PEACH
FUZZER

Peach Fuzzer
Client-led Trial Guide

Peach Tech v0.0.0

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. Create configurations for each pit in your trial for the applications on the target VM.
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

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. Setting up your Client-led trial

The Target VM ISO image contains various applications pre-installed. It also runs the Peach Agent so that you can easily configure the pits to run those applications on the VM.

3.1. Prerequisites

In order to run your trial of Peach Fuzzer you will first need to complete the following steps:

1. Install Peach Fuzzer on your local machine and activate your license. See the Peach Fuzzer Installation Guide for instructions on how to install on your platform.
2. Install software capable of running a Virtual Machine. VMWare Workstation and VirtualBox will both work, although you may use any VM software that is capable of creating a VM from an ISO on your platform.
3. Create a Virtual Machine from the TargetVM ISO you downloaded from Portal.
4. Configure networking with the Target VM so that your installed Peach Fuzzer instance can communicate with the VM by IP address and your Target VM can also communicate back to your installed Peach Fuzzer instance.

It is most likely that you will want to configure the networking mode of your VM software to `NAT` instead of `Bridged`.

5. Note the IP address of the Virtual Machine as this will be required for configuring most of the pits.

If you have any trouble installing Peach Fuzzer, creating the Target VM, or ensuring connectivity between the Target VM and Peach Fuzzer, please contact support@peach.tech for assistance.

If you need to connect to the Target VM, you can use ssh with the user name `peach` and the password `peach`.

4. Sample Configurations

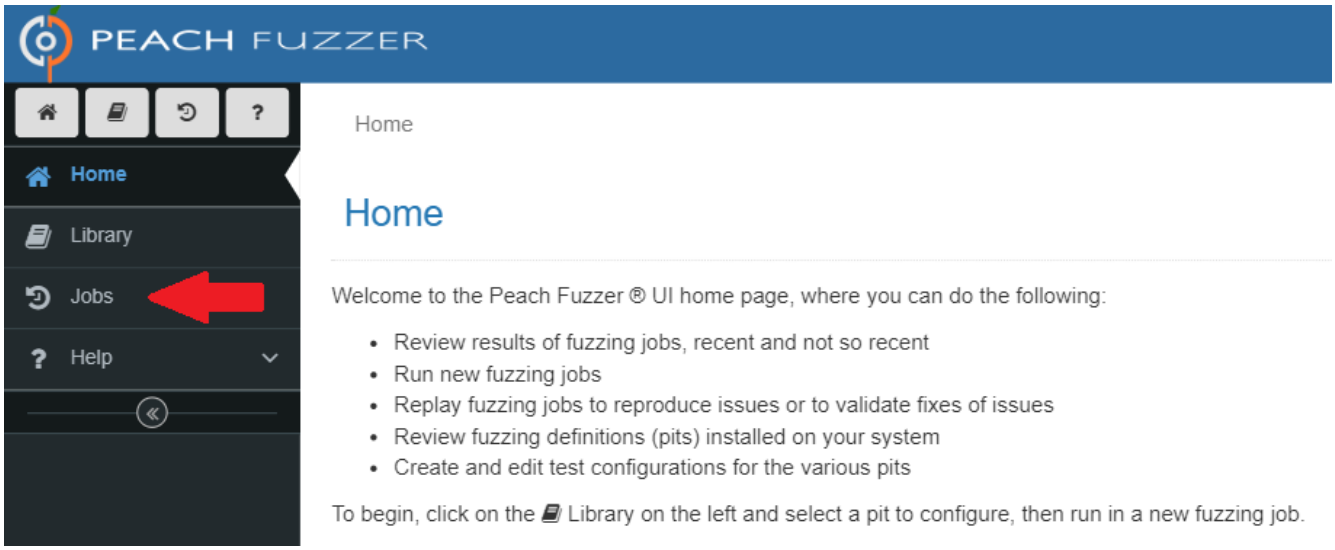
The following are sample configurations for several different protocols as well as file types that Peach Fuzzer can fuzz. You can run the sample configurations with the supplied seed values in this guide. Each section below has instructions for how to set up the sample configuration, including a brief 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.

For the purposes of this document we will assume the target VM has an IP address of `192.168.17.145` and this IP address will be used wherever the host's IP address is required. If you see this value used for an IP address in a configuration, replace it with the actual IP address of your VM.

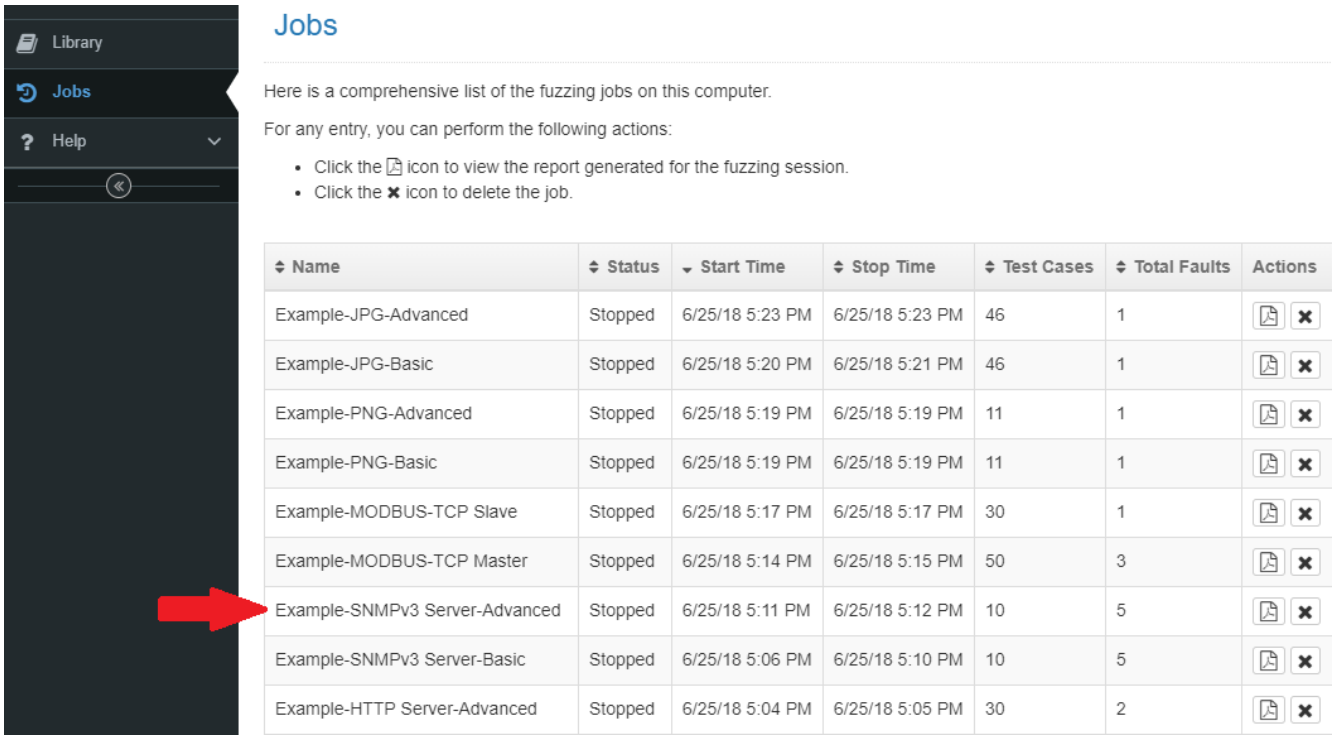
4.1. Viewing the Sample Job Results

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.



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

You will need to configure the pits to run with the target VM instance you have created. Select a PIT listed below for detailed instructions on how to configure the pit for a test run. The exact Configurations available will depend on what Pits are included with your trial license. . Enter the appropriate values for **Seed** and **Stop Test Case**. . 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.

This pit configuration requires that opendnp3 be able to communicate from the target VM back to the host operating system where Peach Fuzzer is running. As a result, the GDB or Process monitor will be configured to pass your host operating system's IP address as an argument.

4.3.1. Configuring the test

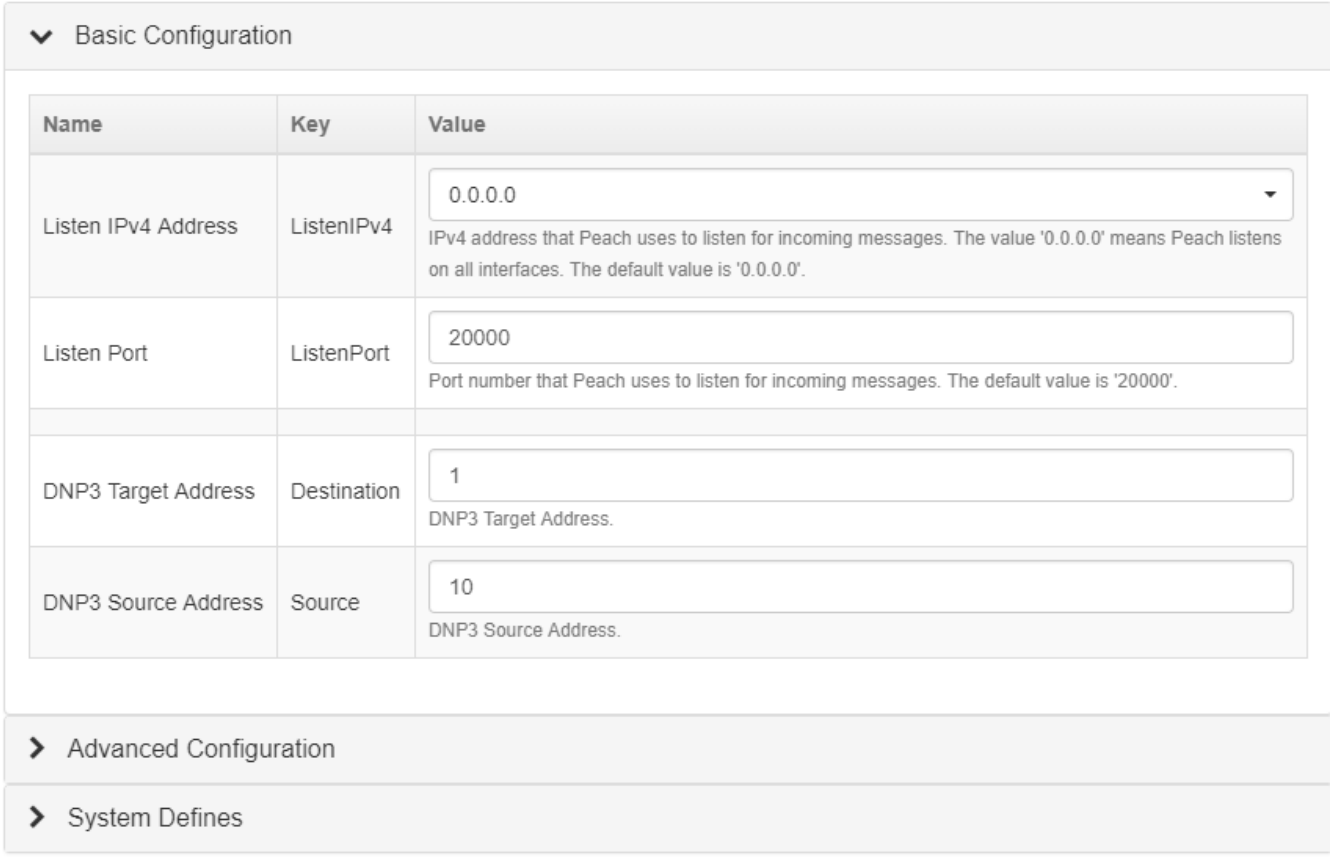
These steps will create the same configuration as is in use in the DNP3_Master configuration that is already present on the trial instance. The steps are the same for both the basic and advanced configuration except where indicated. To create the configuration:

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:

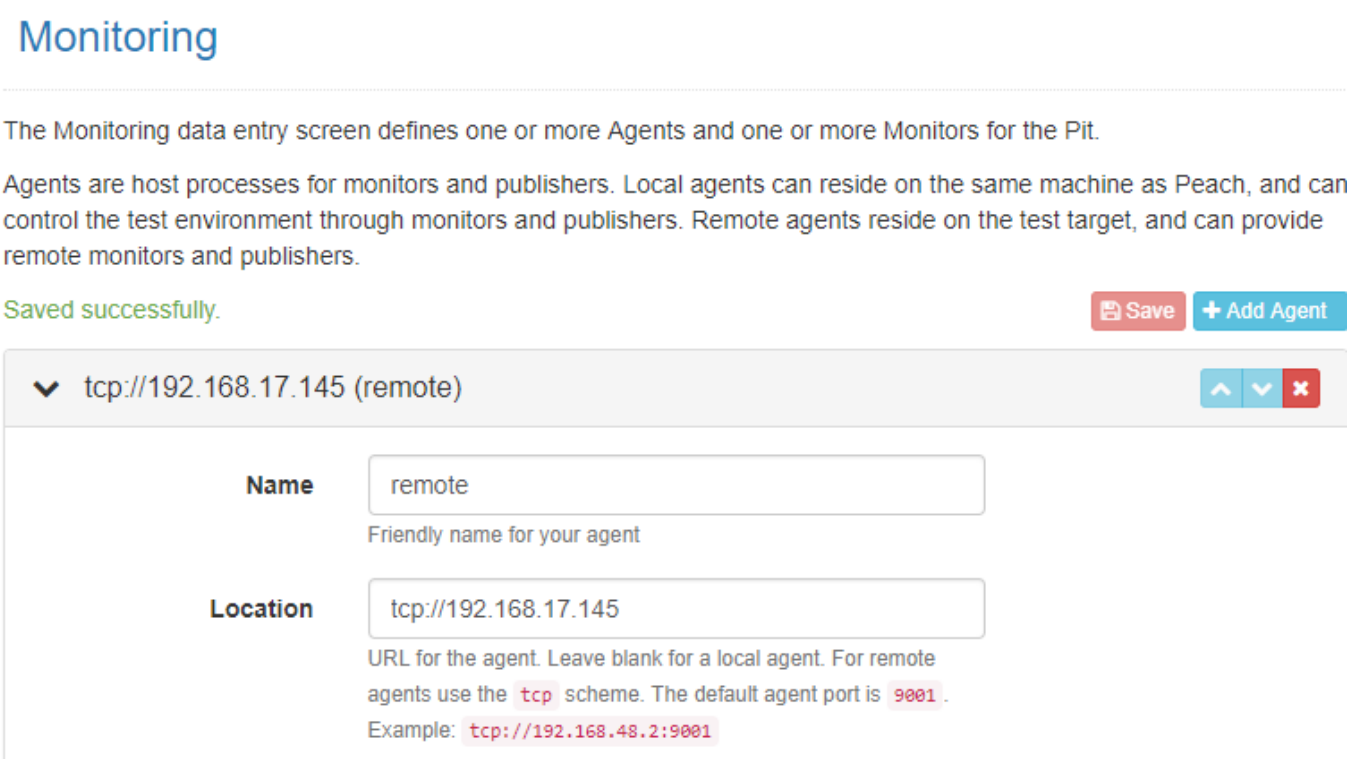
1. Click **Configure Variables**.
2. Configure your variables as appropriate for your application. The following should be used for opendnp3 running on the trial instance:
    a. Listen IPv4 Address: set to **0.0.0.0** so that Peach Fuzzer will listen for incoming connections on all network interfaces.
    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**.



