



PEACH  
FUZZER

## Peach Fuzzer Hosted Trial Guide



# 1. Preface

This document supports the user through a trial of using Peach Fuzzer to test several different real-world applications. It shows how Peach Fuzzer should be configured for each application, what monitors are useful for each scenario, and has examples of test runs that show the types of faults that Peach Fuzzer can find and the information it gathers for those faults. This should give the user an overview of the capabilities of Peach Fuzzer and set expectations for what the user will need to do to use Peach Fuzzer to test their own software.

## 1.1. Goals

After reading this document, you should be able to accomplish the following:

1. Access your trial instance of Peach Fuzzer
2. Run tests against several target applications with Peach Fuzzer
3. View the results of a test run to see the vulnerabilities that were found and the data was gathered
4. Understand how Peach Fuzzer is configured to test various applications based on the nature of the application
5. Understand why the configurations have the specific settings and Monitors that they do and how those settings relate to the specifics of the application they are testing
6. (Optional) Create a new configuration to test one of the existing applications that exist on your trial instance

## 1.2. Non-Goals

This document is NOT intended to be comprehensive documentation for how to configure Peach Fuzzer or a full demonstration of every feature that Peach Fuzzer offers. It is also not meant to help the user install, configure, or troubleshoot Peach Fuzzer for testing their own applications. Users needing assistance with any of those things should consult the Peach Fuzzer User Guide or contact support for assistance.

## 1.3. Target Applications

The applications being tested in the trial are all free and open source where possible. In many cases, the source was forked from the original version and contrived security vulnerabilities were intentionally introduced into the code in order to better demonstrate the features of Peach Fuzzer. Therefore, the vulnerabilities found by Peach Fuzzer on the trial instance should not be assumed to be present in the publicly available versions of those applications. In addition, Peach Tech has not done a comprehensive test of those applications so we cannot guarantee that additional vulnerabilities are not present in those applications' original source code.

## 2. Configurations

Each pit available in the trial has at least one configuration. In some cases there will be two; a basic configuration and an advanced configuration.



In protocols where both a client and server are present and being tested, the client and server are considered separate protocols for purposes of this document. It is therefore possible that both the client and the server will each have a basic and advanced configuration.

### 2.1. Basic Configuration

This configuration shows how a user would typically set up Peach Fuzzer to test an application that they either can't compile or can't re-compile with additional compiler or linker options. The applications used in these configurations have typically been compiled with debug and no optimizations e.g. `gcc -g main.cpp` or similar. The GDB Monitor is used in most cases to detect faults.

### 2.2. Advanced Configuration

This configuration shows how a user would typically set up Peach Fuzzer to test an application that they have recompiled with various compiler options such as Address Sanitizer (the advanced configurations used in the trial are all compiled this way unless indicated otherwise) and various optimization levels e. g. `gcc -g -O1 -fsanitize=address main.cpp` or similar.

With ASan in use, the GDB Monitor is generally not advised since ASan will terminate the program without sending a signal (e.g. SIGSEGV, SIGABRT, etc.) that the debugger will detect. The Process Monitor is therefore used instead, as it will recognize that ASan has terminated the process for some reason and can gather information from the ASan output in the report for the fault on that particular iteration.

## 3. Accessing your trial instance

You should have received the following for your trial instance:

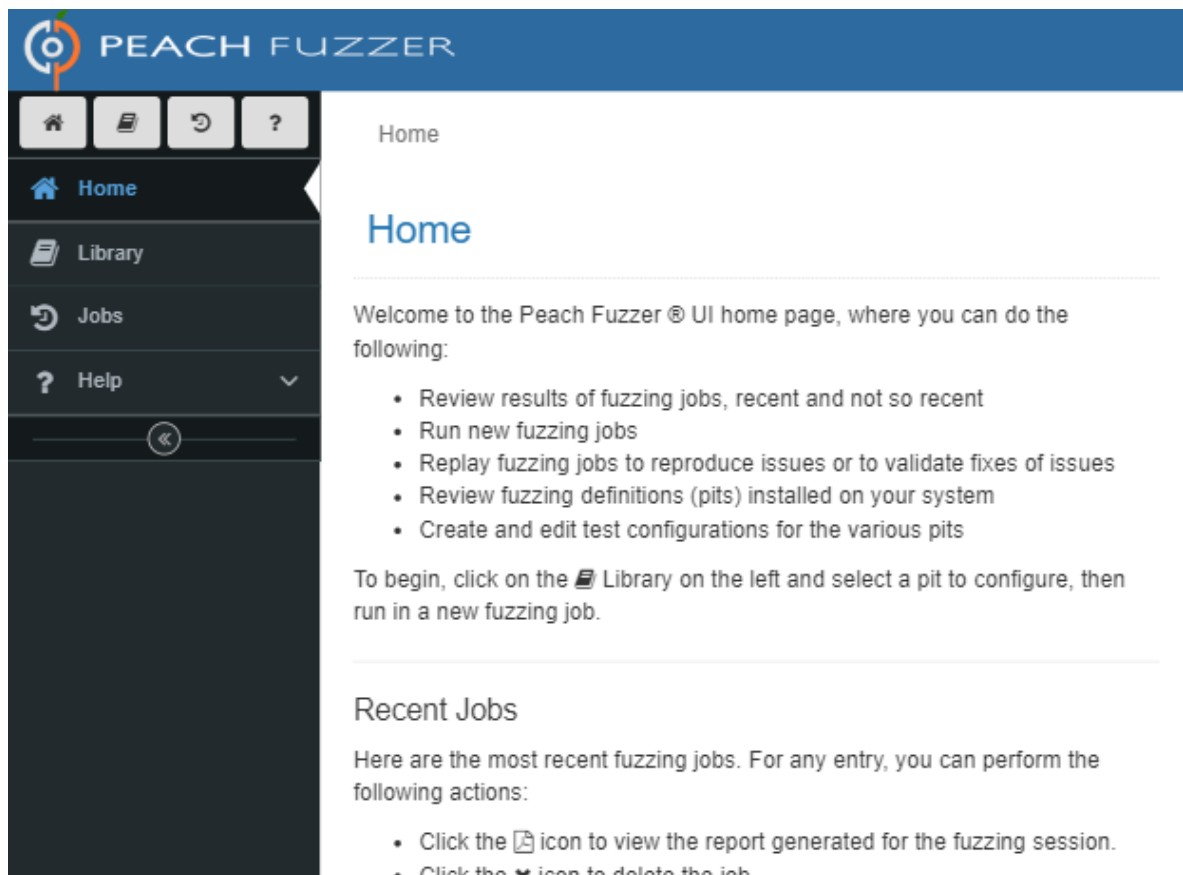
- The URL to access the instance
- The login/password for your trial instance

If you do not have this information, please contact [support@peach.tech](mailto:support@peach.tech) for assistance.

