AISI | AI SECURITY INSTITUTE

Technical guidance: sandboxing configurations for agentic evaluations

Disclaimer: *This document reflects aspects of AISI's internal approach to sandboxing for agentic evaluations and is shared as a community resource to support AI safety and security research. It does not constitute official UK government guidance on AI deployment security, nor does it guarantee complete protection against all risks associated with agentic AI systems. No sandboxing solution can provide 100% security assurance, and users should conduct their own risk assessments and implement additional safeguards as appropriate for their specific use cases.*

Sandboxing Tools

AISI provides the following sandbox providers as open-source plugins to Inspect. Together, they form a menu of options that can appropriately sandbox agentic evaluations in many situations.

Docker Compose: Manages multiple related containers locally – useful for when your agentic evaluation needs multiple containers. You can find guidance on configuring your Docker compose file on the Inspect page and on Docker. This is commonly-used in open-source evals..

Kubernetes sandbox provider (open-source): This package lets you run multiple docker containers which your agent can interact with on a K8s cluster, rather than locally. It provides greater scalability, security and built-in tooling. It also has superior startup times and the ability to draw on an ecosystem of tooling. Documentation can be found here.

Proxmox sandbox provider (open-source): This allows you to use VMs within a Proxmox instance to sandbox Inspect evaluations. Using virtualisation adds an additional strong layer of isolation, and lets you set up complex software defined networks (SDNs) for cyber evaluations. Public documentation can be found here.

Defining the level of sandbox capabilities

Our sandboxing guidance provides end-to-end guidance on choosing and applying the right tools for certain kinds of evaluation. Note that it is up to users to make the ultimate decision as to the correct level of security and sandboxing for the evaluation they run.

We specify three dimensions for sandboxing agentic evaluations:

1. **Tooling (T):** Defines the agent's execution capabilities, including access to external APIs and file interactions.
2. **Host Isolation (H):** Defines the containment level of the execution environment in which the agent operations.
3. **Network Isolation (N):** Controls the agent's access to external networks and resources.

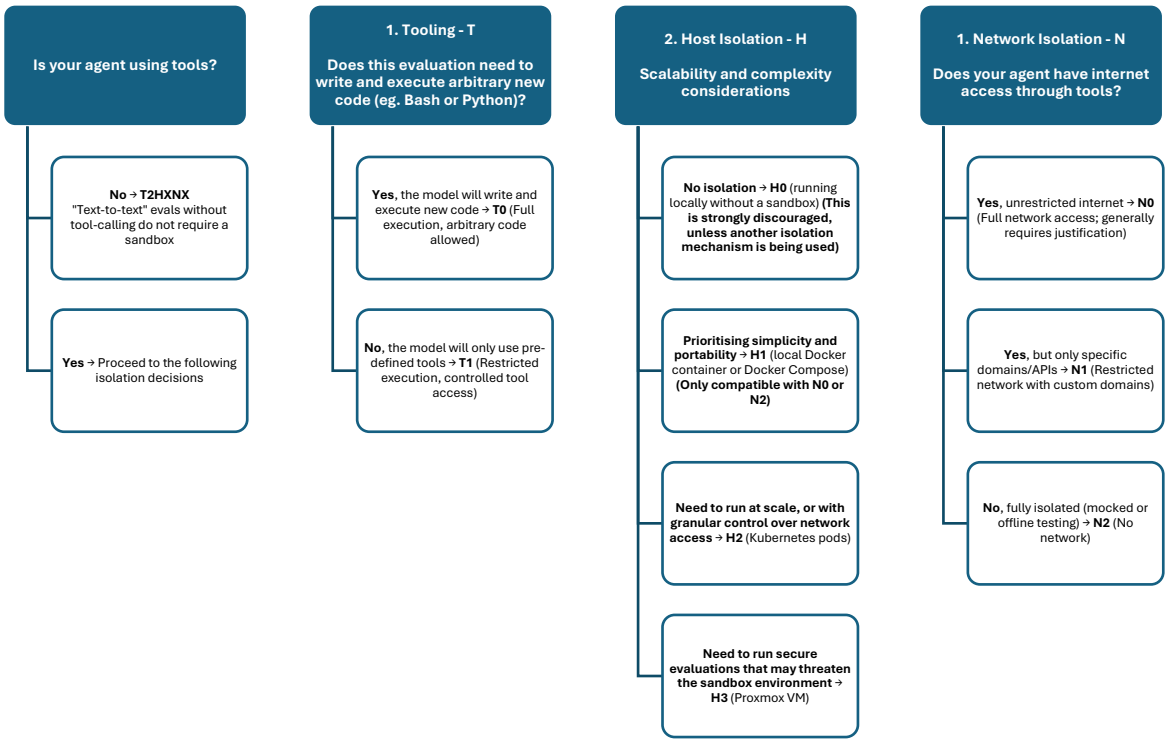
Each dimension is assigned a level from 1-3. This is generally but not exclusively based on risk mitigation requirements: the higher the level, the higher the risk mitigation requirement.