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 remote agent running on the Target VM is required.

1. Click **Monitoring**.
2. Click **Add Agent**.
3. Enter a name. Set the **Location** setting to include the IP address of your target VM e.g. **tcp://192.168.17.145**
4. Click **Save**.



For more detailed instructions, see Adding an agent.

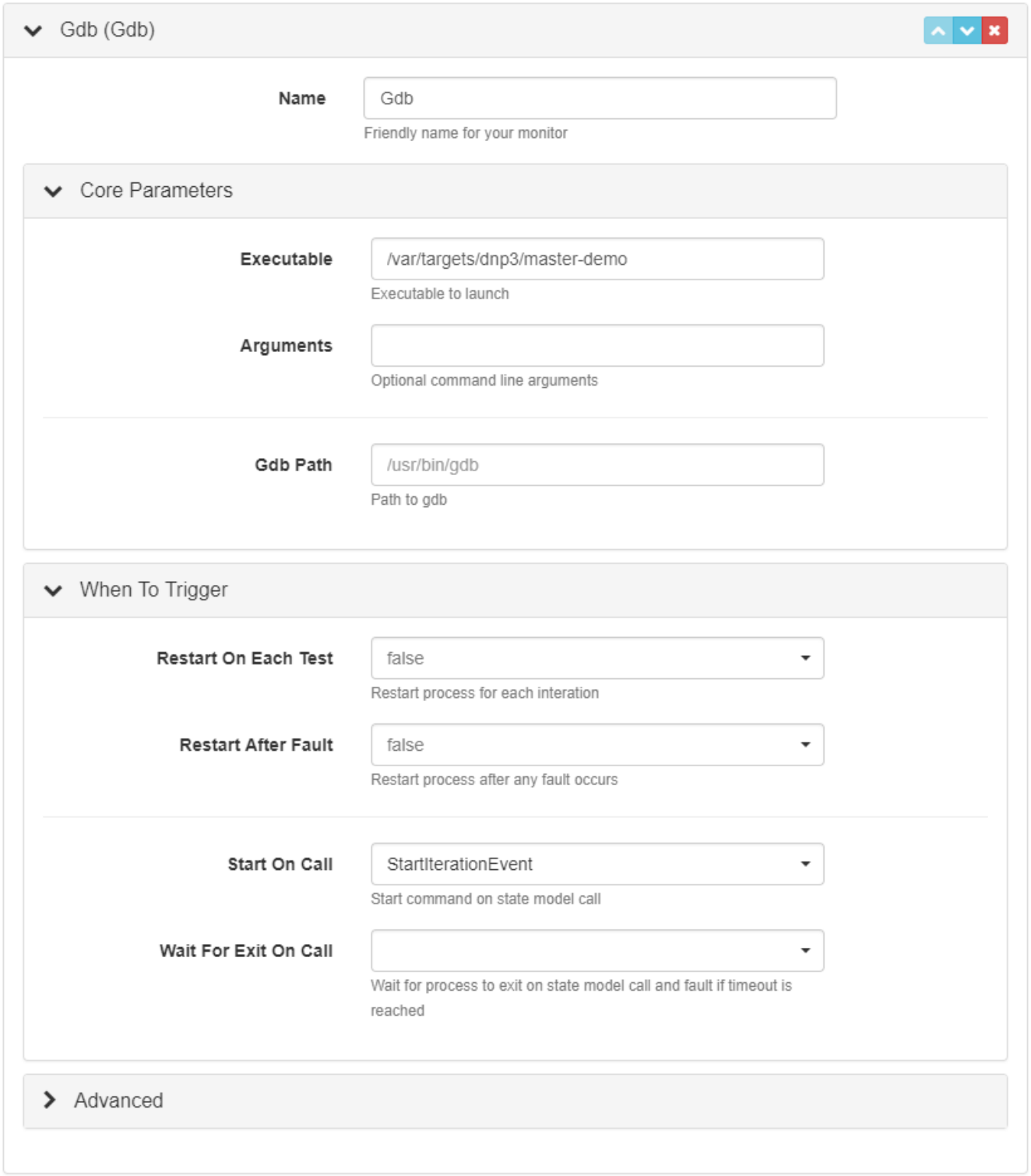
Configuring the monitors for basic

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

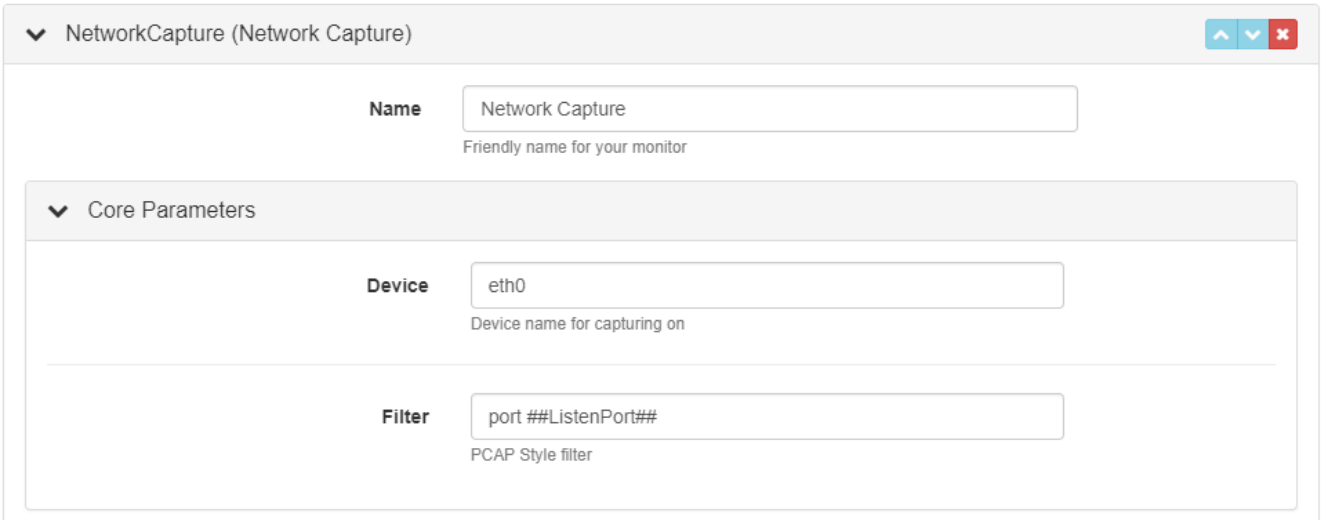
- **Gdb Monitor**. This will allow Peach Fuzzer to launch opendnp3 from within GDB so that GDB attaches to opendnp3. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor**. Since DNP3 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/master-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. Under **Arguments**, enter **the IP address of the Host operating system, e.g. 10.0.1.127**.
4. Click **Save**.



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 **eth0** (opendnp3 is listening on the target VM's eth0 interface)
7. 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.
8. 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:

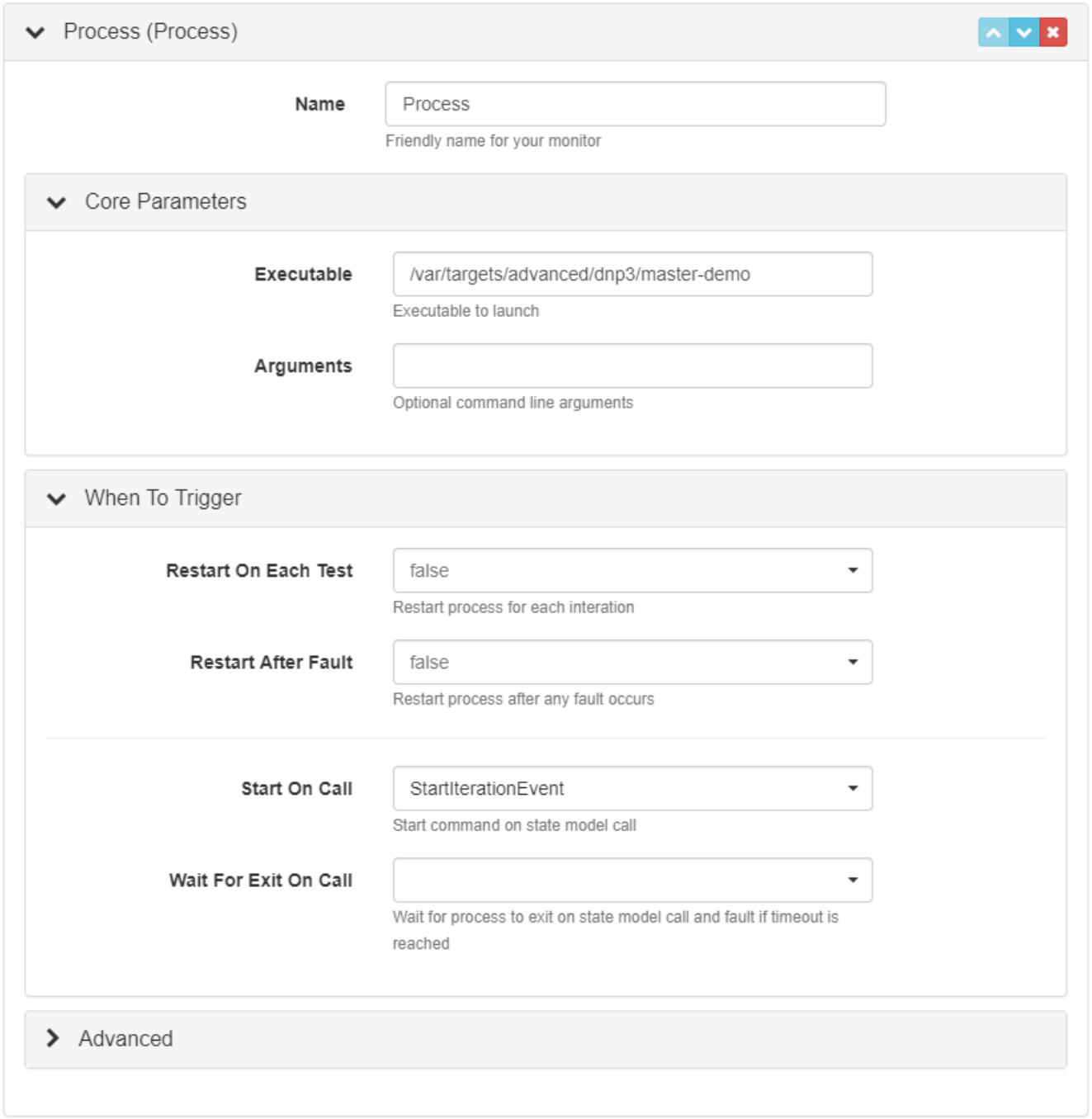
- **Process Monitor**. This will allow Peach Fuzzer to launch opendnp3. Because opendnp3 is compiled with ASan, the Process Monitor will detect the program exited due to an error and gather the output from ASan about the nature of the crash.

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

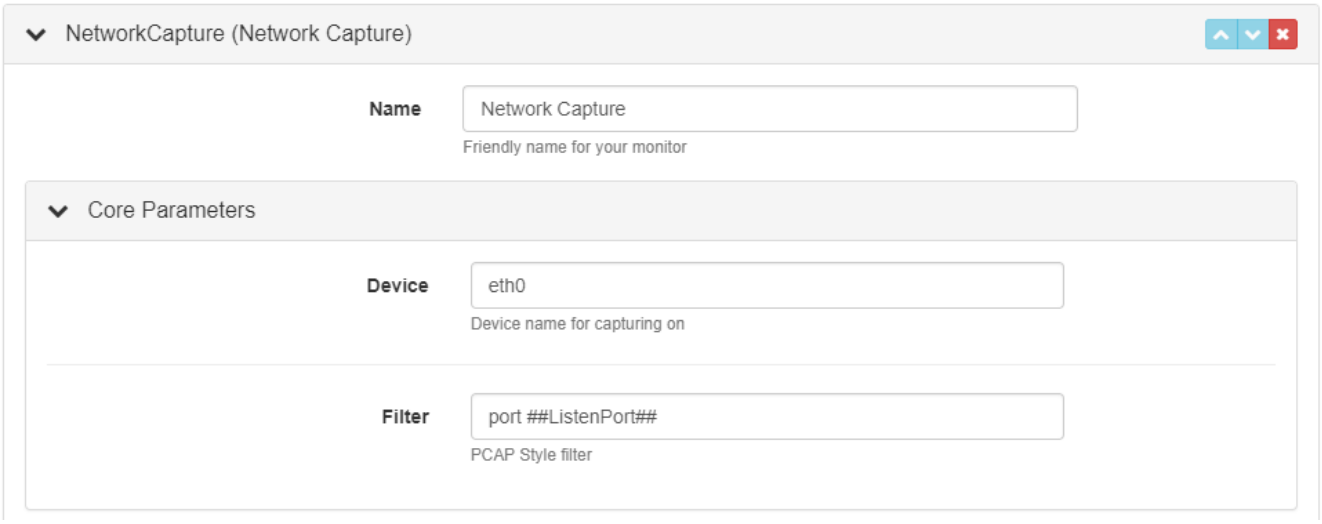
- **Network Capture monitor**. Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in 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. Under **Arguments**, enter **the IP address of the Host operating system, e.g. 10.0.1.127**.
4. Do not change any of the other settings for this monitor.
5. Click **Save**.



6. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
7. Under **Device**, enter **eth0** (opendnp3 is listening on the target VM's eth0 interface)
8. 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.
9. Click **Save**.



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**.