### 3.1. Logging in to your trial instance

To log in to your trial instance, follow these steps:

1. Enter the URL into your web browser. It should be something similar to <https://mycompany.demo.peach.tech>
2. When prompted, enter your credentials.
3. Click "Accept" to accept the license agreement.
4. If prompted a second time for credentials, enter the same credentials you used in step 2.
5. You should now see a screen similar to what is pictured below



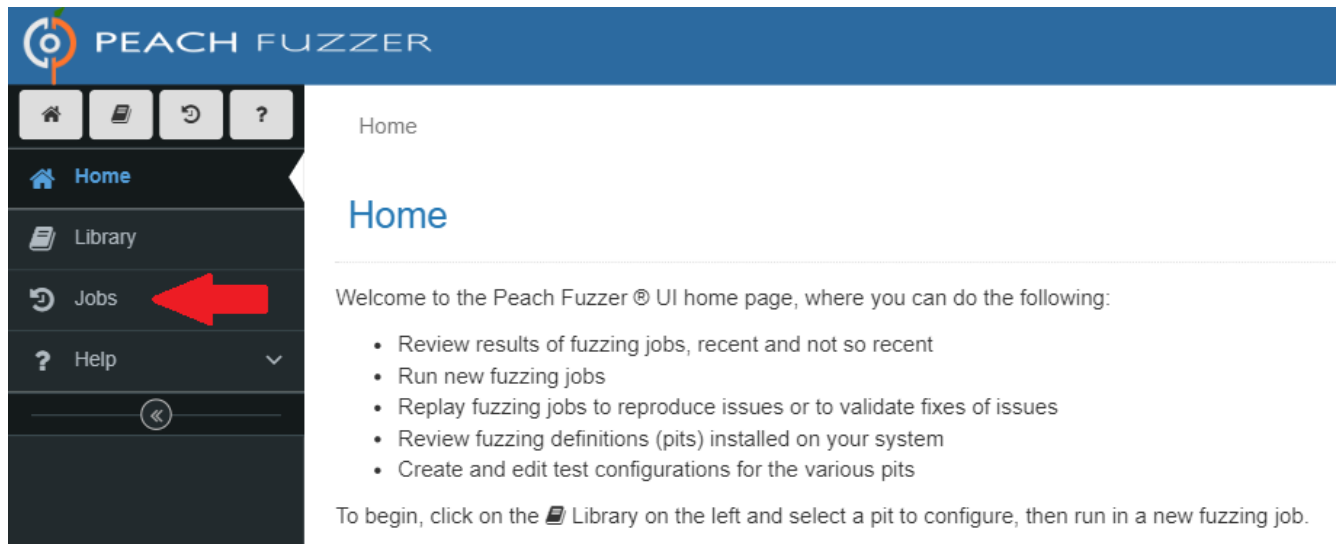
## 4. Sample Configurations

The following are sample configurations for several different protocols as well as file types that Peach Fuzzer can fuzz. Each sample will already be present and configured on your trial instance. In addition, a test run for each sample is already present on the trial instance so that you can easily view the test results and see the types of faults that Peach Fuzzer can find. You can run the sample configurations with the supplied seed values in this guide. Each section below has instructions for how to set up the sample configuration, including a brief explanation of why Peach Fuzzer has been configured this way for this particular application. It is strongly recommended that you use the supplied values in each section for the **Seed** and **Stop Test Case** (and **Start Test Case** if indicated). These values have been selected to guarantee that you can create a test run that will find faults in the target applications within a few minutes.

### 4.1. Viewing the Sample Job Results

Your trial instance already has test runs for each available sample configuration. These can be viewed from the Jobs page. Any additional test runs you perform will be available on the Jobs page. To view the results:

1. From the Home page, click the Jobs tab.



2. Click the job you wish to view from the list of available jobs.

Library



**Jobs**



















Help

## Jobs

Here is a comprehensive list of the fuzzing jobs on this computer.

For any entry, you can perform the following actions:

- Click the  icon to view the report generated for the fuzzing session.
- Click the  icon to delete the job.


Name	Status	Start Time	Stop Time	Test Cases	Total Faults	Actions
Example-JPG-Advanced	Stopped	6/25/18 5:23 PM	6/25/18 5:23 PM	46	1	 
Example-JPG-Basic	Stopped	6/25/18 5:20 PM	6/25/18 5:21 PM	46	1	 
Example-PNG-Advanced	Stopped	6/25/18 5:19 PM	6/25/18 5:19 PM	11	1	 
Example-PNG-Basic	Stopped	6/25/18 5:19 PM	6/25/18 5:19 PM	11	1	 
Example-MODBUS-TCP Slave	Stopped	6/25/18 5:17 PM	6/25/18 5:17 PM	30	1	 
Example-MODBUS-TCP Master	Stopped	6/25/18 5:14 PM	6/25/18 5:15 PM	50	3	 
Example-SNMPv3 Server-Advanced	Stopped	6/25/18 5:11 PM	6/25/18 5:12 PM	10	5	 
Example-SNMPv3 Server-Basic	Stopped	6/25/18 5:06 PM	6/25/18 5:10 PM	10	5	 
Example-HTTP Server-Advanced	Stopped	6/25/18 5:04 PM	6/25/18 5:05 PM	30	2	 

3. The results of the selected job will now be displayed. You can see the overall results which will indicate the parameters with which the job was run and the faults that were found. You can examine the [faults](#) individually or [download a report](#) that summarizes all of the findings in this job run.

## 4.2. Running the Samples

To run the pre-configured examples:

1. Click **Library**

 PEACH FUZZER

Home

**Home**

Library

Jobs

Help

## Home

Welcome to th

- Review
- Run new
- Replay 1
- Review

2. Under **Configurations**, click the name of the sample configuration that you wish to run

## Configurations

The Configurations section contains existing Peach Pit configurations. Selecting an existing configuration allows editing the configuration and starting a new fuzzing job.

Image			
<a href="#">Example-JPG-Advanced</a>	<a href="#">Example-PNG-Advanced</a>		
<a href="#">Example-JPG-Basic</a>	<a href="#">Example-PNG-Basic</a>		
Net			
<a href="#">Example-DICOM Net Provider</a>	<a href="#">Example-DNP3 Slave-Basic</a>	<a href="#">Example-HTTP Server-Basic</a>	<a href="#">Example-SNMPv3 Server-Advanced</a>
<a href="#">Example-DNP3 Master-Advanced</a>	<a href="#">Example-HL7 Net TCP MLLP Receiver</a>	<a href="#">Example-MODBUS-TCP Master</a>	
<a href="#">Example-DNP3 Master-Basic</a>		<a href="#">Example-MODBUS-TCP Slave</a>	<a href="#">Example-SNMPv3 Server-Basic</a>
<a href="#">Example-DNP3 Slave-Advanced</a>	<a href="#">Example-HTTP Server-Advanced</a>		



Your trial instance may not have every Configuration pictured here. The exact Configurations available will depend on what Pits are included with your trial license.

3. Enter the appropriate values for **Seed** and **Stop Test Case**.
4. Enter the appropriate value for **Start Test Case** if specified. Otherwise, leave the default of **1**.

## 4.3. DNP3-Master

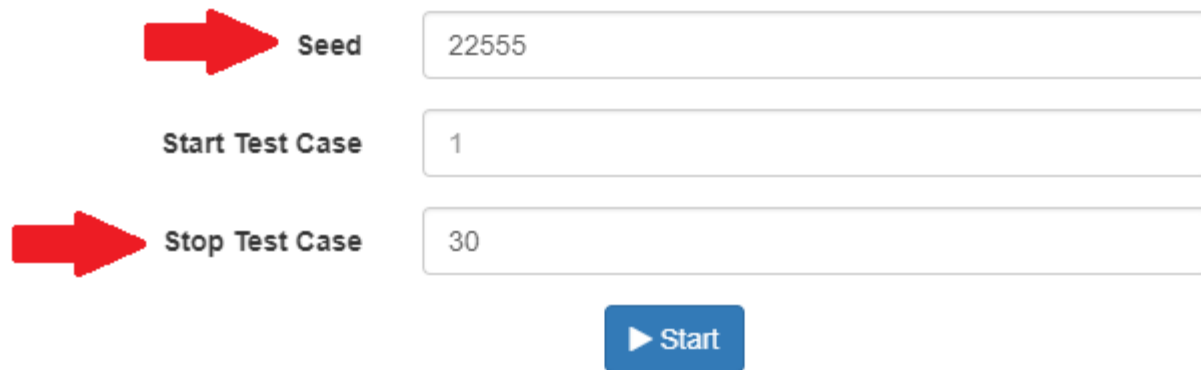
This configuration will test an application using DNP3. It has two versions, a basic configuration and an advanced configuration. The basic configuration is compiled only with the Debug option. The advanced configuration is compiled with Address Sanitizer, Debug, and Optimization Level 1.

