



PEACH  
FUZZER

## Peach Fuzzer Client-led 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. 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](mailto: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.

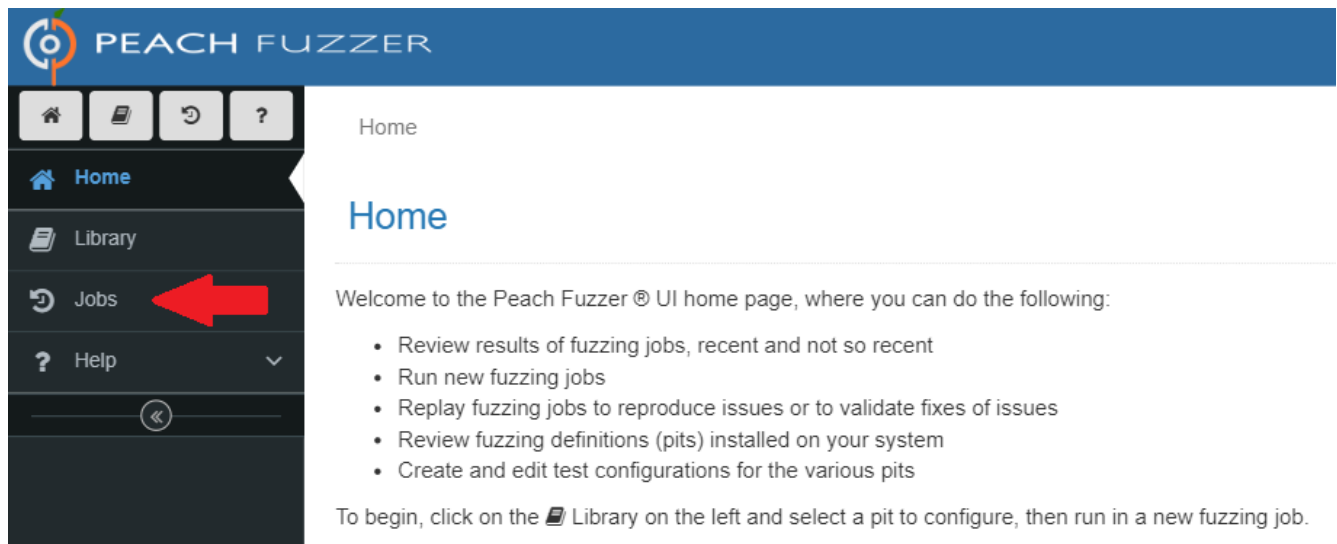


For the purposes of this document we will assume the target VM has an IP address of **192.168.17.145** and this IP address will be used wherever the host's IP address is required. If you see this value used for an IP address in a configuration, replace it with the actual IP address of your VM.

### 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.



















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

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. HTTP-Server

This configuration will test an application using HTTP. 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.

### 4.3.1. Configuring the test

These steps will create the same configuration as is in use in the HTTP\_Server 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 **HTTP\_Server**.

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. Both the Basic and Advanced configuration will use the same variables with the same values. 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 Mongoose 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: set to 4040 as Mongoose will also use to this port.
  - c. HTTP Host Header: set to **peachapisec** as Mongoose has been configured to use this host name.
  - d. Under **Advanced Configuration**, leave all the defaults as they are acceptable for Mongoose.
  - e. 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>192.168.17.145</div> <div>IPv4 address of the target machine.</div>
Target Port	TargetPort	<div>4040</div> <div>Port number the target machine uses to receive messages. The default value is '80'.</div>
HTTP Host Header	HostHeader	<div>peachapisec</div> <div>HTTP Host header provided during an HTTP request. The value should match the DNS hostname used to access resources. The HTTP Host Header is used by web servers to perform virtual hosting.</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 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**.





## 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**

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

### Configuring the monitors for basic

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

- **Gdb Monitor**. This will allow Peach Fuzzer to launch Mongoose from within GDB so that GDB attaches to Mongoose. 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 HTTP 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/mongoose/restful\_server** which is the location of Mongoose'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 **-p 4040**.
4. Click **Save**.

**Name**   
Friendly name for your monitor

**Core Parameters**

**Executable**   
Executable to launch

**Arguments**   
Optional command line arguments

**Gdb Path**   
Path to gdb

**When To Trigger**

**Restart On Each Test**   
Restart process for each iteration

**Restart After Fault**   
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 **eth0** (Mongoose 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**.

NetworkCapture (Network Capture)

Name
Network Capture

Friendly name for your monitor

Core Parameters

Device
eth0

Device name for capturing on

Filter
port ##TargetPort##

PCAP Style filter

## Configuring the monitors for advanced

The advanced configuration is targeting Mongoose 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 Mongoose. Because Mongoose 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 HTTP 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/mongoose/restful_server` which is the location of Mongoose's launcher. This will allow the monitor to launch the application when fuzzing starts.
3. Under **Arguments**, enter `-p 4040`.
4. Do not change any of the other settings for this monitor.
5. Click **Save**.