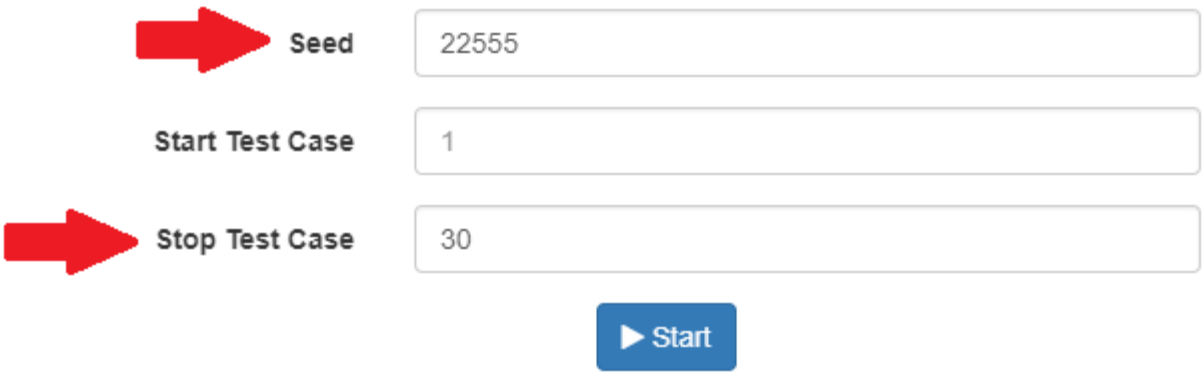
You can now fuzz the application.

Follow the steps under Running the test to start testing the application.

4.3.2. Running the test

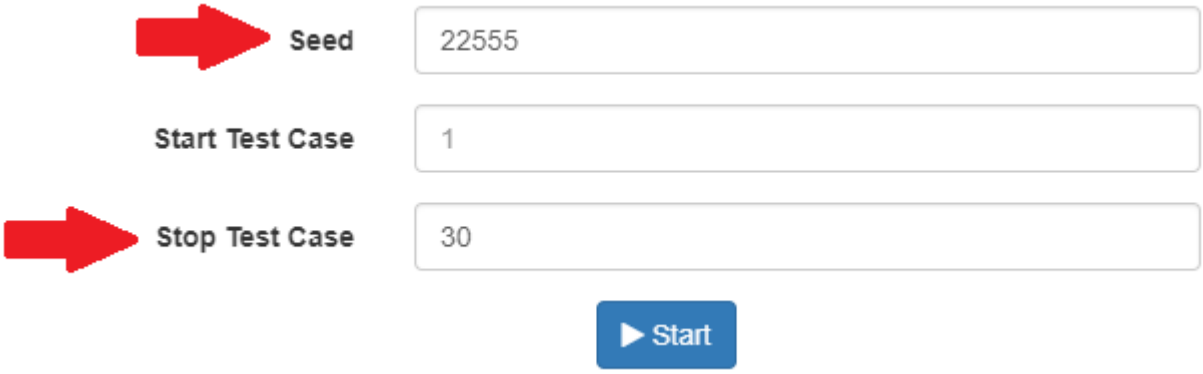
To run the pre-configured basic test:

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**.



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**.



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. Configuring the test

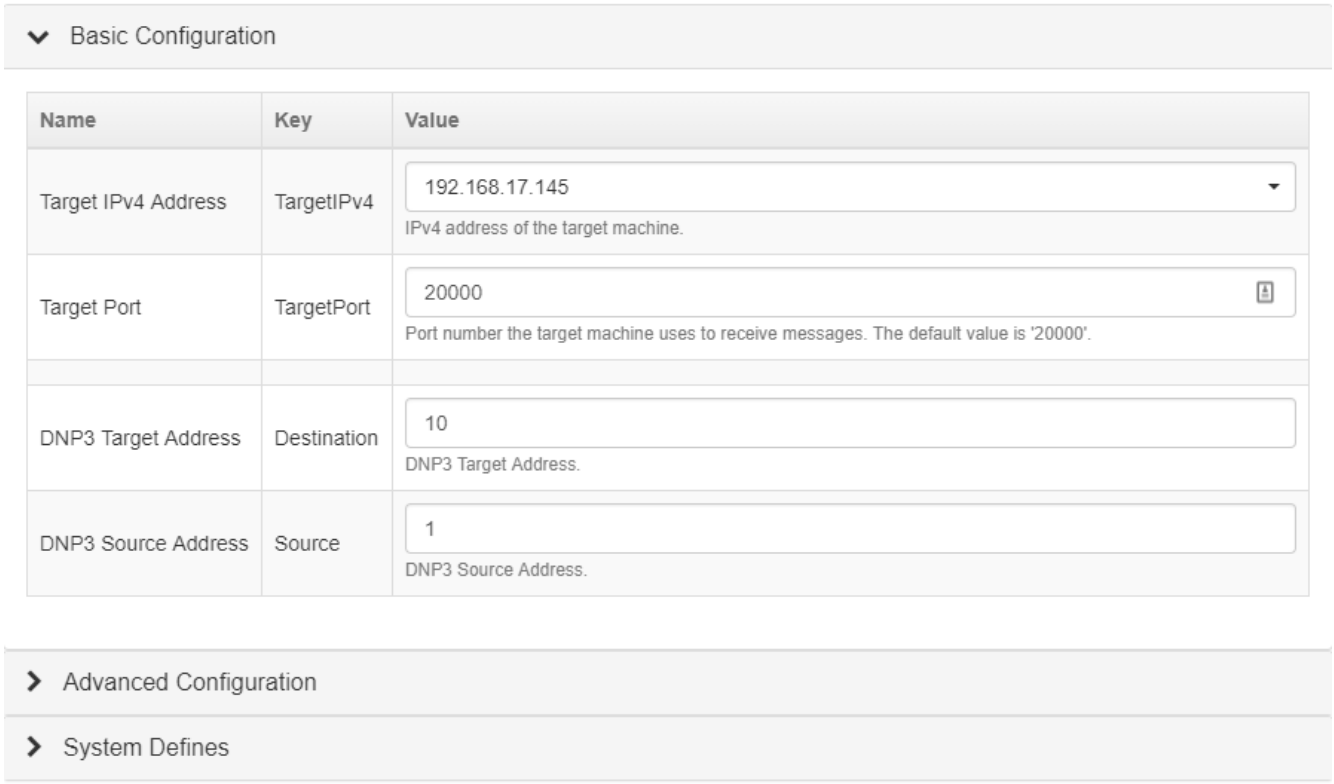
These steps will create the same configuration as is in use in the DNP3_Slave configuration that is already present on the trial instance. The steps are the same for both the basic and advanced configuration except where indicated. To create the configuration:

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 the IP address of the target VM, e.g. **192.168.17.145**
   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**.

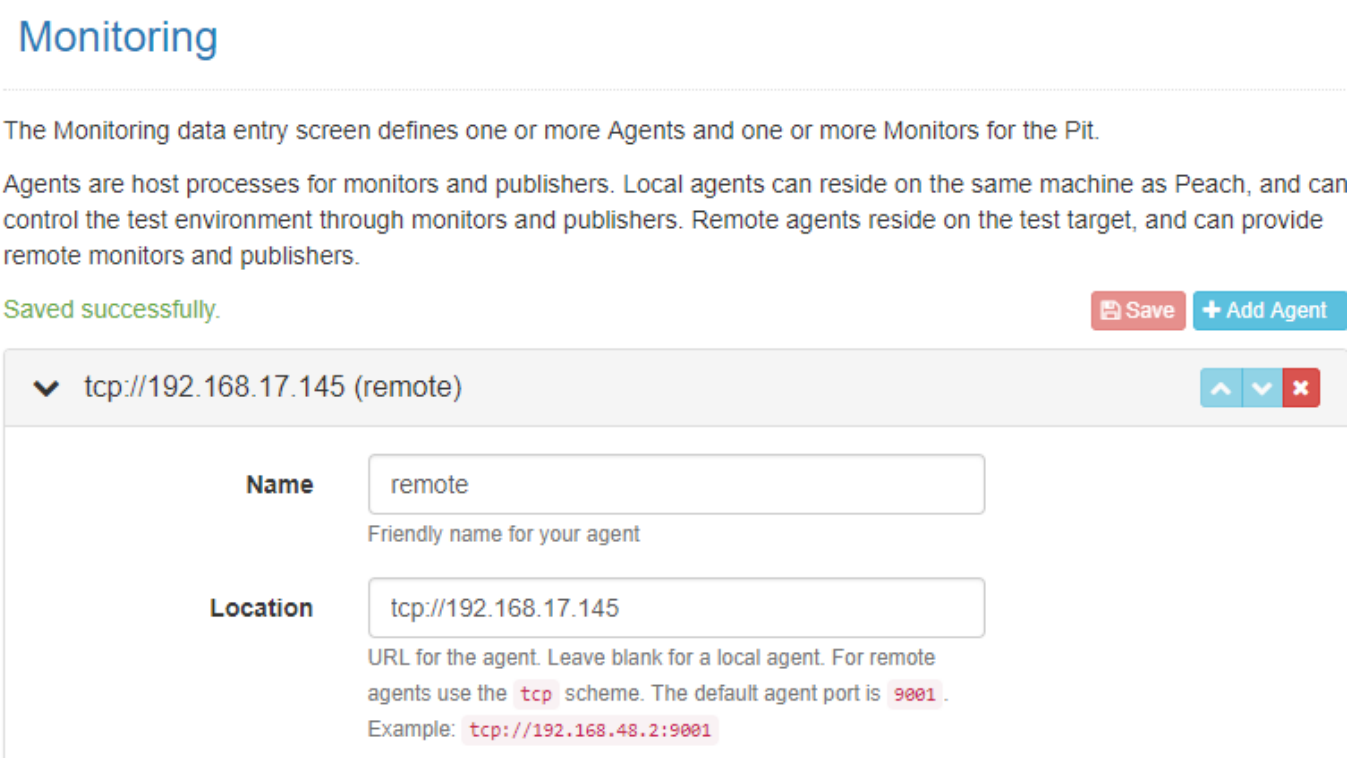


Configuring Agents

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 remote agent running on the Target VM is required.

1. Click **Monitoring**.

2. Click **Add Agent**.
3. Enter a name. Set the **Location** setting to include the IP address of your target VM e.g. **tcp://192.168.17.145**
4. Click **Save**.



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:

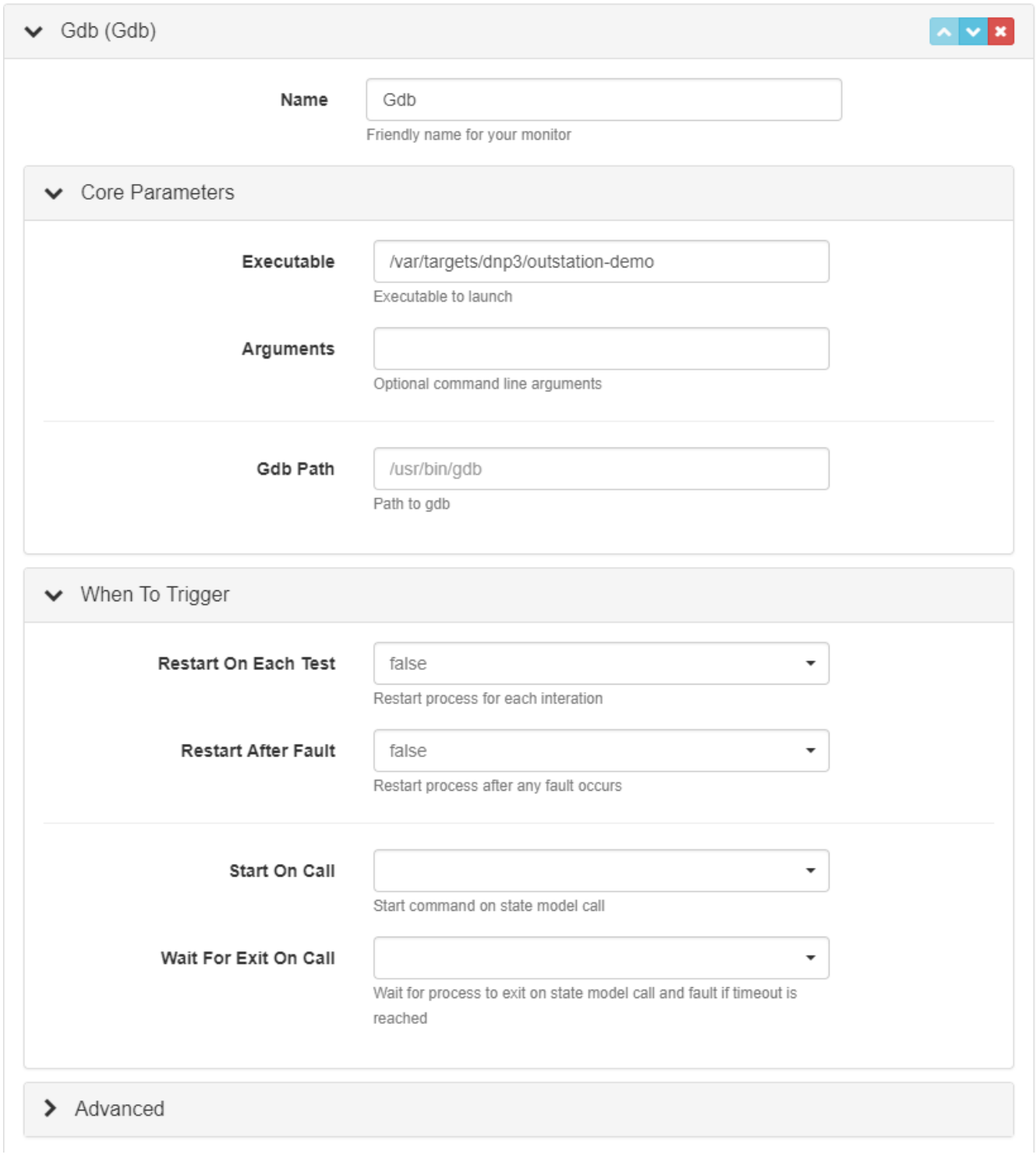
- **Gdb Monitor**. This will allow Peach Fuzzer to launch opendnp3 from within GDB so that GDB attaches to opendnp3. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor**. Since DNP3 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

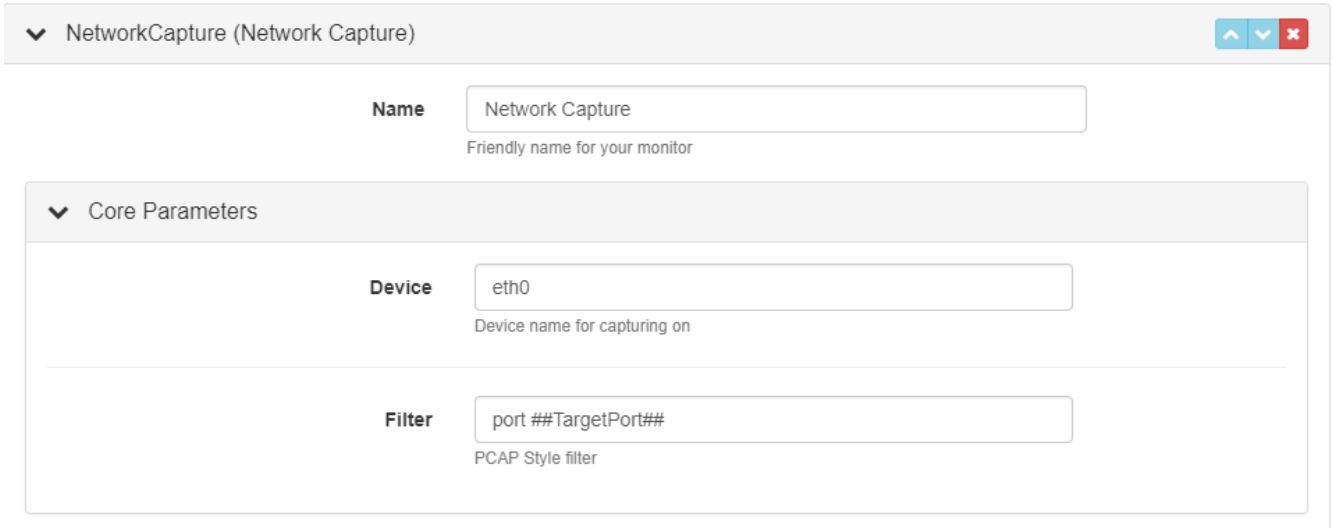
any of the other settings for this monitor.