### 4.3.1. Running the test

To run the pre-configured basic test:


1. Click **Example-DNP3\_Master-Basic**.
2. Enter a seed value of **22555**.
3. Enter a Stop Test Case value of **30**.
4. Click **Start**.





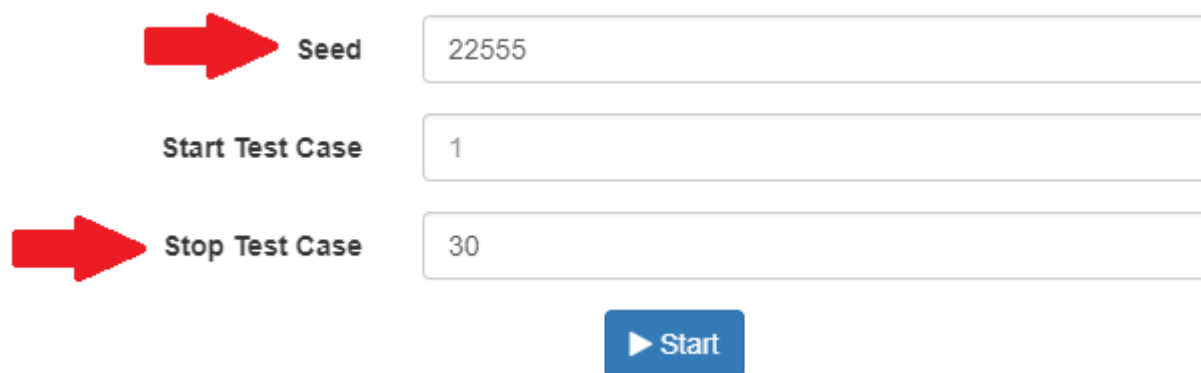
The screenshot shows a configuration form with three input fields and a button. The first field is labeled 'Seed' with a red arrow pointing to it, containing the value '22555'. The second field is labeled 'Start Test Case' and contains the value '1'. The third field is labeled 'Stop Test Case' with a red arrow pointing to it, containing the value '30'. Below these fields is a blue button with a play icon and the text 'Start'.

Seed	22555
Start Test Case	1
Stop Test Case	30



To run the pre-configured advanced test:

1. Click **Example-DNP3\_Master-Advanced**.
2. Enter a seed value of **22555**.
3. Enter a Stop Test Case value of **30**
4. Click **Start**.



This is a duplicate of the screenshot above, showing the same configuration form with 'Seed' (22555), 'Start Test Case' (1), 'Stop Test Case' (30), and a 'Start' button.

### 4.3.2. Configuring the test

These steps will create the same configuration as is in use in the DNP3\_Master configuration that is already present on the trial instance. The steps are the same for both the basic and advanced configuration except where indicated. To create the configuration:

1. Click Library and then click **DNP3\_Master**.
2. Enter a name when prompted and optionally a description, then click **Submit**.

### Configuring Variables

The first thing you need to configure are the variables that control how Peach Fuzzer will test the application. Follow these steps to create a working configuration on your trial instance:

1. Click **Configure Variables**.
2. Configure your variables as appropriate for your application. The following should be used for

opendnp3 running on the trial instance:

- a. Listen IPv4 Address: set to **0.0.0.0** so that Peach Fuzzer will listen for incoming connections on all network interfaces.
  - b. Listen Port: Leave the default value of **20000** since opendnp3 will also default to this port.
  - c. DNP3 Target Address: Set to **1** to match the configuration of opendnp3.
  - d. DNP3 Source Address: Set to **10** to match the configuration of opendnp3.
  - e. Under **Advanced Configuration**, leave all the defaults as they are as they are acceptable for opendnp3.
  - f. Under System Defines, do NOT change any of the values present. These values normally do not require changing.
3. Once all the settings have the desired values, click **Save**.

▼ Basic Configuration

Name	Key	Value
Listen IPv4 Address	ListenIPv4	<div>0.0.0.0</div> <div>IPv4 address that Peach uses to listen for incoming messages. The value '0.0.0.0' means Peach listens on all interfaces. The default value is '0.0.0.0'.</div>
Listen Port	ListenPort	<div>20000</div> <div>Port number that Peach uses to listen for incoming messages. The default value is '20000'.</div>
DNP3 Target Address	Destination	<div>1</div> <div>DNP3 Target Address.</div>
DNP3 Source Address	Source	<div>10</div> <div>DNP3 Source Address.</div>

➤ Advanced Configuration

➤ System Defines

## Configuring Agents

An Agent runs either in-process of Peach Fuzzer or can be installed and run on a remote machine. For this configuration, only a single local agent is required.

1. Click **Monitoring**.
2. Click **Add Agent**.
3. Enter a name. Leave the **Location** setting to the default **local://**.

4. Click **Save**.





## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Saved successfully.

 Save  + Add Agent

 local:// (local)   

**Name**

local

Friendly name for your agent

**Location**

local://

URL for the agent. Leave blank for a local agent. For remote agents use the **tcp** scheme. The default agent port is **9001**. Example: **tcp://192.168.48.2:9001**

For more detailed instructions, see [Adding an agent](#).

## Configuring the monitors for basic

The basic configuration is targeting openssl compiled with Debug enabled. You will therefore want the following monitors:

- **Gdb Monitor.** This will allow Peach Fuzzer to launch openssl from within GDB so that GDB attaches to openssl. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor.** Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Gdb**. Click **Ok**.
2. Under **Executable**, enter **/var/targets/dnp3/master-demo** which is the location of openssl's launcher. This will allow the monitor to launch the application when fuzzing starts. Do not change any of the other settings for this monitor.
3. Click **Save**.

Gdb (Gdb)

Name

Gdb

Friendly name for your monitor

Core Parameters

Executable

/var/targets/dnp3/master-demo

Executable to launch

Arguments

Optional command line arguments

Gdb Path

/usr/bin/gdb

Path to gdb

When To Trigger

Restart On Each Test

false

Restart process for each iteration

Restart After Fault

false

Restart process after any fault occurs

Start On Call

StartIterationEvent

Start command on state model call

Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

> Advanced

4. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
5. Under **Device**, enter **lo** (for loopback, since opendnp3 is listening on the loopback interface)
6. Under **Filter**, enter **port ##TargetPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
7. Click **Save**.

## Configuring the monitors for advanced

The advanced configuration is targeting `opendnp3` compiled with Debug, Address Sanitizer (ASan), and Optimization level 1. You will therefore want the following monitors:

- **Process Monitor.** This will allow Peach Fuzzer to launch opendnp3. Because opendnp3 is compiled with ASan, the Process Monitor will detect the program exited due to an error and gather the output from ASan about the nature of the crash.



Do not use the GDB Monitor for applications that are compiled with ASan. It is not compatible with ASan.

- **Network Capture monitor.** Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Process**. Click **Ok**.
2. Under **Executable**, enter **/var/targets/advanced/dnp3/outstation-demo** which is the location of opendnp3's launcher. This will allow the monitor to launch the application when fuzzing starts.
3. Do not change any of the other settings for this monitor.
4. Click **Save**.

Process (Process)

Name

Process

Friendly name for your monitor

Core Parameters

Executable

/var/targets/advanced/dnp3/master-demo

Executable to launch

Arguments

Optional command line arguments

When To Trigger

Restart On Each Test

false

Restart process for each iteration

Restart After Fault

false

Restart process after any fault occurs

Start On Call

StartIterationEvent

Start command on state model call

Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

Advanced

- Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
- Under **Device**, enter **lo** (for loopback, since opendnp3 is listening on the loopback interface)
- Under **Filter**, enter **port ##TargetPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
- Click **Save**.

