**
>
> The `-env-vars` / `-ev` option is off by default. Users who have not explicitly turned it on are not affected by the information disclosure aspect of this vulnerability.

Affected Users

- **CLI users** running multi-step templates (with extractors or flow-based request chaining) that reuse response-derived values against untrusted or attacker-controlled targets, with the `-ev` flag enabled.
- **SDK users** who have integrated Nuclei into platforms where `EnvironmentVariables` is set to `true` and scan targets are not fully trusted.

Patches

- The vulnerability is fixed in Nuclei v3.8.0. Upgrading to this version is strongly recommended.
- Relevant fix references: #7221, #7321.

Mitigation

Upgrade to Nuclei v3.8.0. The updated evaluation logic now collects expressions from the original template text before placeholder substitution and only evaluates those template-authored expressions.

If you have `-ev` enabled, disable it when scanning untrusted targets to avoid environment variable disclosure.

Workarounds

If upgrading is not an option right now, make sure `-env-vars` / `-ev` is not enabled when running multi-step templates against untrusted targets.

Acknowledgments

Thanks to **@gnuletik** for reporting this issue through responsible disclosure via security@projectdiscovery.io

Severity

Moderate 5.3 / 10

CVSS v3 base metrics

Attack vector	Network

Attack complexity	High
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:N/A:N

CVE ID

No known CVE

Weaknesses

▶ CWE-94

Credits

gnuletik Finder