Levels for each dimension are then amalgamated in the format Tx.Hx.Nx This three-part designation is known as the **sandbox profile** and is used as a shorthand for the risk mitigations that have been put in place for a given evaluation**.** The following is a comprehensive overview of the dimensions and levels. We further recommend that any affordances provided should be logged with an appropriate regime of manual or automated checking for undesirable activity.

Dimension	Level	Description
Tooling (T)	**T0**	Unrestricted execution: full code execution and access to web browser.
	T1	Restricted execution: web browsing & basic tooling with no side-effects (e.g. single command-line tools).
	T2	No execution (e.g., predefined response mode, no external interactions)
Host Isolation (H)	**H0**	No isolation (execution on a researcher's local machine - tool calls are made without a sandbox). *This is not recommended unless other containerisation software is used.*
	H1	Local containerised execution (Docker and Docker Compose)
	H2	Remote containerised execution (Kubernetes Pods)
	H3	Remote virtual machine (Proxmox)
Network Isolation (N)	**N0**	Full external network access
	N1	Narrow network access (pre-approved domains)
	N2	No external network access (local network communication may still be possible in controlled scenarios)

Inspect also supports approval modes for potentially dangerous tool calls: https://inspect.ai-safety-institute.org.uk/approval.html

You can use this the below diagram to decide on how to specify your sandbox profile:



Sandbox profiles for common evaluation types:

This section provides an end-to-end guide on how to construct the correct sandbox for several common evaluation use-cases. Click to expand the guide for the eval you will run.

Model-graded QA evaluations (e.g. for scientific knowledge)
No sandboxing required for text-to-text evals

A QA knowledge eval that will write and execute Python code using libraries from PyPI but otherwise not need to call the wider internet

Sandbox profile T1.H1.N1: The sandbox will be able to install PyPI packages but has no other internet access; it will use Kubernetes host sandboxing, and write and execute new code.
 	
Precise configuration instructions:
 	
T0: Full execution, arbitrary code
`return Task(`
`  solver= react()`
`  tools=[bash(), python(), read_file(), write_file(), web_search(), web_browser(), computer(), ...]`
`  )`
`)`
 	
H1: Local containerised execution

```
return Task(
  sandbox="docker"
)
```

N1: Narrow network access for pre-approved domains

For K8s (H2), you can enable internet access for your pre-approved domains in your helm-values.yaml file:

```
allowDomains:
- "pypi.org"
```

Remember to use *aisitools*

An agentic cyber eval that uses a Ghidra tool for decompiling binaries

The Ghidra tool happens in its own sandbox separate from the agent, at **sandbox profile T1.H2.N1**. This is because the tool call can invoke the Ghidra CLI in the sandbox but not write or execute arbitrary code and without network access.

T1: Web browsing and basic tooling

```
return Task(
  solver=basic_agent(
    tools=[web_search(), web_browser()]
  )
)
```

H2: Remote, scalable containerised execution in K8s

```
return Task(
  sandbox="k8s"
)
```

N1: Narrow network access (pre-approved domains & trusted datasets)

This is not practical for local/no sandboxing (H0) or Docker (H1) sandboxes. Therefore in practice, the sandbox profile of this evaluation would be **N1.H2.T1**

For K8s (H2), you can enable internet access for your pre-approved domains in your helm-values.yaml file:

```
allowDomains:
# For example:
- "pypi.org"
- "files.pythonhosted.org"
# And others, depending on your eval
```

An agentic cyber eval that tries to find a zero-day exploit in the Docker runtime

Sandbox profile T0.H2.N2: No internet is required for the task and the model is being actively hostile to its environment, but it will need to write and execute code to do so.

T0: Unrestricted execution and web browser access

```
return Task(
  solver=basic_agent(
    tools=[bash(), python(), read_file(), write_file(), web_search(), web_browser(), computer(), ...]
  )
)
```

H2: Remote, scalable containerised execution in K8s

```
return Task(
  sandbox="k8s"
)
```

N2: No network access

For K8s (H2), network access is disabled by default unless you supply *allowDomains* in your *helm-values.yaml* file.

A "deep research"-style eval that attempts information retrieval from a web search tool

Sandbox profile T1.H1.N0: It requires full internet access but only with limited tool-calling, and the default host sandboxing is appropriate.