NetworkCapture (Network Capture)

Name: Network Capture  
Friendly name for your monitor

Core Parameters

Device: lo  
Device name for capturing on

Filter: port ##ListenPort##  
PCAP Style filter

## Testing your configuration

You should always test your configuration to ensure that Peach Fuzzer is able to fuzz your application. This will run some control iterations to ensure that your application responds as expected to normal input.

1. Click **Test**
2. Click **Begin Test**
3. Once testing is complete, you will see the results. Correct any errors in your Variables or Monitoring if the test is not successful.
4. Once the test has passed, click **Continue**.

You can now fuzz the application.

Follow the steps under [Running the test](#) to start testing the application.

## 4.4. DNP3-Slave

This configuration will test an application using DNP3. It has two versions, a basic configuration and an advanced configuration. The basic configuration is compiled only with the Debug option. The advanced configuration is compiled with Address Sanitizer, Debug, and Optimization Level 1 options.

### 4.4.1. Running the test

To run the pre-configured basic test:

1. Click **Example-DNP3\_Slave-Basic**.

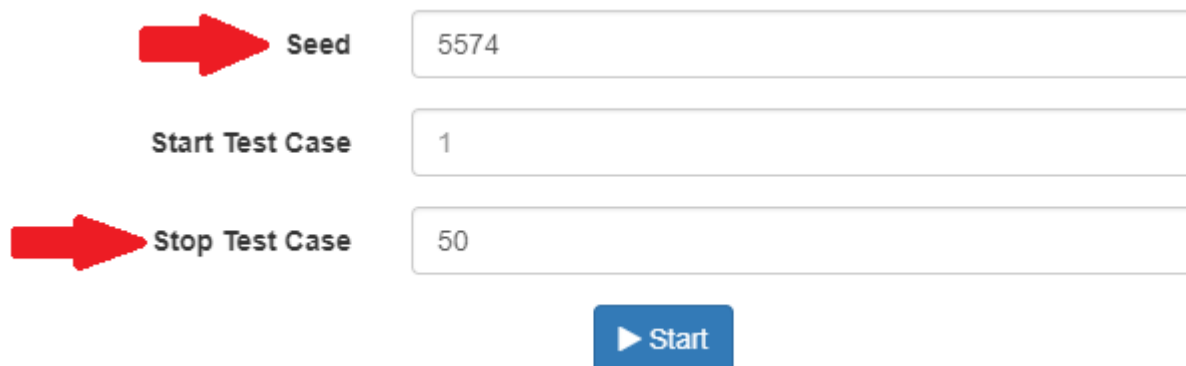
2. Enter a seed value of 5574.
3. Enter a Stop Test Case value of 50.
4. Click **Start**.



A configuration form with three input fields and a button. The first field is labeled 'Seed' with a red arrow pointing to it, containing the value '5574'. The second field is labeled 'Start Test Case' and contains the value '1'. The third field is labeled 'Stop Test Case' with a red arrow pointing to it, containing the value '50'. Below the fields is a blue button with a play icon and the text 'Start'.

To run the pre-configured advanced test:

1. Click **Example-DNP3\_Slave-Advanced**.
2. Enter a seed value of 5574.
3. Enter a Stop Test Case value of 50
4. Click **Start**.



A configuration form with three input fields and a button. The first field is labeled 'Seed' with a red arrow pointing to it, containing the value '5574'. The second field is labeled 'Start Test Case' and contains the value '1'. The third field is labeled 'Stop Test Case' with a red arrow pointing to it, containing the value '50'. Below the fields is a blue button with a play icon and the text 'Start'.

#### 4.4.2. Configuring the test

These steps will create the same configuration as is in use in the DNP3\_Slave configuration that is already present on the trial instance. The steps are the same for both the basic and advanced configuration except where indicated. To create the configuration:

1. Click Library and then click **DNP3\_Slave**.
2. Enter a name when prompted and optionally a description, then click **Submit**.

#### Configuring Variables

The first thing you need to configure are the variables that control how Peach Fuzzer will test the



application. Follow these steps to create a working configuration on your trial instance:

1. Click **Configure Variables**.
2. Configure your variables as appropriate for your application. The following should be used for opendnp3 running on the trial instance:
  - a. Target IPv4 Address: set to **127.0.0.1**
  - b. Target Port: Leave the default value of **20000** since opendnp3 will also default to this port.
  - c. DNP3 Target Address: Set to **10** to match the configuration of opendnp3.
  - d. DNP3 Source Address: Set to **1** to match the configuration of opendnp3.
  - e. Under **Advanced Configuration**, leave all the defaults as they are acceptable for opendnp3.
  - f. Under System Defines, do NOT change any of the values present. These values normally do not require changing.
3. Once all the settings have the desired values, click **Save**.

▼ Basic Configuration

Name	Key	Value
Target IPv4 Address	TargetIPv4	<div>127.0.0.1</div> <div>IPv4 address of the target machine.</div>
Target Port	TargetPort	<div>20000</div> <div>Port number the target machine uses to receive messages. The default value is '20000'.</div>
DNP3 Target Address	Destination	<div>10</div> <div>DNP3 Target Address.</div>
DNP3 Source Address	Source	<div>1</div> <div>DNP3 Source Address.</div>

➤ Advanced Configuration

➤ System Defines

## Configuring Agents

An Agent runs either in-process of Peach Fuzzer or can be installed and run on a remote machine. For this configuration, only a single local agent is required.

1. Click **Monitoring**.
2. Click **Add Agent**.

3. Enter a name. Leave the **Location** setting to the default **local://**.

4. Click **Save**.

## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Saved successfully.

 Save  Add Agent

▼ local:// (local)

Name

local

Friendly name for your agent

Location

local://

URL for the agent. Leave blank for a local agent. For remote agents use the **tcp** scheme. The default agent port is **9001**. Example: **tcp://192.168.48.2:9001**

For more detailed instructions, see [Adding an agent](#).

## Configuring the monitors for basic

The basic configuration is targeting opendnp3 compiled with Debug enabled. You will therefore want the following monitors:

- **Gdb Monitor.** This will allow Peach Fuzzer to launch opendnp3 from within GDB so that GDB attaches to opendnp3. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor.** Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Gdb**. Click **Ok**.
2. Under **Executable**, enter **/var/targets/dnp3/outstation-demo** which is the location of opendnp3's launcher. This will allow the monitor to launch the application when fuzzing starts. Do not change any of the other settings for this monitor.
3. Click **Save**.

Gdb (Gdb)

Name

Gdb

Friendly name for your monitor

Core Parameters

Executable

/var/targets/dnp3/outstation-demo

Executable to launch

Arguments

Optional command line arguments

Gdb Path

/usr/bin/gdb

Path to gdb

When To Trigger

Restart On Each Test

false

Restart process for each iteration

Restart After Fault

false

Restart process after any fault occurs

Start On Call

Start command on state model call

Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

> Advanced

4. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
5. Under **Device**, enter **lo** (for loopback, since opendnp3 is listening on the loopback interface)
6. Under **Filter**, enter **port ##TargetPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
7. Click **Save**.

▼ NetworkCapture (Network Capture)

Name

Network Capture

Friendly name for your monitor

▼ Core Parameters

Device

lo

Device name for capturing on

Filter

port ##TargetPort##

PCAP Style filter

## Configuring the monitors for advanced

The advanced configuration is targeting `opendnp3` compiled with Debug, Address Sanitizer (ASan), and Optimization level 1. You will therefore want the following monitors:

- **Process Monitor.** This will allow Peach Fuzzer to launch opendnp3. Because opendnp3 is compiled with ASan, the Process Monitor will detect the program exited due to an error and gather the output from ASan about the nature of the crash.