3. Click **Save**.



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 **eth0** (opendnp3 is listening on the target VM's eth0 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:

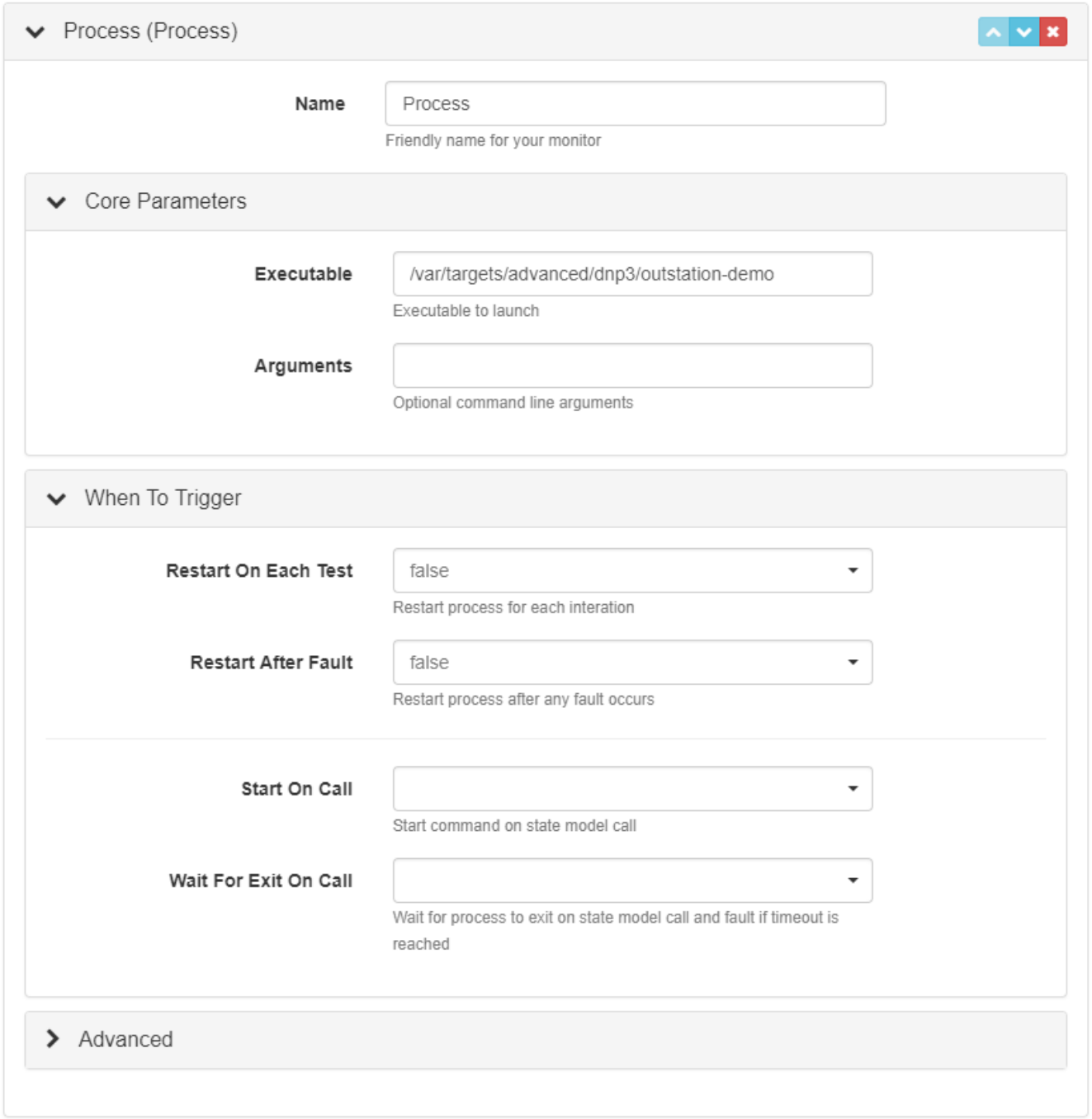
- **Process Monitor**. This will allow Peach Fuzzer to launch opendnp3. Because opendnp3 is compiled with ASan, the Process Monitor will detect the program exited due to an error and gather the output from ASan about the nature of the crash.

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

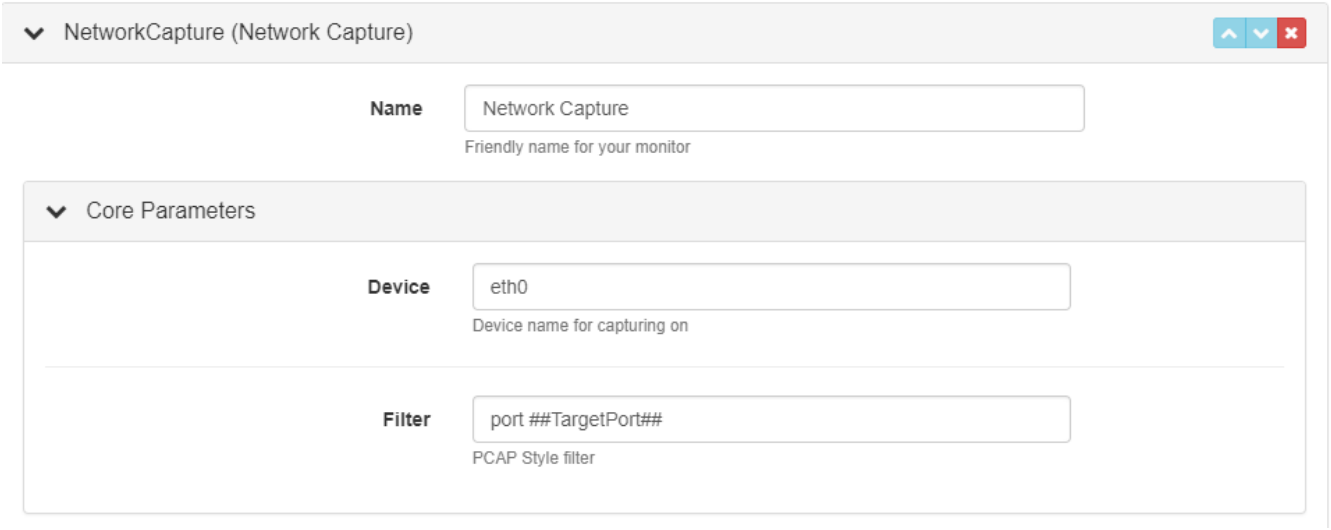
- **Network Capture monitor**. Since DNP3 is a network protocol, this monitor will allow Peach Fuzzer to capture the actual data that was sent and received as a pcap. This will help determine what may have caused a fault and could also be useful in trying to create a repro or test case that can aid in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



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 **eth0** (opendnp3 is listening on the target VM's eth0 interface)
7. 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.
8. Click **Save**.



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.

4.4.2. 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**.

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**.

4.5. MODBUS

This configuration will test an application using MODBUS.

4.5.1. 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

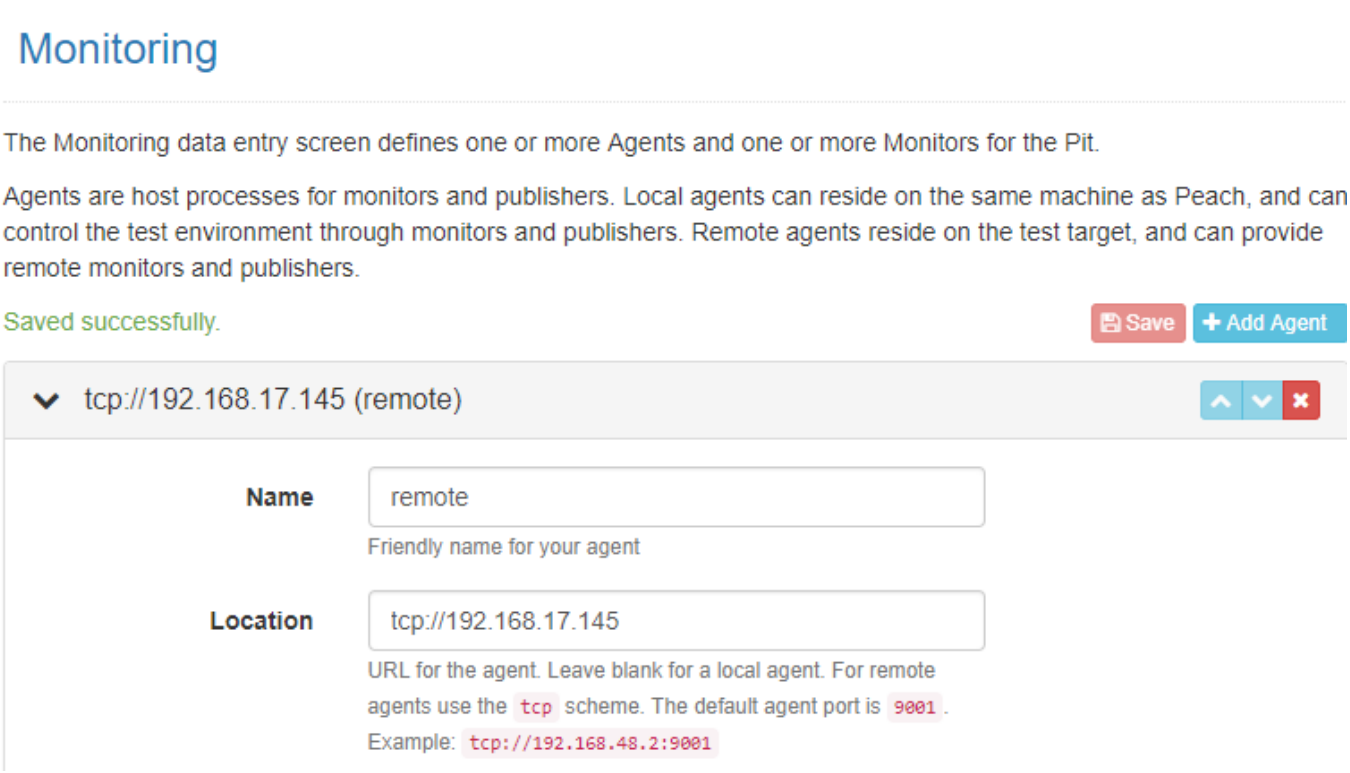
The first thing you need to configure are the variables that control how Peach Fuzzer will test the application. Follow these steps to create a working configuration on your trial instance:

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**.

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 remote agent running on the Target VM is required.

1. Click **Monitoring**.
2. Click **Add Agent**.
3. Enter a name. Set the **Location** setting to include the IP address of your target VM e.g. **tcp://192.168.17.145**
4. Click **Save**.



For more detailed instructions, see Adding an agent.

