AI Defense Validation

AI Validation tests allow you to find and assess vulnerabilities in the generative AI models your organization uses. It sends a set of attacks comprised of attack techniques and intents to the model and evaluates whether they are successful or not. An attack is considered successful if the model returns a response that aligns with the (usually malicious) intent.

Use the Validation page to create, find, and run tests, and to review their results.

Validation Results

The Validation page lists running and past validation tests, and it lets you inspect the results of any completed test.

Filter results

To filter results, use the fields at the top of the page to specify and of the following. As you enter your filter criteria, the list updates automatically. You can filter on:

- **Test name or asset name**: Checks for a match against the test, model, or application name
- **Start/end date and time**: The time span during which the test ran
- **Asset type**: Whether the test subject was an AI model or application
- **AI asset name**: Name of the model or application as stored in AI Defense.

Results list

The test results summary list displays running and past tests. Click the name of any completed test to inspect its results. For each test, the summary list shows:

- **Test name**: Each test has a name for easy lookup later.
- **Asset type**: Whether an AI model or application was tested
- **AI asset name**: Name of the model or application tested
- **Test run date**: Timestamp indicating when the test was started
- **Attack success rate** : Percentage of attacks that succeeded
- **Status**: Whether this test is in progress, completed, or failed. A completed test is one in which all attacks were sent and their responses evaluated.

Finding AI Asset

Use the AI Assets tab for an alphabetical list of discovered models and applications in your environment. If the asset you're looking for is missing, see "Add an AI Asset," below.

Adding an AI Asset

AI Validation can test AI models and applications:

- An **AI model** is an LLM that has been auto-discovered using Multicloud Defense. Currently, only AWS Bedrock is supported. You can see the discovered AI models in the **AI Assets: Cloud visibility** tab. If the model you're looking for is missing, make sure you've connected to the cloud service that hosts your models and applications. See AI Defense's **Administration** page to connect to a cloud service.
- An **application** is an an LLM-powered application, such as a chatbot, that you have manually registered using AI Defense's **Applications** page.

Configure and run a validation test of an AI model

A model validation test requires a set of parameters that will be used to connect to and test the AI model. To configure and run a validation test for a model:

1. Click the **Run validation** button in the Validation page.
2. In **Asset type**, choose "Model."
3. Specify the test parameters and the model to be tested:
   - **AI asset name**: The name of the model as discovered by Multicloud Defense. See the **AI Assets** page for a list.
   - **Model ID**: The model ID as stored in the platform hosting the model.
   - **Test name**: Give this test a memorable name to better find it later.
   - **Prompt template**: This is the JSON request payload that will be sent to the model's inference API in order to test it. This must include a `{{prompt}}` placeholder where the AI Defense-generated test prompts will be inserted.
   - **Response**: a JSON path that specifies where in the HTTP response. See "Formatting the response path" below.
4. Click **Submit**. The test will run immediately.

Configure and run a validation test of an AI application

An application validation test requires a set of parameters that will be used to connect to and test the application's AI model. Follow the steps below to configure and run a validation test for an application.

Note An application validation evaluates the model connected to the application, not the application itself.

1. Click the **Run validation** button in the Validation page.
2. In **Asset type**, choose "Application."

3. Specify the test parameters and the model to be tested:
   - **Application**: Application to be tested.
   - **Test name**: Give this test a memorable name to better find it later.
   - **Endpoint**: The endpoint of the LLM used by the application
   - **Inference API path**: The API path for model inference calls. For example: `/openai/deployments/gpt3.5`
   - **Prompt template**: This is the JSON request payload that will be sent to the model's inference API in order to test it. This must include a `{{prompt}}` placeholder where the AI Defense-generated test prompts will be inserted.
   - **Response**: a JSON path that specifies where in the HTTP response. See "Formatting the response path" below.
   - **HTTP headers**: Headers for the inference API connection. Specify the authorization values here.
4. Click **Submit**. The test will run immediately.

Formatting the response path

In the **Response** field, provide a JSON path that specifies where in the HTTP response JSON payload AI Defense can find the LLM's response string in order to validate whether the attack was successful. The path must point to a string value in the JSON payload.

Remember Each model provider uses its own response format. Check your model provider's API documentation for the correct format before you set the response path.

Note
- Whitespace and other special characters can be encoded as unicode (`\u0020`).
- Periods in JSON fields can be escaped with a backslash.
- Array elements can be specified with the element's index in square brackets, for example by including `[0]` when you want to retrieve the first element.

Response path examples

To retrieve a response from a top-level field:

- For example, if the endpoint returns a response like `{"response": "I am an AI Chatbot, how can I assist you?"}`
- You would set a **Response** value of `response`

To retrieve a response from a nested JSON field:

- For example, if the endpoint returns a nested response like

```
{"response": {"llmResponse": "I am an AI Chatbot, how can I assist
          you?"}}
```

- You would set a **Response** value of `response.llmResponse`

To extract a response string from an array, specify the element's index in square brackets.

For example, if the endpoint returns a nested response like:

```
{
    "content": [
        {
            "text": "Bonjour, je suis Claude!",
            "type": "text"
        }
    ],
    "id": "msg459674598",
    "model": "claude-3-5-sonnet-2024-08-20",
    "role": "assistant"
}
```

- You would set a **Response** value of: `content.[0].text`

To handle periods in field names, use a backslash:

- For example, if the endpoint returns a nested response like `{"llm.response": "hello"}`
- You would set a **Response** value of `llm\.response`
- The syntax applies to dot notation only, such as `myfield.myotherfield` or `myarray.1`

Initial Configuration of AI Validation

To set up AI Validation:

1. In your cloud service (currently AWS Bedrock is supported), find the IAM role ARN for an account with access to your models.
2. Open the AI Defense **Administration** tab, go to the AWS Bedrock card, and click **Connect**, and provide the API key details to complete the connection. *See the* AI Defense Administration documentation *for details.*
3. Make sure Multicloud Defense is connected to AI Defense. If the Multicloud Defense card on the Administration tab shows a **Disconnect** button, then Multicloud Defense is connected. If it's not connected, *see the section* Set up AI Asset discovery.
4. Proceed to the sections, Find AI Asset and Add an AI Asset, above, to add the AI models and applications you wish to scan.