Do not use the GDB Monitor for applications that are compiled with ASan. It is not compatible with ASan.

- **Network Capture monitor.** Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Process**. Click **Ok**.
2. Under **Executable**, enter `/var/targets/advanced/dnp3/outstation-demo` which is the location of opendnp3's launcher. This will allow the monitor to launch the application when fuzzing starts.
3. Do not change any of the other settings for this monitor.
4. Click **Save**.

Process (Process)

Name

Process

Friendly name for your monitor

Core Parameters

Executable

/var/targets/advanced/dnp3/outstation-demo

Executable to launch

Arguments

Optional command line arguments

When To Trigger

Restart On Each Test

false

Restart process for each iteration

Restart After Fault

false

Restart process after any fault occurs

Start On Call

Start command on state model call

Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

Advanced

- Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
- Under **Device**, enter **lo** (for loopback, since opendnp3 is listening on the loopback interface)
- Under **Filter**, enter **port ##TargetPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
- Click **Save**.

NetworkCapture (Network Capture)

**Name**   
Friendly name for your monitor

**Core Parameters**

**Device**   
Device name for capturing on

**Filter**   
PCAP Style filter

## Testing your configuration

You should always test your configuration to ensure that Peach Fuzzer is able to fuzz your application. This will run some control iterations to ensure that your application responds as expected to normal input.

1. Click **Test**
2. Click **Begin Test**
3. Once testing is complete, you will see the results. Correct any errors in your Variables or Monitoring if the test is not successful.
4. Once the test has passed, click **Continue**.

You can now fuzz the application.

Follow the steps under [Running the test](#) to start testing the application.

## 4.5. MODBUS

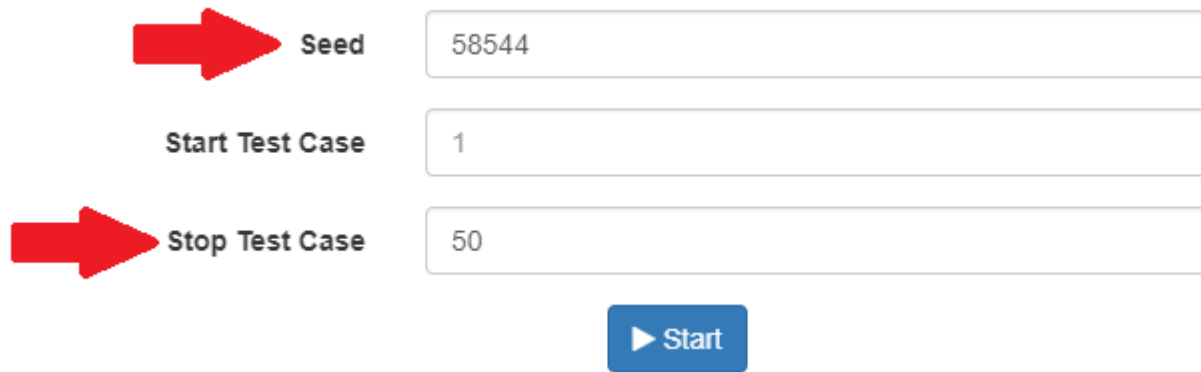
This configuration will test an application using MODBUS.

### 4.5.1. Running the test

To run the pre-configured test:

1. Click **Example-MODBUS-TCP-Master**.
2. Enter a seed value of **58544**.
3. Enter a Stop Test Case value of **50**.

4. Click **Start**.



Seed 58544

Start Test Case 1

Stop Test Case 50

▶ Start

### 4.5.2. Configuring the test

These steps will create the same configuration as is in use in the MODBUS-TCP\_Master configuration that is already present on the trial instance. To create the configuration:

1. Click Library and then click **MODBUS-TCP\_Master**.
2. Enter a name when prompted and optionally a description, then click **Submit**.

### Configuring Variables

The first thing you need to configure are the variables that control how Peach Fuzzer will test the application. Follow these steps to create a working configuration on your trial instance:

1. Click **Configure Variables**.
2. Configure your variables as appropriate for your application. The following should be used for modpoll running on the trial instance:
  - a. Listen IPv4 Address: Leave the default of **0.0.0.0** so that peach will listen on all interfaces.
  - b. Listen Port: Leave the default of **502** since modpoll will also use this port.
  - c. Under **Advanced Configuration**, leave all the defaults as they are acceptable for modpoll.
  - d. Under System Defines, do NOT change any of the values present. These values normally do not require changing.
3. Once all the settings have the desired values, click **Save**.





following monitors:

- **Gdb Monitor.** This will allow Peach Fuzzer to launch modpoll from within GDB so that GDB attaches to modpoll. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor.** Since MODBUS is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Gdb**. Click **Ok**.
2. Under **Executable**, enter **/var/targets/modbus/modpoll/linux/modpoll** which is the location of modpoll's launcher. This will allow the monitor to launch the application when fuzzing starts. Do not change any of the other settings for this monitor.
3. Under **Arguments**, enter **-m tcp 127.0.0.1**.
4. Click **Save**.

▼ Gdb (Gdb)

Name

Gdb

Friendly name for your monitor

▼ Core Parameters

### Executable

```
/var/targets/modbus/modpoll/linux/modpoll
```

Executable to launch

## Arguments

```
-m tcp 127.0.0.1
```

### Optional command line arguments

### Gdb Path

```
/usr/bin/gdb
```

Path to gdb

### ▼ When To Trigger

### Restart On Each Test

false

Restart process for each iteration

### Restart After Fault

false

Restart process after any fault occurs

## Start On Call

StartIterationEvent

Start command on state model call

### Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

➤ Advanced

5. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
6. Under **Device**, enter **lo** (for loopback, since modpoll is listening on the loopback interface)
7. Under **Filter**, enter **port ##ListenPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
8. Click **Save**.

NetworkCapture (Network Capture)

Name: Network Capture  
Friendly name for your monitor

Core Parameters

Device: lo  
Device name for capturing on

Filter: port ##ListenPort##  
PCAP Style filter

## Testing your configuration

You should always test your configuration to ensure that Peach Fuzzer is able to fuzz your application. This will run some control iterations to ensure that your application responds as expected to normal input.

1. Click **Test**
2. Click **Begin Test**
3. Once testing is complete, you will see the results. Correct any errors in your Variables or Monitoring if the test is not successful.
4. Once the test has passed, click **Continue**.

You can now fuzz the application.

Follow the steps under [Running the test](#) to start testing the application.

## 4.6. MODBUS-Slave

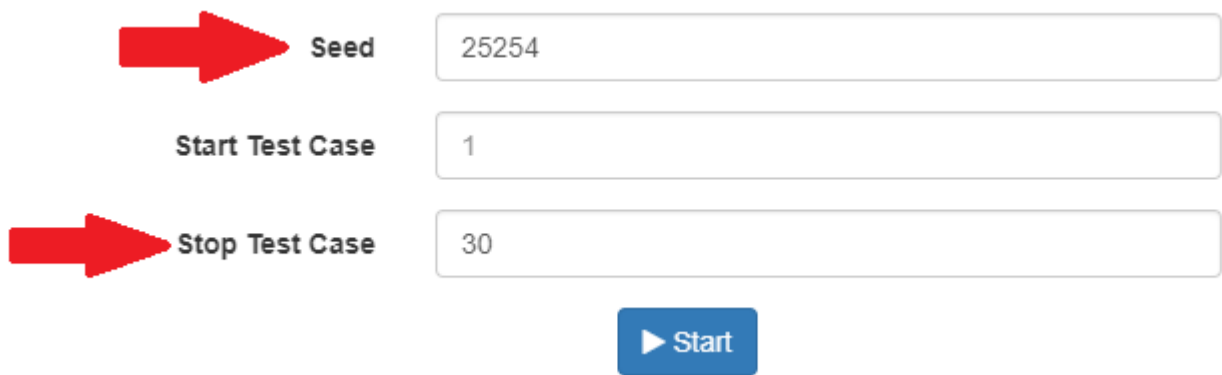
This configuration will test an application using MODBUS.