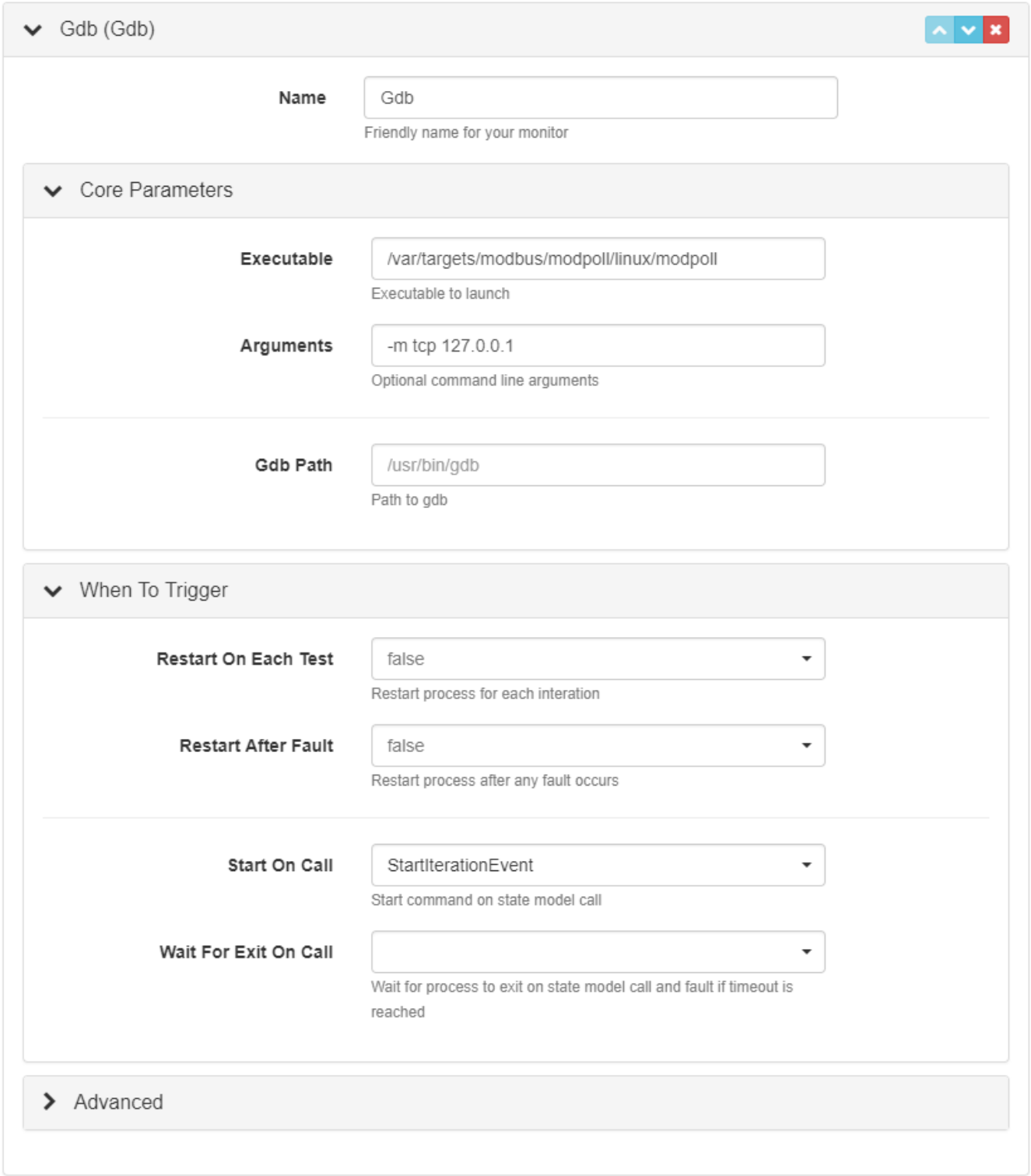
Configuring the monitors for basic

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

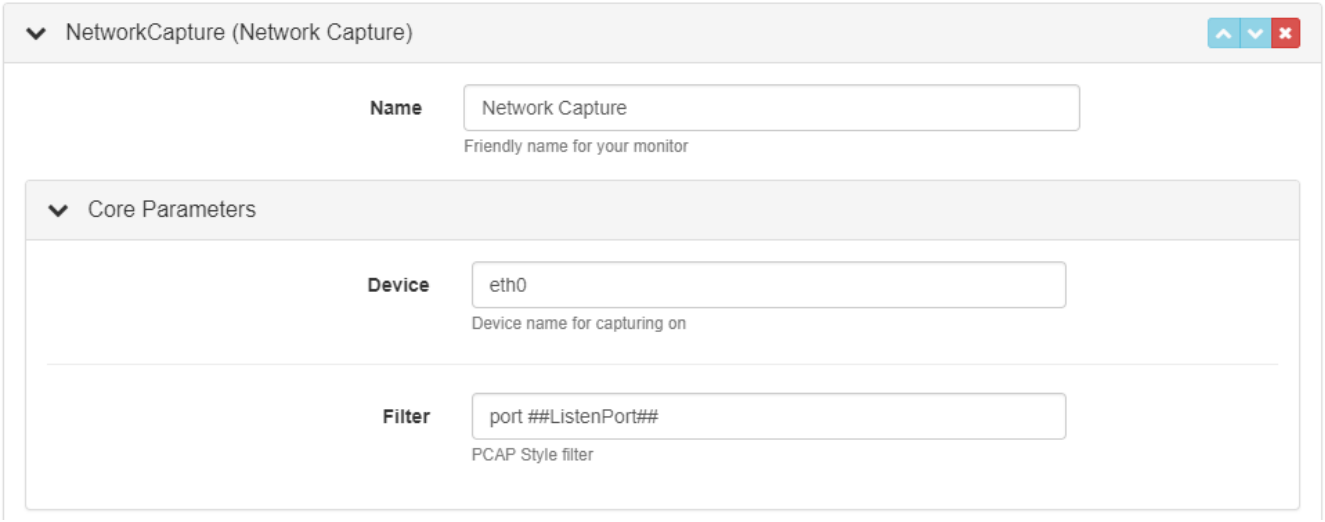
- **Gdb Monitor**. This will allow Peach Fuzzer to launch modpoll from within GDB so that GDB attaches to modpoll. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor**. Since MODBUS 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 10.0.1.127**.
4. Click **Save**.



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 **eth0** (modpoll is listening on the target VM's eth0 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**.



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.

4.5.2. 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**.

4.6. MODBUS-Slave

This configuration will test an application using MODBUS.

4.6.1. 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**.

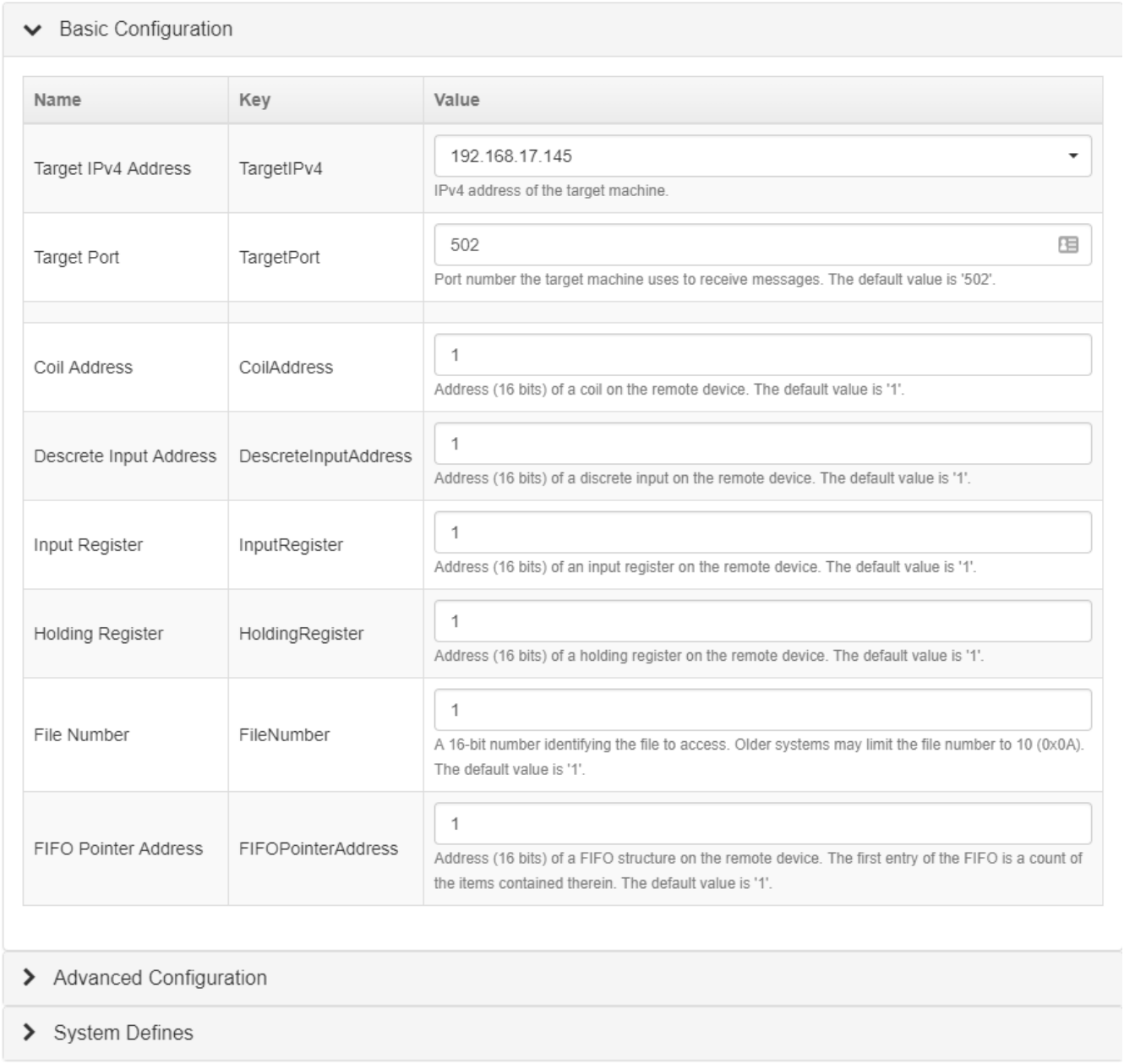
Configuring Variables

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 the IP address of the target VM e.g. **192.168.17.145** 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. Descrete 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**.

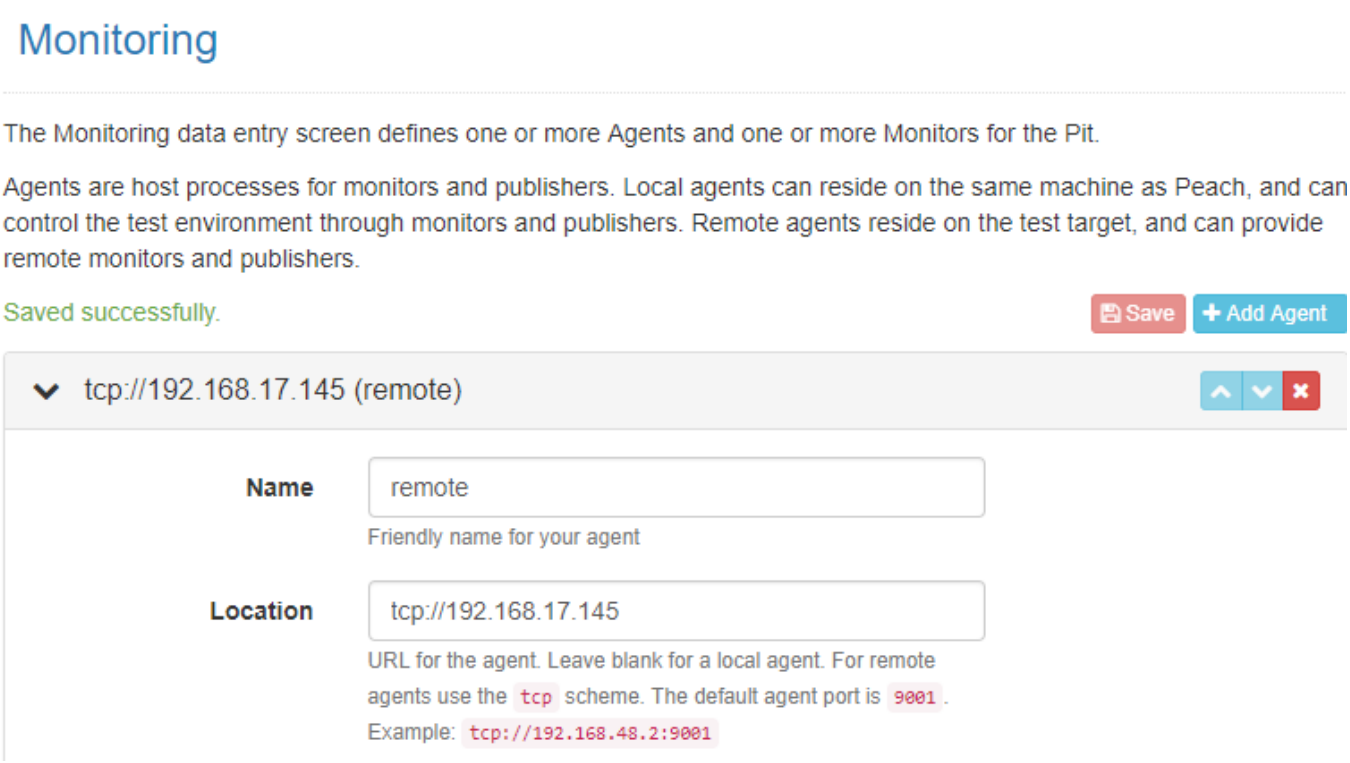


Configuring Agents

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 remote agent running on the Target VM is required.

1. Click **Monitoring**.
2. Click **Add Agent**.
3. Enter a name. Set the **Location** setting to include the IP address of your target VM e.g. **tcp://192.168.17.145**

4. Click **Save**.



For more detailed instructions, see Adding an agent.

