Remote Code Execution via YAML Deserialization in Agent Definition Loading

Critical **MervinPraison** published **GHSA-32vr-5gcf-3pw2** yesterday

Package

praisonai (pip)

Affected versions

<= 4.5.114

Patched versions

>= 4.5.115

Description

Summary

The `AgentService.loadAgentFromFile` method uses the `js-yaml` library to parse YAML files without disabling dangerous tags (such as `!!js/function` and `!!js/undefined`). This allows an attacker to craft a malicious YAML file that, when parsed, executes arbitrary JavaScript code. An attacker can exploit this vulnerability by uploading a malicious agent definition file via the API endpoint, leading to remote code execution (RCE) on the server.

Details

The vulnerability exists in the YAML deserialization process. The `js-yaml` library's `load` function is used without specifying a safe schema (e.g., `JSON_SCHEMA` or `DEFAULT_SAFE_SCHEMA`). This enables the parsing of JavaScript functions and other dangerous types. When a malicious YAML file containing a `!!js/function` tag is parsed, the function is evaluated, leading to arbitrary code execution.

The vulnerable code is located in `src/agents/agent.service.ts` at line 55.

PoC

An attacker can create a malicious agent YAML file with the following content:

```
!!js/function >
  function(){ require('child_process').execSync('touch /tmp/pwned') }
```

Then, upload this file as an agent definition via the API endpoint that uses `AgentService.loadAgentFromFile`. When the agent is loaded (either during startup or via an API call that triggers loading), the payload will execute the command `touch /tmp/pwned`, demonstrating arbitrary code execution.

Impact

This vulnerability allows an unauthenticated attacker (if the API endpoint is unprotected) or an authenticated attacker with the ability to upload agent definitions to execute arbitrary code on the server. This can lead to complete compromise of the server, data theft, or further network penetration.

Recommended Fix

Replace the unsafe `load` method with a safe alternative. Specifically, use the `load` method with a safe schema, such as `JSON_SCHEMA` or `DEFAULT_SAFE_SCHEMA`. For example:

```
import yaml from 'js-yaml';
import { JSON_SCHEMA } from 'js-yaml';

// In the loadAgentFromFile method
const agent = yaml.load(fileContent, { schema: JSON_SCHEMA });
```

Alternatively, if the application requires only a subset of YAML features, consider using the `safeLoad` method from an older version (though note it was deprecated). The key is to avoid loading tags that can execute code.

Additionally, validate and sanitize all user input, especially file uploads. Ensure that agent definition files are only uploaded by trusted users and consider storing them in a secure location with proper access controls.

Severity

Critical 9.8 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None

Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

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

CVE ID

CVE-2026-39890

Weaknesses

▶ CWE-502