### 4.6.1. Running the test

To run the pre-configured test:

1. Click **Example-MODBUS-TCP\_Slave**.
2. Enter a seed value of **25254**.
3. Enter a Stop Test Case value of **30**.

4. Click **Start**.



Seed 25254

Start Test Case 1

Stop Test Case 30

Start

### 4.6.2. Configuring the test

These steps will create the same configuration as is in use in the MODBUS-TCP\_Slave configuration that is already present on the trial instance. To create the configuration:

1. Click Library and then click **MODBUS-TCP\_Slave**.
2. Enter a name when prompted and optionally a description, then click **Submit**.

#### Configuring Variables

The first thing you need to configure are the variables that control how Peach Fuzzer will test the application. Follow these steps to create a working configuration on your trial instance:

1. Click **Configure Variables**.
2. Configure your variables as appropriate for your application. The following should be used for diagslave running on the trial instance:
  - a. Target IPv4 Address: Set to **127.0.0.1** to ensure that Peach Fuzzer can connect to diagslave.
  - b. Target Port: Leave the default of **502** since diagslave will use this value by default.
  - c. Coil Address: Leave the default of **1**.
  - d. Discrete Input Address: Leave the default of **1**.
  - e. Input Register: Leave the default of **1**.
  - f. Holding Register: Leave the default of **1**.
  - g. File Number: Leave the default of **1**.
  - h. FIFO Pointer Address: Leave the default of **1**.
  - i. Under **Advanced Configuration**, leave all the defaults as they are acceptable for diagslave.
  - j. Under System Defines, do NOT change any of the values present. These values normally do not require changing.
3. Once all the settings have the desired values, click **Save**.

### ▼ Basic Configuration

Name	Key	Value
Target IPv4 Address	TargetIPv4	<div>127.0.0.1</div> <div>IPv4 address of the target machine.</div>
Target Port	TargetPort	<div>502</div> <div>Port number the target machine uses to receive messages. The default value is '502'.</div>
Coil Address	CoilAddress	<div>1</div> <div>Address (16 bits) of a coil on the remote device. The default value is '1'.</div>
Discrete Input Address	DiscreteInputAddress	<div>1</div> <div>Address (16 bits) of a discrete input on the remote device. The default value is '1'.</div>
Input Register	InputRegister	<div>1</div> <div>Address (16 bits) of an input register on the remote device. The default value is '1'.</div>
Holding Register	HoldingRegister	<div>1</div> <div>Address (16 bits) of a holding register on the remote device. The default value is '1'.</div>
File Number	FileNumber	<div>1</div> <div>A 16-bit number identifying the file to access. Older systems may limit the file number to 10 (0x0A). The default value is '1'.</div>
FIFO Pointer Address	FIFOPointerAddress	<div>1</div> <div>Address (16 bits) of a FIFO structure on the remote device. The first entry of the FIFO is a count of the items contained therein. The default value is '1'.</div>

➤ Advanced Configuration

## ➤ System Defines

## Configuring Agents

An Agent runs either in-process of Peach Fuzzer or can be installed and run on a remote machine. For this configuration, only a single local agent is required.

1. Click **Monitoring**.
2. Click **Add Agent**.
3. Enter a name. Leave the **Location** setting to the default **local://**.
4. Click **Save**.

## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Saved successfully.

Save + Add Agent

▼ local:// (local)

Name

local

Friendly name for your agent

Location

local://

URL for the agent. Leave blank for a local agent. For remote agents use the `tcp` scheme. The default agent port is `9001`. Example: `tcp://192.168.48.2:9001`

For more detailed instructions, see [Adding an agent](#).

### Configuring the monitors for basic

The basic configuration is targeting diagslave compiled with Debug enabled. You will therefore want the following monitors:

- **Gdb Monitor.** This will allow Peach Fuzzer to launch diagslave from within GDB so that GDB attaches to diagslave. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor.** Since MODBUS is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in creating a fix for the application. Any configuration that is fuzzing a network protocol should typically have this monitor.

To add and configure the monitors:

1. Click **Add Monitor**. In the pop-up, scroll down and select **Gdb**. Click **Ok**.
2. Under **Executable**, enter `/var/targets/modbus/diagslave/linux/diagslave` which is the location of diagslave's launcher. This will allow the monitor to launch the application when fuzzing starts. Do not change any of the other settings for this monitor.
3. Under **Arguments**, enter `-m tcp`.
4. Click **Save**.

▼ Gdb (Gdb)

Name

Gdb

Friendly name for your monitor

▼ Core Parameters

**Executable**

```
/var/targets/modbus/diagslave/linux/diagslave
```

Executable to launch

## Arguments

-m tcp

### Optional command line arguments

### Gdb Path

```
/usr/bin/gdb
```

Path to gdb

### ▼ When To Trigger

### Restart On Each Test

false

Restart process for each iteration

### Restart After Fault

false

Restart process after any fault occurs

### Start On Call

Start command on state model call

### Wait For Exit On Call

Wait for process to exit on state model call and fault if timeout is reached

➤ Advanced

5. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
6. Under **Device**, enter **lo** (for loopback, since diagslave is listening on the loopback interface)
7. Under **Filter**, enter **port ##ListenPort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
8. Click **Save**.

▼ NetworkCapture (Network Capture)

Name

Network Capture

Friendly name for your monitor

▼ Core Parameters

Device

lo

Device name for capturing on

Filter

port ##TargetPort##

PCAP Style filter

## Testing your configuration

You should always test your configuration to ensure that Peach Fuzzer is able to fuzz your application. This will run some control iterations to ensure that your application responds as expected to normal input.

1. Click **Test**
2. Click **Begin Test**
3. Once testing is complete, you will see the results. Correct any errors in your Variables or Monitoring if the test is not successful.
4. Once the test has passed, click **Continue**.

You can now fuzz the application.

Follow the steps under [Running the test](#) to start testing the application.



# Appendix A: Common Tasks

This section includes more detailed instructions on how to perform the more frequent tasks in this document.

## A.1. Adding an agent

An Agent runs either in-process of Peach Fuzzer or can be installed and run on a remote machine.

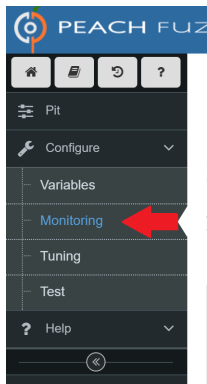


It is typically not necessary to configure multiple agents for the same machine. A single agent is capable of running multiple different monitors.

### A.1.1. Add a local agent

To add a local agent, follow these steps:

- 1. Click Monitoring



- 2. Click Add Agent

#### Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.



- 3. Enter a name for the agent e.g. `Local`. Leave the default value of `local://` for the agent's location.


## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Save + Add Agent

▼ local:// (local) ^ v x

 **Name**

Friendly name for your agent

**Location**

URL for the agent. Leave blank for a local agent. For remote agents use the `tcp` scheme. The default agent port is `9001`. Example: `tcp://192.168.48.2:9001`


### 4. Click Save

## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Saved successfully.