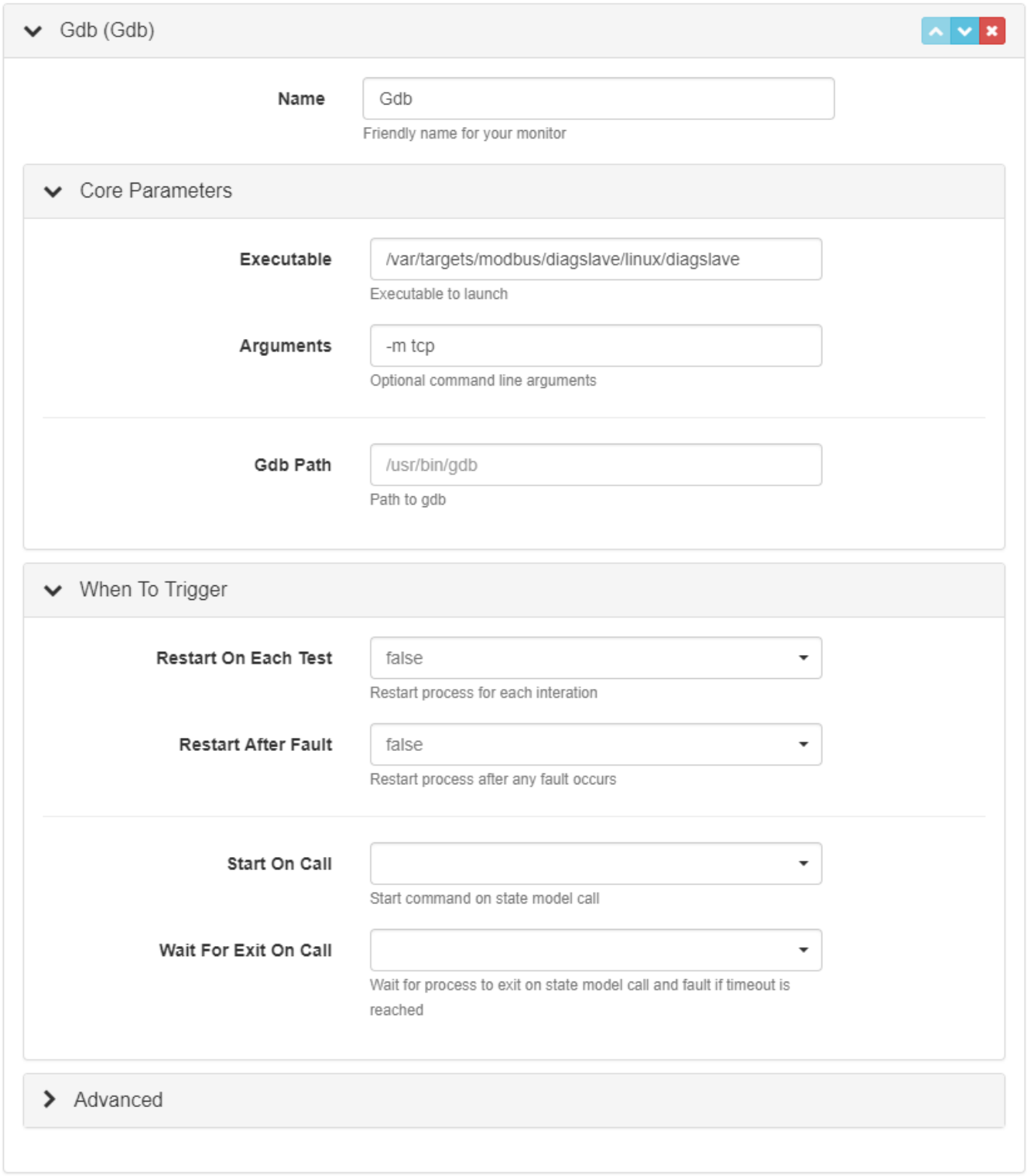
Configuring the monitors for basic

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

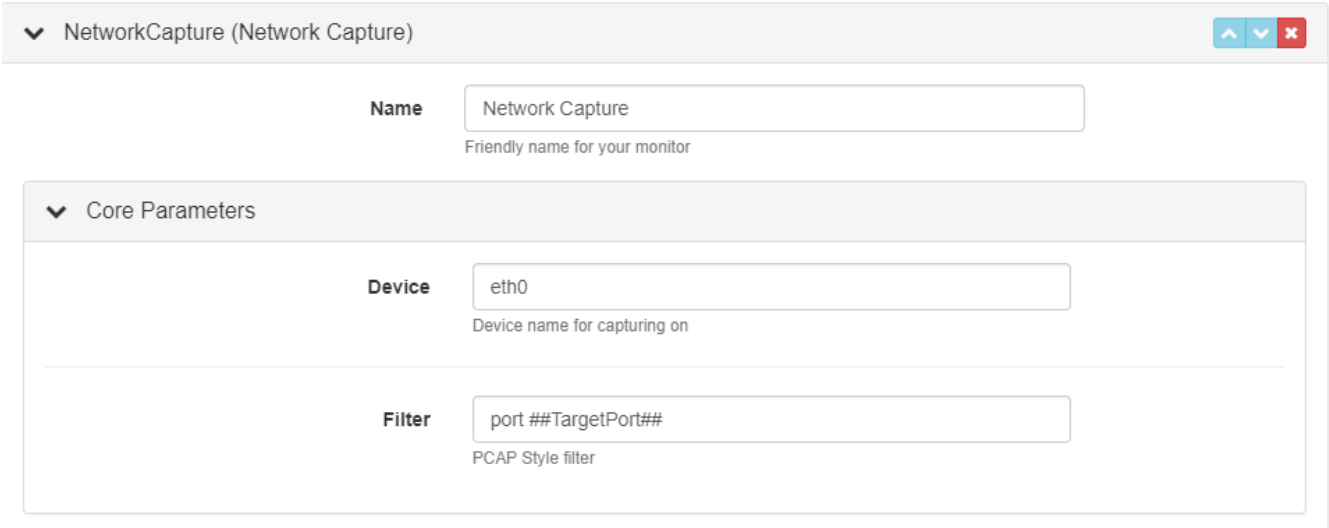
- **Gdb Monitor**. This will allow Peach Fuzzer to launch diagslave from within GDB so that GDB attaches to diagslave. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **Network Capture monitor**. Since MODBUS 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**.



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 **eth0** (diagslave is listening on the target VM's eth0 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**.



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.

4.6.2. 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

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

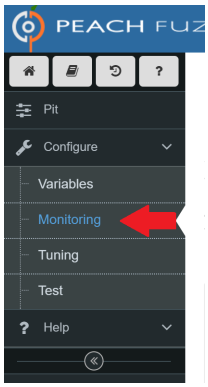
An Agent runs either in-process of Peach Fuzzer or can be installed and run on a remote machine.

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

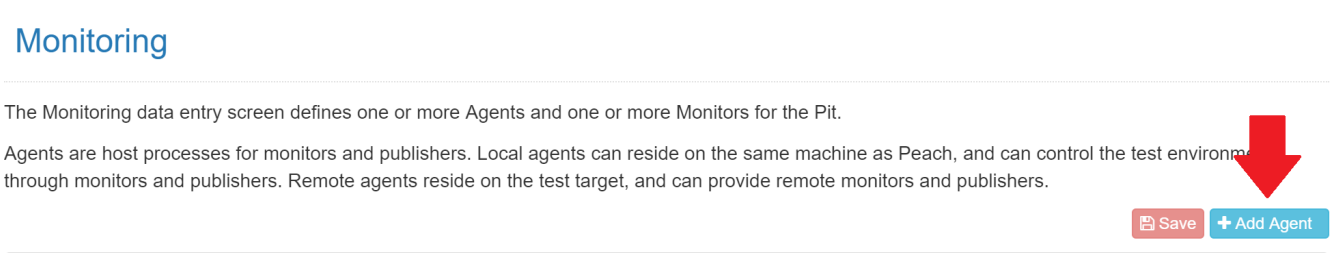
A.1.1. Add a local agent

To add a local agent, follow these steps:

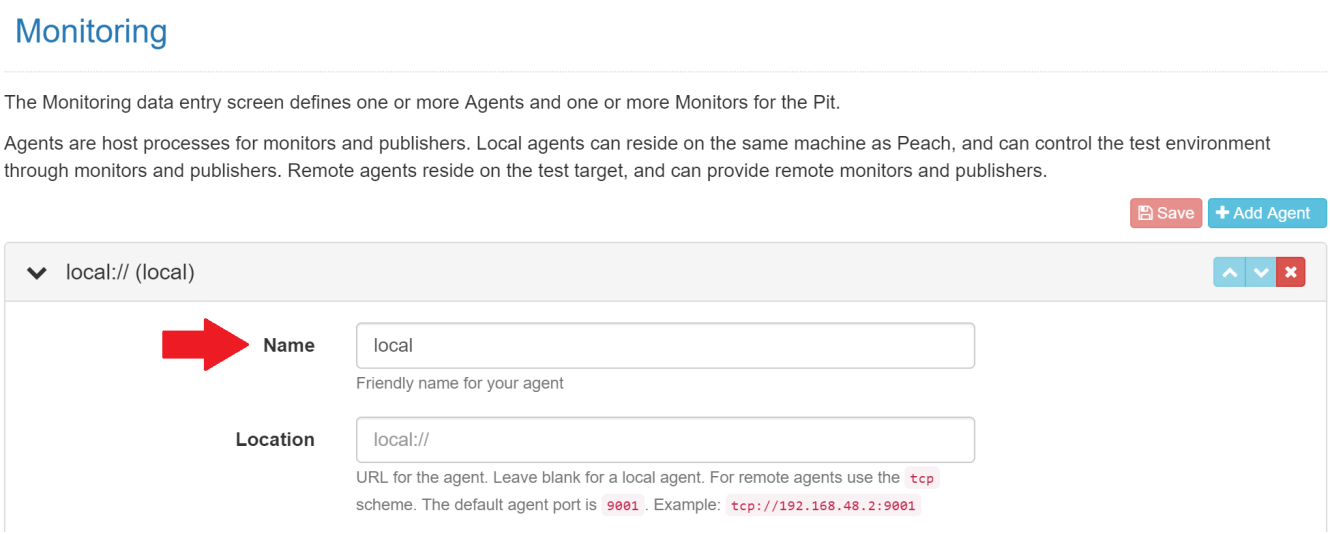
1. Click Monitoring



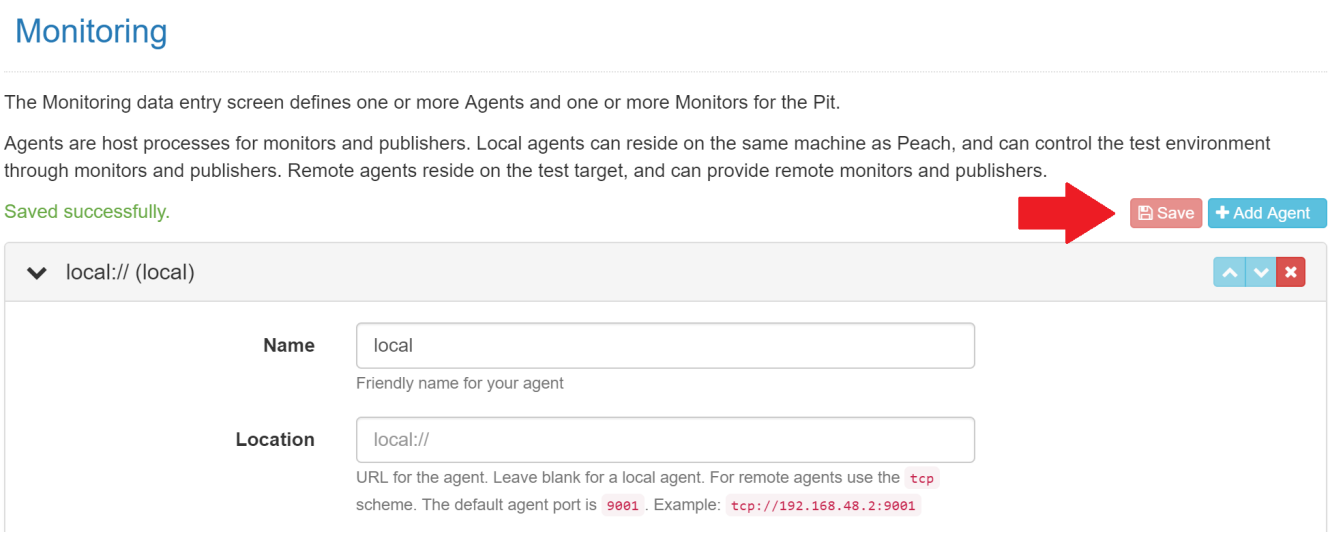
2. Click Add Agent



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



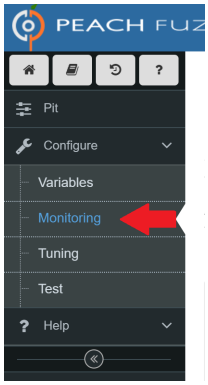
4. Click Save



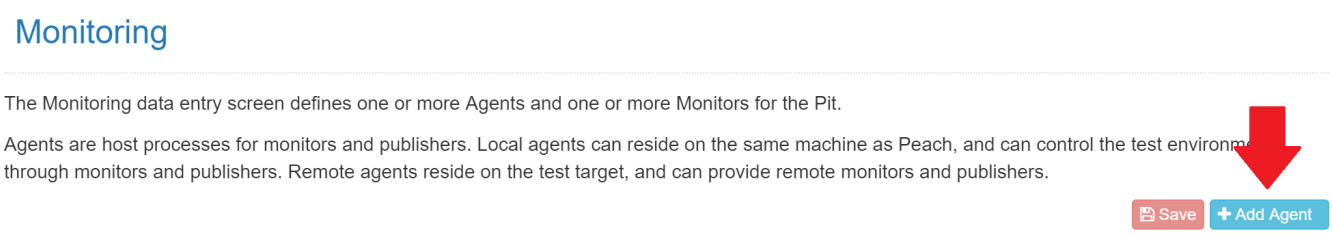
A.1.2. Add a remote agent

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:

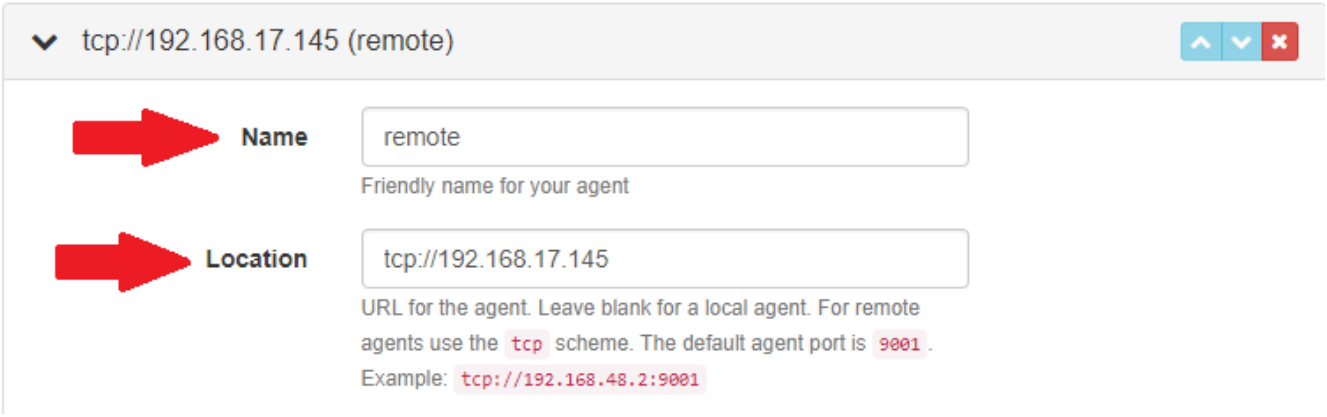
1. Click Monitoring



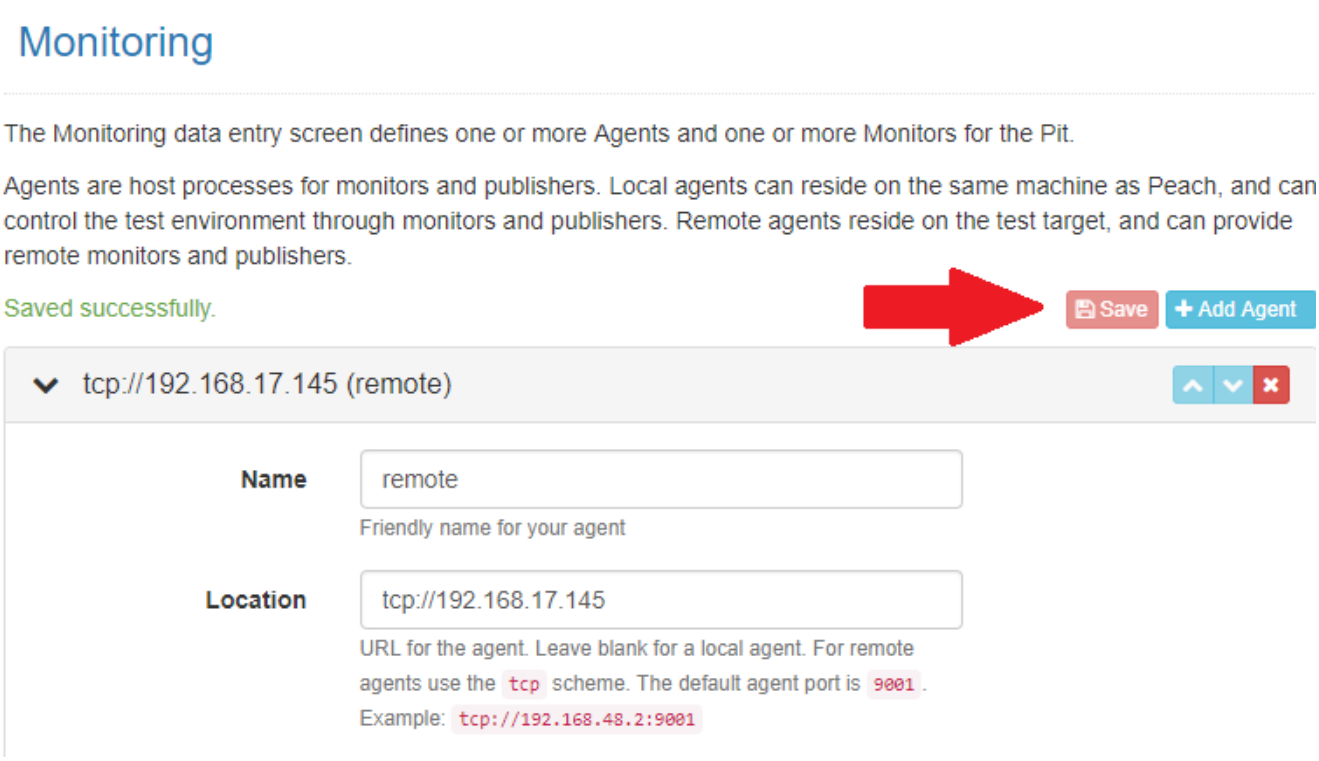
2. Click Add Agent



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.



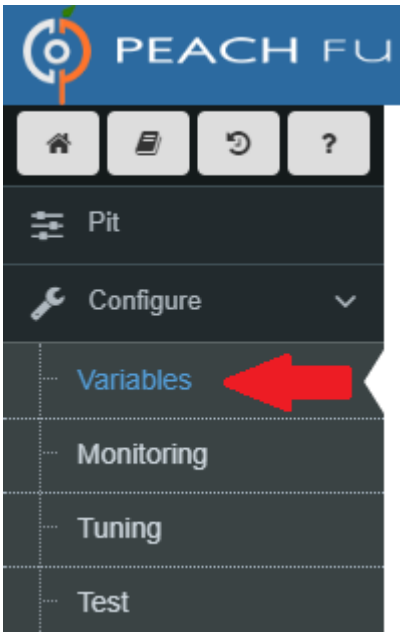
4. Click Save



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**.

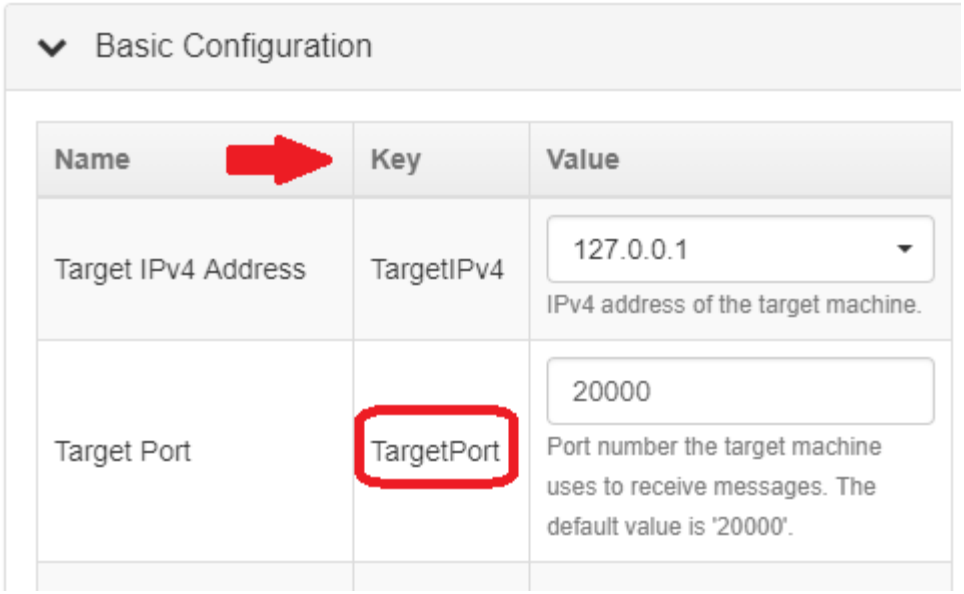


Figure 1. The TargetPort variable is shown here.

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##**.

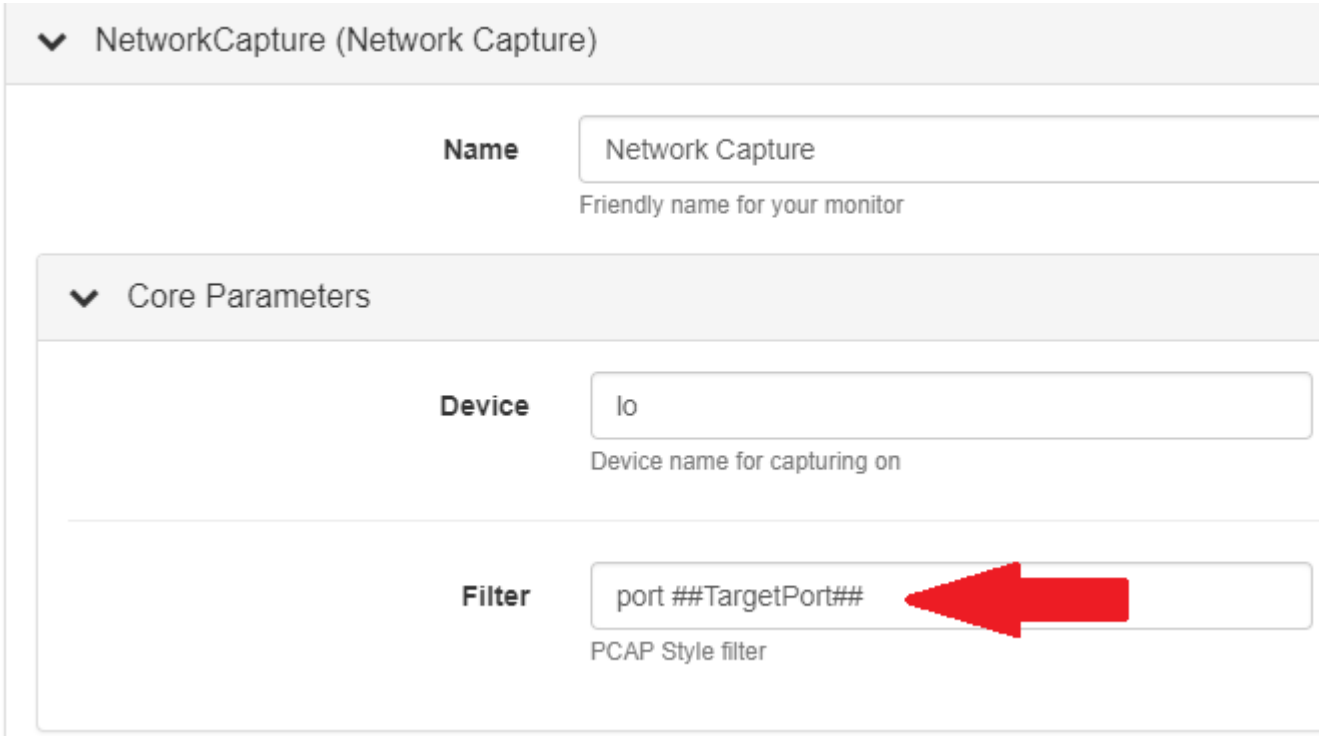


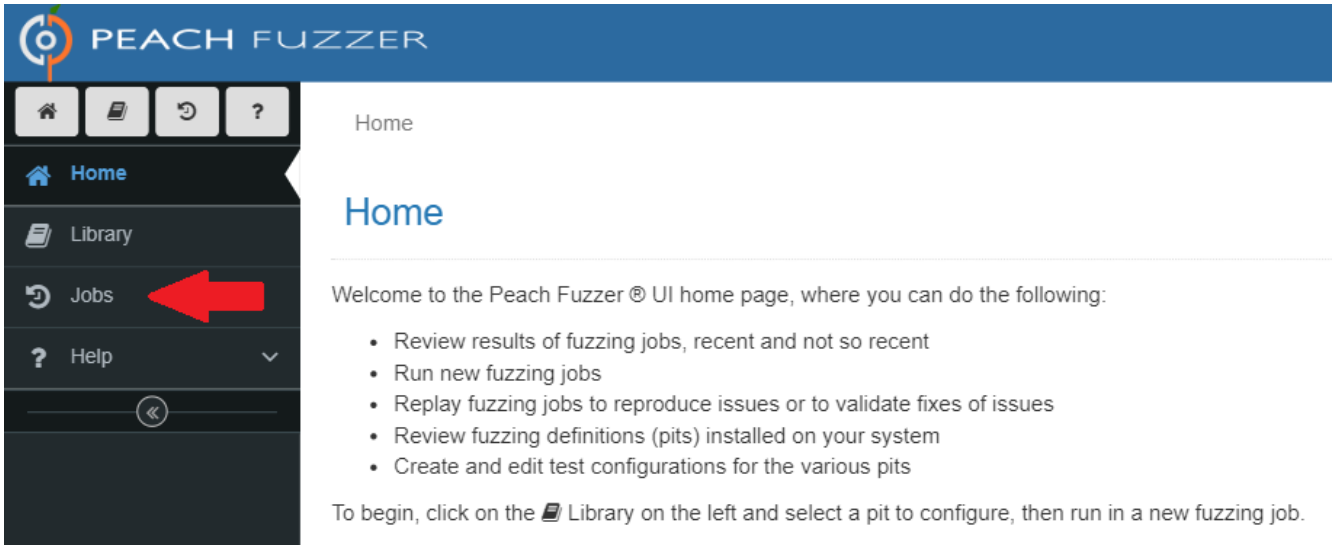
Figure 2. The TargetPort variable is used here to set a Network Capture Monitor to capture all traffic for this configuration. In this example, traffic to and from port 20000 will be captured.

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

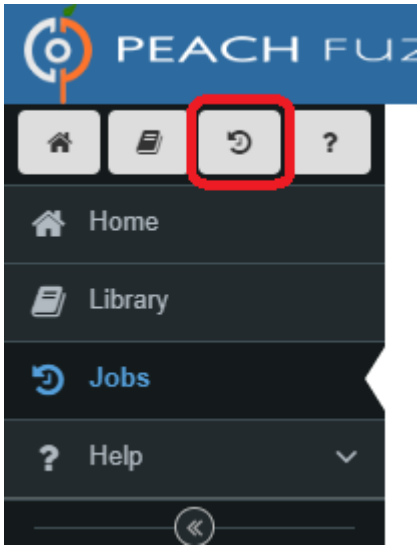
A.3. Examining faults

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.

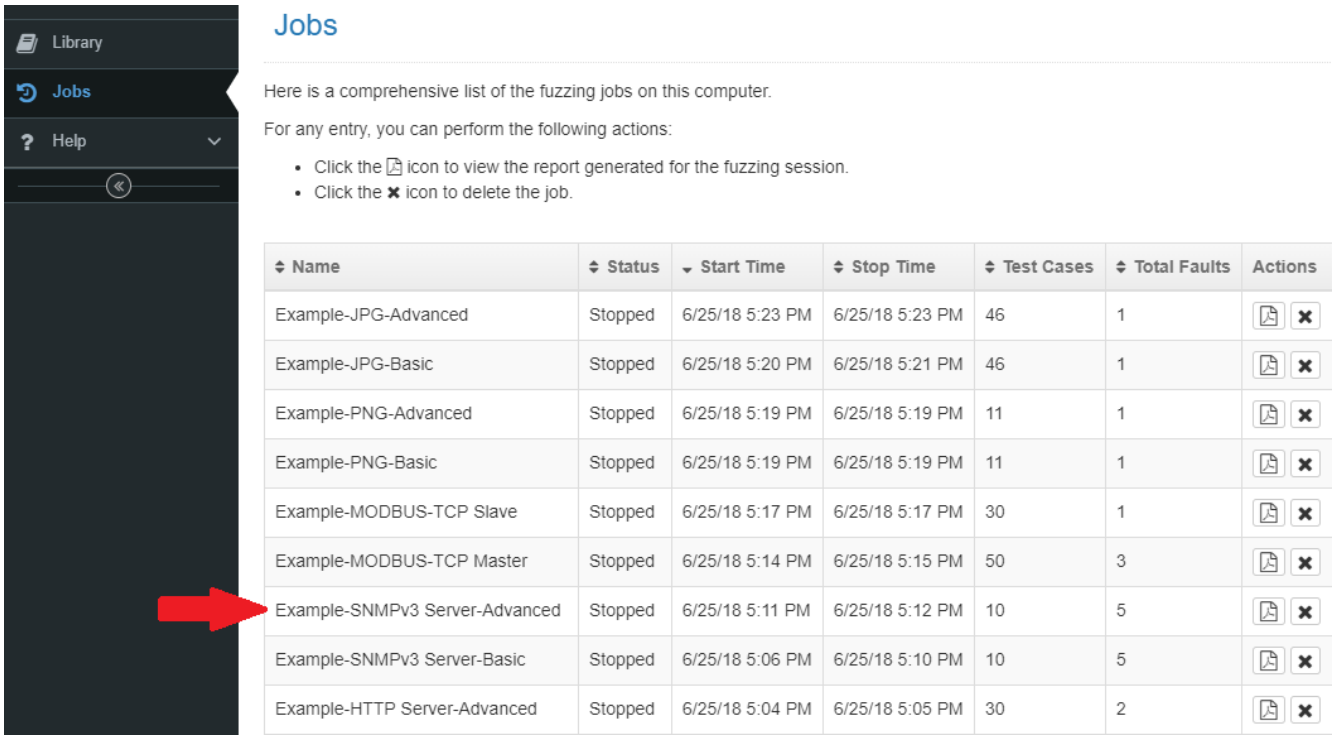
1. First, navigate to the Jobs page by clicking on the Jobs tab