T1: Web browsing and basic tooling

```
return Task(
  solver=basic_agent(
    tools=[web_search(), web_browser()]
  )
)
```

H1: Local containerised execution

```
return Task(
  sandbox="docker"
)
```

N0: Full external network access

By default in Docker Compose (H1), containers have access to the host network which will in turn have access to the internet.

```
services:
 default:
  image: ubuntu
  # This is the default
  network_mode: host
```

A 'deep research'-style eval that attempts information retrieval using a general-use 'computer tool' or set of tools for general tasks.

Likely **sandbox profile** T1.H1.N0 (as above) but could be considered **T0.H1.N0**. We may want to give the model unrestricted tool access in order to test its accuracy in an environment in which its tool use would not be restricted.

T0: Unrestricted tool use and code execution

```
return Task(
  solver=basic_agent(
    tools=[bash(), python(), read_file(), write_file(), web_search(), web_browser(), computer(), ...]
  )
)
```

H1: Local containerised execution

```
return Task(
  sandbox="docker"
)
```

N0: Full external network access

By default in Docker Compose (H1), containers have access to the host network which will in turn have access to the internet.

```
services:
 default:
  image: ubuntu
  # This is the default
  network_mode: host
```

Full steps to configure T, H and N settings within Inspect

In case the above end-to-end guides do not apply to your use-case, you can find individual setup instructions for each element of the sandboxing taxonomy below. This lets you mix-and-match to create your own sandbox profile. We also note that users can customise the tools and domains to meet their security and scalability requirements.

Tooling

For agentic evals in Inspect, one would typically configure the available tools as part of the agent's definition in the task definition. Be sure to check whether any agents you are using will automatically make certain tools available.

T0: Unrestricted execution and web browser access

```
return Task(
  solver=basic_agent(
    tools=[bash(), python(), read_file(), write_file(), web_search(), web_browser(), computer(), ...]
  )
)
```

T1: Web browsing and basic tooling

```
return Task(
  solver=basic_agent(
    tools=[web_search(), web_browser()]
  )
)
```

T2: No execution

No tools provided.

```
return Task(
  solver=[generate()]
)
```

Host isolation

This is controlled by the type of sandbox environment.

H0: No isolation

This is not sandboxing - tools run directly on the same machine as Inspect is running and is not recommended.

```
return Task(
  sandbox="local"
)
```

H1: Local containerised execution with Docker

```
return Task(
  sandbox="docker"
)
```

H2: Remote, scalable execution with Kubernetes

```
return Task(
  sandbox="k8s"
```

```
)
```

H3: Temporary, fully-isolated, remote VM with Proxmox

```
return Task(
  sandbox="proxmox"
)
```

Network Isolation

NB: Network isolation is partly dependent on host isolation. For the local sandbox (H0), the sandbox will have the same level of network access as your machine does and is not easily configurable.

N0: Full external network access

By default in Docker Compose host isolation (H1), containers have access to the host network which will in turn have access to the internet.

```
services:
 default:
  image: ubuntu
  # This is the default
  network_mode: host
```

For K8s host isolation (H2) (assuming you're using the built-in Helm Chart), the default is to have no internet access. You can enable full internet access in your helm-values.yaml file:

```
allowDomains: "*"
```

N1: Narrow network access (pre-approved domains & trusted datasets)

This is only compatible with H2 sandboxes (Remote and containerised execution via Kubernetes).

When using the K8s sandbox provider (H2), instead of setting *'network_mode'* (as you would do in Docker Compose), you can enable internet access for a restricted set of pre-approved domains. To do this, set the '*allowDomains*' field in your *helm-values.yaml file:*

```
allowDomains:
- "pypi.org"
- "files.pythonhosted.org"
```

N2: No external network access

In Docker Compose host isolation (H1), you can disable network access:

```
services:
 default:
  image: ubuntu
  network_mode: none
```

This is the default in Inspect if you don't provide your own Docker Compose file.

For K8s host isolation (H2), network access is disabled by default unless you supply *allowDomains* in *your helm-values.yaml* file.

Note that at this sandbox profile, local network communication may be possible in the controlled scenarios.

Logging and monitoring

Inspect has an Extension API for lifecycle hooks, which can be used to record events (including sandbox metadata) for post-hoc analysis.

Inspect also supports approval modes for potentially dangerous tool calls: https://inspect.ai-safety-institute.org.uk/approval.html

Incompatible configurations

Some configurations combinations do not make sense. For example, having a sandbox (H1-H3) when there are no tools available (T2).

Docker Compose (H1) does not make fine-grained network isolation practical (N1) – it is better to have Kubernetes Host Isolation here (H2).

For no sandboxing (H0), network isolation cannot be easily enforced as you would have to manually configure your machine which may prevent model access via API(unless it is a locally hosted model).

Incompatible configurations in red:

	T0	T1	T2	H0	H1	H2	H3	N1	N2
T0									
T1									
T2									
H0									
H1									
H2									
H3									
N1									
N2									
N3									

Changelog

- 2025/08/06: First version