 Save + Add Agent

▼ local:// (local) ^ v x

**Name**

Friendly name for your agent

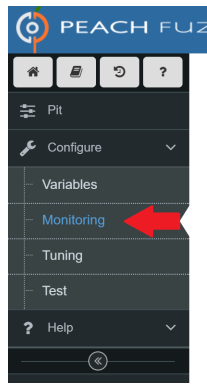
**Location**

URL for the agent. Leave blank for a local agent. For remote agents use the `tcp` scheme. The default agent port is `9001`. Example: `tcp://192.168.48.2:9001`

### A.1.2. Add a remote agent

Assume you have peachagent running on a host with the IP address 192.168.17.145. To add a remote agent to this host, follow these steps:

#### 1. Click Monitoring



## 2. Click Add Agent

### Monitoring


The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.


## 3. Enter a name for the agent e.g. **Remote**. Use **tcp://192.168.17.145** to indicate the agent is running on the remote host.

✓ tcp://192.168.17.145 (remote)



**Name**

Friendly name for your agent



**Location**

URL for the agent. Leave blank for a local agent. For remote agents use the **tcp** scheme. The default agent port is **9001**.  
Example: **tcp://192.168.48.2:9001**

## 4. Click Save

## Monitoring

The Monitoring data entry screen defines one or more Agents and one or more Monitors for the Pit.

Agents are host processes for monitors and publishers. Local agents can reside on the same machine as Peach, and can control the test environment through monitors and publishers. Remote agents reside on the test target, and can provide remote monitors and publishers.

Saved successfully.



Save

+ Add Agent

▼ tcp://192.168.17.145 (remote)

Name

remote

Friendly name for your agent

Location

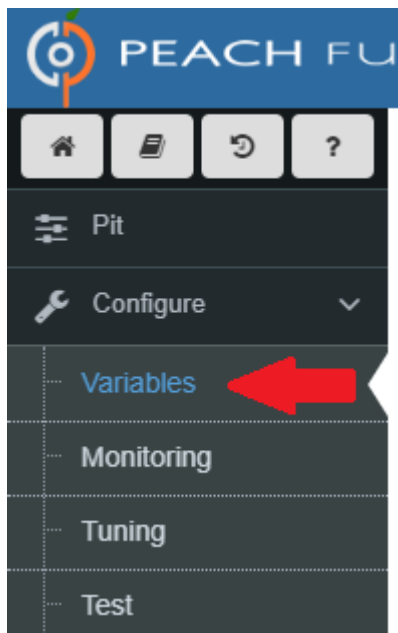
tcp://192.168.17.145

URL for the agent. Leave blank for a local agent. For remote agents use the `tcp` scheme. The default agent port is `9001`.  
Example: `tcp://192.168.48.2:9001`

## A.2. Using variables

Peach Fuzzer supports using variables for things such as parameters for monitors and values for other variables. All variables are surrounded by double hash marks e.g. `##`. It is generally recommended to use variables whenever possible.

Variables are defined under the **Variables** section of a configuration.



A variable is referred to by its **Key** value in this section. For example, this shows a variable called `TargetPort` that has a value of `20000`.

▼

Basic Configuration

Name	Key	Value
Target IPv4 Address	TargetIPv4	<div>127.0.0.1</div> <div>IPv4 address of the target machine.</div>
Target Port	TargetPort	<div>20000</div> <div>Port number the target machine uses to receive messages. The default value is '20000'.</div>

**Figure 1. The TargetPort variable is shown here.**

This variable could be used anywhere the Target Port is needed, such as an argument passed to the command of a process monitor or a PCAP expression on a Network Capture Monitor. To use the `TargetPort` variable elsewhere in the configuration, reference it as `##TargetPort##`.

▼

NetworkCapture (Network Capture)

Name

Network Capture

Friendly name for your monitor

▼

Core Parameters

Device

lo

Device name for capturing on

Filter

port ##TargetPort##

PCAP Style filter

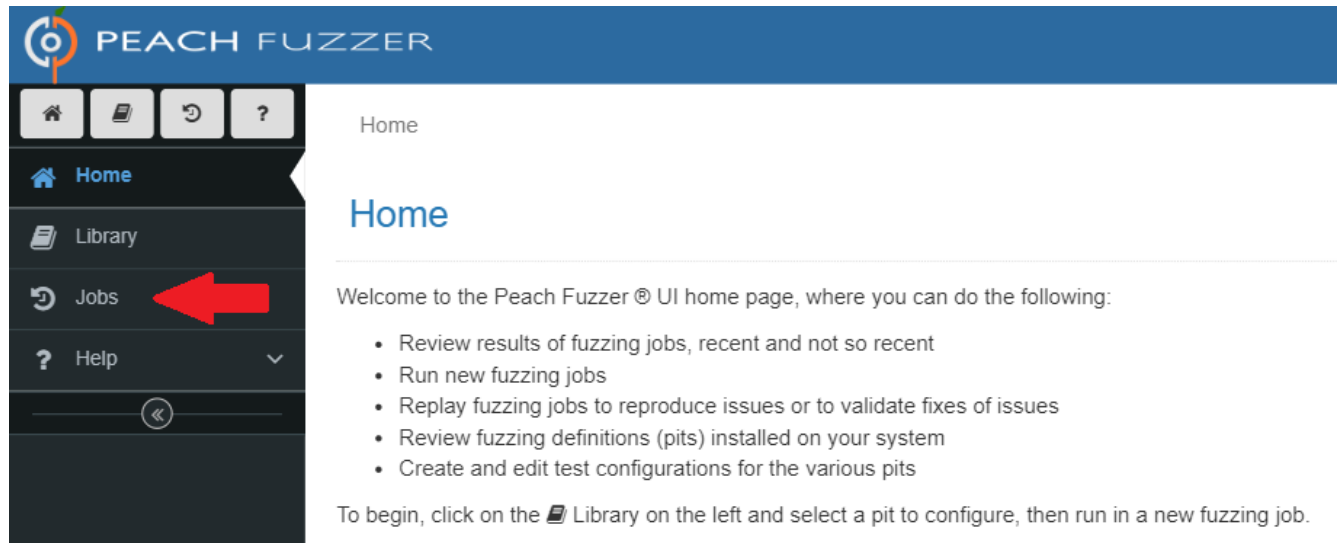
**Figure 2. The TargetPort variable is used here to set a Network Capture Monitor to capture all traffic for this configuration. In this example, traffic to and from port 20000 will be captured.**

If the value of a variable changes, it will automatically be applied everywhere the next time the configuration is run.

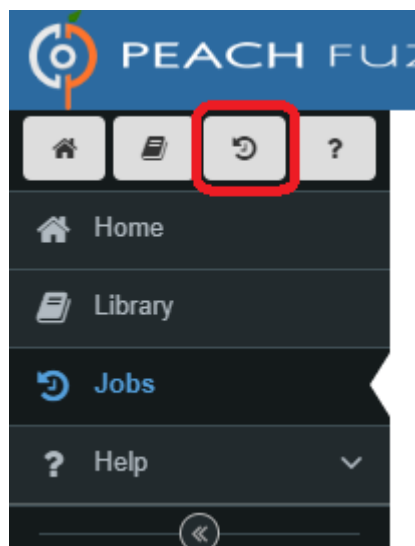
## A.3. Examining faults

Each job represents a single test run using a specific configuration. A job will show information on the duration of the job, the settings used to run that job, and the faults that were found. When you [run a test](#) on a configuration, a new job is created and you will see all the relevant information for that job. If you wish to view the information on a previous job, you can select a specific job from the Jobs page.

1. First, navigate to the Jobs page by clicking on the Jobs tab



or by clicking on the Jobs icon





2. Next, select the job you wish to view.



















Library
Jobs
Help

## Jobs

Here is a comprehensive list of the fuzzing jobs on this computer.