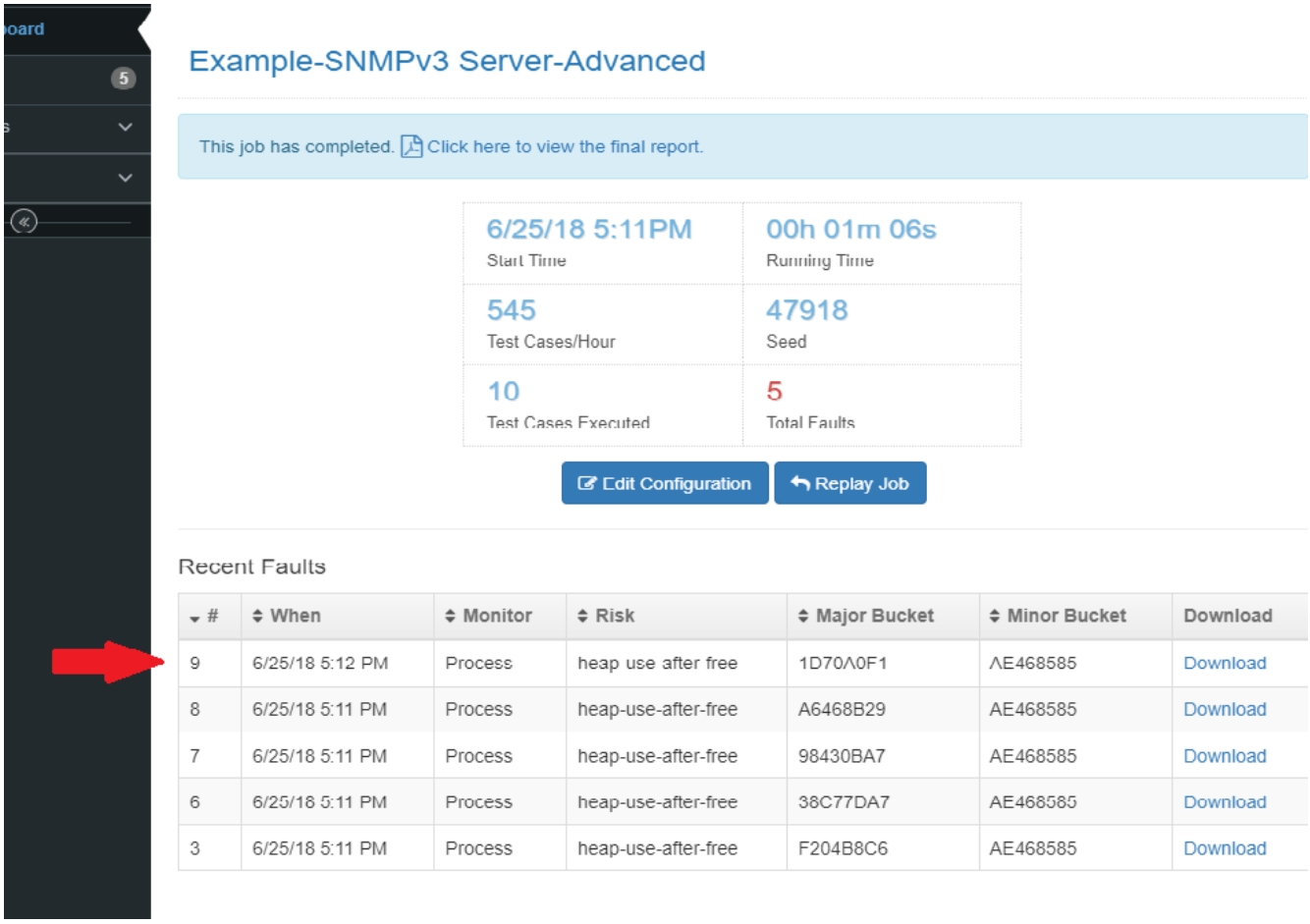
   or by clicking on the Jobs icon



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

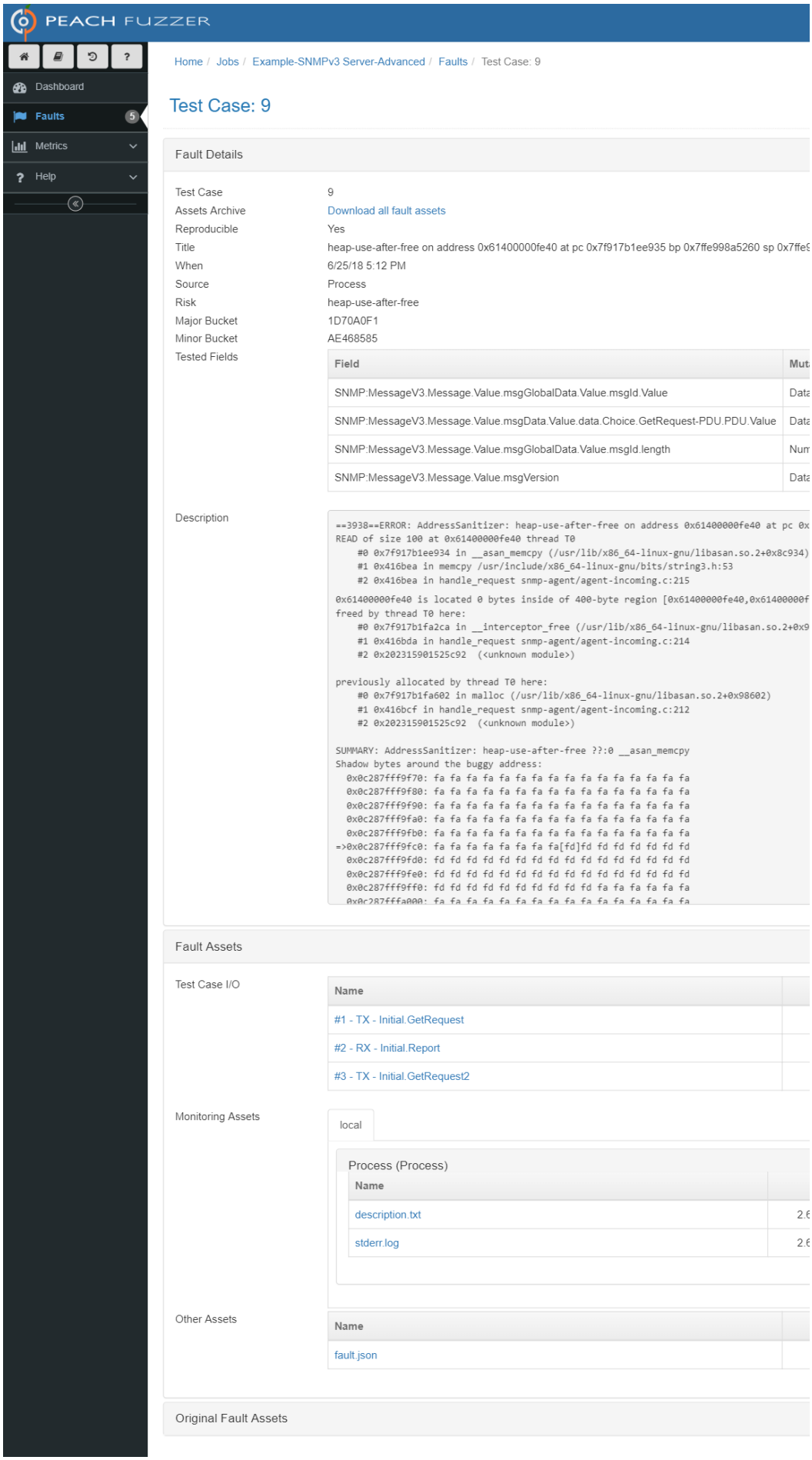


3. The job you selected will now be displayed. Select a fault to view more information.

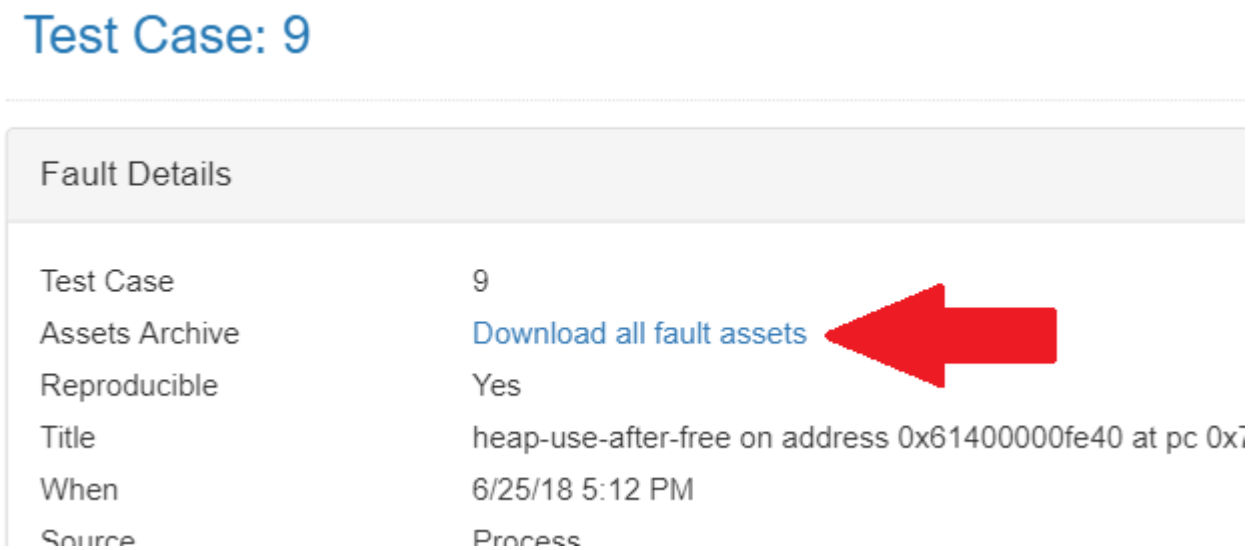


4. The selected fault will contain detailed information about the type of fault, how it was discovered, and the information collected from the various Monitors that were running when the fault

occurred.


PEACH FUZZER
Dashboard
Faults 5
Metrics
Help
Home / Jobs / Example-SNMPv3 Server-Advanced / Faults / Test Case: 9
Test Case: 9
Fault Details
Test Case 9
Assets Archive Download all fault assets
Reproducible Yes
Title heap-use-after-free on address 0x61400000fe40 at pc 0x7f917b1ee935 bp 0x7ffe998a5260 sp 0x7ffe9
When 6/25/18 5:12 PM
Source Process
Risk heap-use-after-free
Major Bucket 1D70A0F1
Minor Bucket AE468585
Tested Fields
Field
SNMP:MessageV3.Message.Value.msgGlobalData.Value.msgId.Value
SNMP:MessageV3.Message.Value.msgData.Value.data.Choice.GetRequest-PDU.PDU.Value
SNMP:MessageV3.Message.Value.msgGlobalData.Value.msgId.length
SNMP:MessageV3.Message.Value.msgVersion
Description
==3938==ERROR: AddressSanitizer: heap-use-after-free on address 0x61400000fe40 at pc 0x
READ of size 100 at 0x61400000fe40 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
0x61400000fe40 is located 0 bytes inside of 400-byte region [0x61400000fe40,0x61400000f
freed by thread T0 here:
#0 0x7f917b1fa2ca in __interceptor_free (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x9
#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
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
Fault Assets
Test Case I/O
Name
#1 - TX - Initial.GetRequest
#2 - RX - Initial.Report
#3 - TX - Initial.GetRequest2
Monitoring Assets
local
Process (Process)
Name
description.txt
stderr.log
Other Assets
Name
fault.json
Original Fault Assets

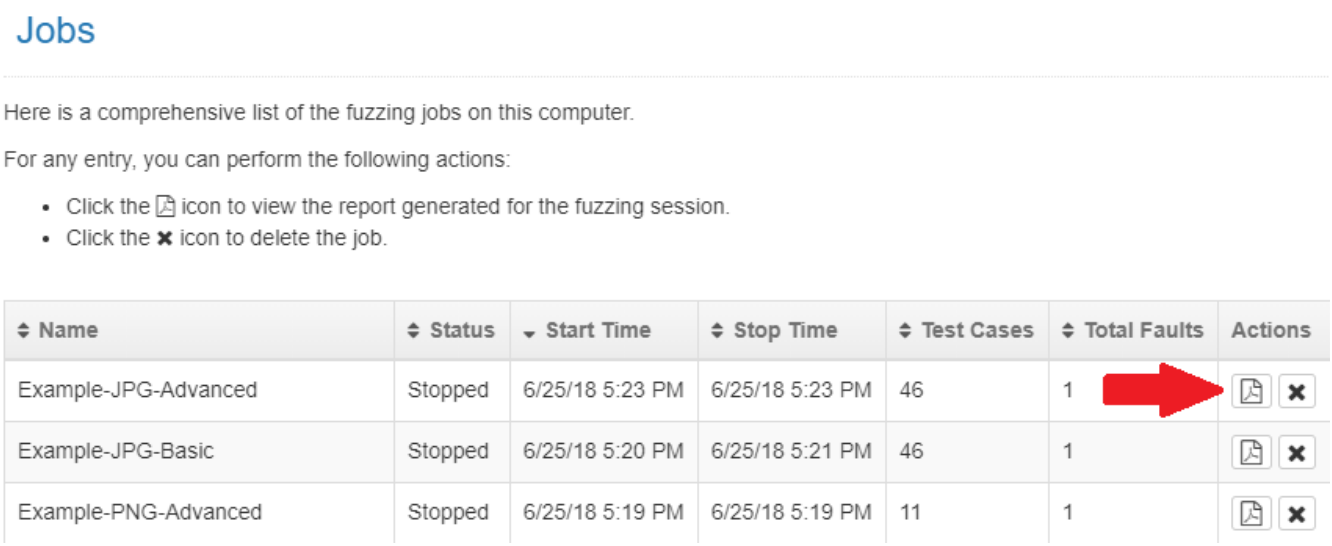
5. You can download all the captured data by clicking **Download all fault assets**.



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



- 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.

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	Download