For any entry, you can perform the following actions:

- Click the  icon to view the report generated for the fuzzing session.
- Click the  icon to delete the job.

Name	Status	Start Time	Stop Time	Test Cases	Total Faults	Actions
Example-JPG-Advanced	Stopped	6/25/18 5:23 PM	6/25/18 5:23 PM	46	1	 
Example-JPG-Basic	Stopped	6/25/18 5:20 PM	6/25/18 5:21 PM	46	1	 
Example-PNG-Advanced	Stopped	6/25/18 5:19 PM	6/25/18 5:19 PM	11	1	 
Example-PNG-Basic	Stopped	6/25/18 5:19 PM	6/25/18 5:19 PM	11	1	 
Example-MODBUS-TCP Slave	Stopped	6/25/18 5:17 PM	6/25/18 5:17 PM	30	1	 
Example-MODBUS-TCP Master	Stopped	6/25/18 5:14 PM	6/25/18 5:15 PM	50	3	 
Example-SNMPv3 Server-Advanced	Stopped	6/25/18 5:11 PM	6/25/18 5:12 PM	10	5	 
Example-SNMPv3 Server-Basic	Stopped	6/25/18 5:06 PM	6/25/18 5:10 PM	10	5	 
Example-HTTP Server-Advanced	Stopped	6/25/18 5:04 PM	6/25/18 5:05 PM	30	2	 

3. The job you selected will now be displayed. Select a fault to view more information.

board
5

## Example-SNMPv3 Server-Advanced

This job has completed. [Click here to view the final report.](#)

6/25/18 5:11PM Start Time	00h 01m 06s Running Time
545 Test Cases/Hour	47918 Seed
10 Test Cases Executed	5 Total Faults

Edit Configuration
Replay Job

### Recent Faults

#	When	Monitor	Risk	Major Bucket	Minor Bucket	Download
9	6/25/18 5:12 PM	Process	heap use after free	1D70A0F1	AE468585	<a href="#">Download</a>
8	6/25/18 5:11 PM	Process	heap-use-after-free	A6468B29	AE468585	<a href="#">Download</a>
7	6/25/18 5:11 PM	Process	heap-use-after-free	98430BA7	AE468585	<a href="#">Download</a>
6	6/25/18 5:11 PM	Process	heap-use-after-free	38C77DA7	AE468585	<a href="#">Download</a>
3	6/25/18 5:11 PM	Process	heap-use-after-free	F204B8C6	AE468585	<a href="#">Download</a>

4. The selected fault will contain detailed information about the type of fault, how it was discovered, and the information collected from the various Monitors that were running when the fault

occurred.



PEACH FUZZER

Home

Faults

Metrics

Help

Dashboard

Faults

Metrics

Help

Home / Jobs / Example-SNMPv3 Server-Advanced / Faults / Test Case: 9

Test Case: 9

Fault Details

Test Case

Assets Archive

Reproducible

Title

When

Source

Risk

Major Bucket

Minor Bucket

Tested Fields

9

[Download all fault assets](#)

Yes

heap-use-after-free on address 0x6140000fe40 at pc 0x7f917b1ee935 bp 0x7ffe998a5260 sp 0x7ffe998a5260

6/25/18 5:12 PM

Process

heap-use-after-free

1D70A0F1

AE468585

Field	Mutator
SNMP.MessageV3.Message.Value.msgGlobalData.Value.msgId.Value	Data
SNMP.MessageV3.Message.Value.msgData.Value.data.Choice.GetRequest-PDU.PDU.Value	Data
SNMP.MessageV3.Message.Value.msgGlobalData.Value.msgId.length	Number
SNMP.MessageV3.Message.Value.msgVersion	Data

Description

```
==3938==ERROR: AddressSanitizer: heap-use-after-free on address 0x6140000fe40 at pc 0x7f917b1ee935 bp 0x7ffe998a5260 sp 0x7ffe998a5260
READ of size 100 at 0x6140000fe40 thread T0
#0 0x7f917b1ee934 in __asan_memcpy (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x8c934)
#1 0x416bea in memcpy /usr/include/x86_64-linux-gnu/bits/string3.h:53
#2 0x416bea in handle_request snmp-agent/agent-incoming.c:215

0x6140000fe40 is located 0 bytes inside of 400-byte region [0x6140000fe40,0x6140000ff40)
freed by thread T0 here:
#0 0x7f917b1fa2ca in __interceptor_free (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
#1 0x416bda in handle_request snmp-agent/agent-incoming.c:214
#2 0x202315901525c92 (<unknown module>)

previously allocated by thread T0 here:
#0 0x7f917b1fa602 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
#1 0x416bcf in handle_request snmp-agent/agent-incoming.c:212
#2 0x202315901525c92 (<unknown module>)

SUMMARY: AddressSanitizer: heap-use-after-free ??:0 __asan_memcpy
Shadow bytes around the buggy address:
0x0c287fff9f70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c287fff9f80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c287fff9f90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c287fff9fa0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c287fff9fb0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c287fff9fc0: fa fa fa fa fa fa fa fa[fd]fd fd fd fd fd fd fd
0x0c287fff9fd0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
0x0c287fff9fe0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
0x0c287fff9ff0: fd fd fd fd fd fd fd fd fd fd fa fa fa fa fa fa
0x0c287fffa000: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
```

Fault Assets

Test Case I/O

Name	
<a href="#">#1 - TX - Initial.GetRequest</a>	
<a href="#">#2 - RX - Initial.Report</a>	
<a href="#">#3 - TX - Initial.GetRequest2</a>	

Monitoring Assets

local

Process (Process)

Name	
<a href="#">description.txt</a>	2.6
<a href="#">stderr.log</a>	2.6

Other Assets


Name	
<a href="#">fault.json</a>	

Original Fault Assets

40

5. You can download all the captured data by clicking **Download all fault assets**.

## Test Case: 9

Fault Details	
Test Case	9
Assets Archive	<a href="#">Download all fault assets</a> 
Reproducible	Yes
Title	heap-use-after-free on address 0x61400000fe40 at pc 0x7f
When	6/25/18 5:12 PM
Source	Process

## A.4. Downloading the final report



Each job contains a final report detailing an overview of all the findings from that job. You can access this report several different ways:








- From the Jobs page, click the **Report** icon for any job in the list to download the report for that job

### Jobs

Here is a comprehensive list of the fuzzing jobs on this computer.

For any entry, you can perform the following actions:

- Click the  icon to view the report generated for the fuzzing session.
- Click the  icon to delete the job.

↕ Name	↕ Status	▼ Start Time	↕ Stop Time	↕ Test Cases	↕ Total Faults	Actions
Example-JPG-Advanced	Stopped	6/25/18 5:23 PM	6/25/18 5:23 PM	46	1 	 
Example-JPG-Basic	Stopped	6/25/18 5:20 PM	6/25/18 5:21 PM	46	1	 
Example-PNG-Advanced	Stopped	6/25/18 5:19 PM	6/25/18 5:19 PM	11	1	 

- If you are already viewing a job, click the link at the top of the page

## Example-SNMPv3 Server-Advanced

This job has completed.  [Click here to view the final report.](#)



<b>6/25/18 5:11PM</b> Start Time	<b>00h 01m 06s</b> Running Time
<b>545</b> Test Cases/Hour	<b>47918</b> Seed
<b>10</b> Test Cases Executed	<b>5</b> Total Faults

[Edit Configuration](#)[Replay Job](#)

### Recent Faults

▼ #	↕ When	↕ Monitor	↕ Risk	↕ Major Bucket	↕ Minor Bucket	Download
9	6/25/18 5:12 PM	Process	heap-use-after-free	1D70A0F1	AE468585	<a href="#">Download</a>