Process (Process)

Name

Process

Friendly name for your monitor

Core Parameters

Executable

/var/targets/advanced/mongoose/restful\_server

Executable to launch

Arguments

-p 4040

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

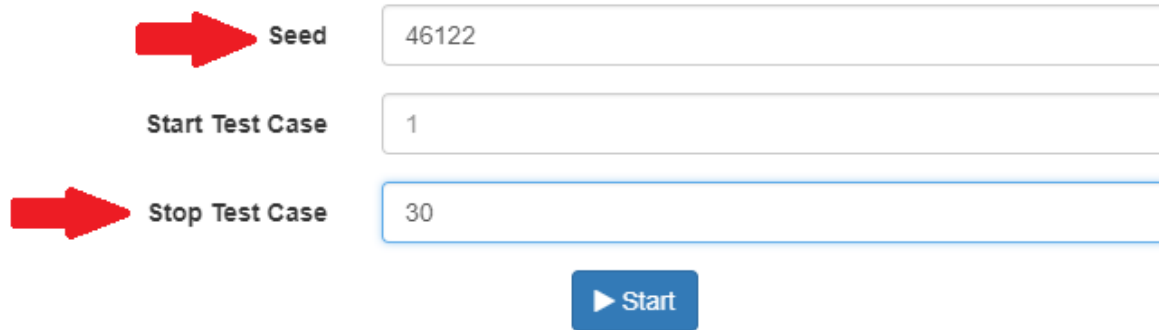
You can now fuzz the application.

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

### 4.3.2. Running the test

To run the pre-configured basic test:

1. Click **Example-HTTP\_Server-Basic**.
2. Enter a seed value of **46122**.
3. Enter a Stop Test Case value of **30**.
4. Click **Start**.



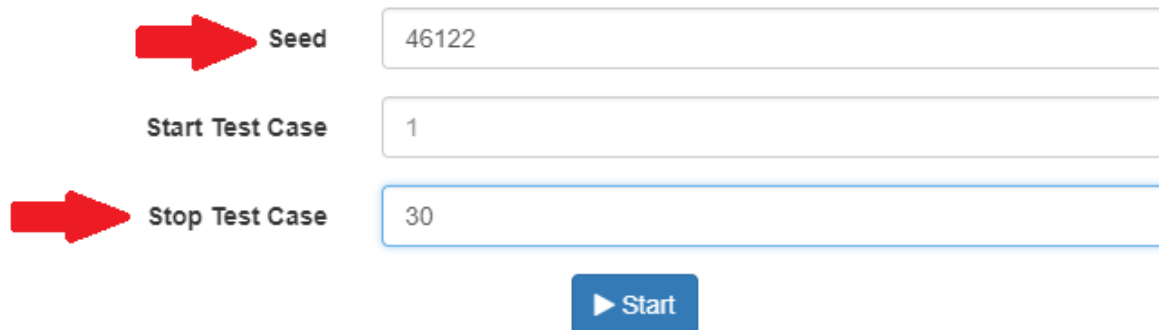
The screenshot shows a configuration form with three input fields and a button. The first field is labeled 'Seed' with a red arrow pointing to it, containing the value '46122'. The second field is labeled 'Start Test Case' and contains the value '1'. The third field is labeled 'Stop Test Case' with a 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	46122
Start Test Case	1
Stop Test Case	30

▶ Start

To run the pre-configured advanced test:

1. Click **Example-HTTP\_Server-Advanced**.
2. Enter a seed value of **46122**.
3. Enter a Stop Test Case value of **30**
4. Click **Start**.



This is a duplicate of the first screenshot, showing the same configuration form with 'Seed' (46122), 'Start Test Case' (1), 'Stop Test Case' (30), and a 'Start' button. Red arrows highlight the 'Seed' and 'Stop Test Case' fields.

Seed	46122
Start Test Case	1
Stop Test Case	30

▶ Start

## 4.4. SNMP-Server

This configuration will test an application using SNMP. 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 SNMPv3\_Server 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 **SNMPv3\_Server**.
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. Both the Basic and Advanced configuration will use the same variables with the same values. 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 osnmpd running on the trial instance:
  - a. Target IPv4 Address: set to **the IP of your Target VM**
  - b. Target Port: set to **161** as osnmpd will also use to this port.
  - c. Source Port: set to **162** as osnmpd is configured to listen for messages from this port.
  - d. Under **Advanced Configuration**, leave all the defaults as they are acceptable for osnmpd.
  - e. 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>192.168.17.145</div> <div>IPv4 address of the target machine.</div>
Target Port	TargetPort	<div>161</div> <div>Port number the target machine uses to receive messages. The default value is '161'.</div>
Source Port	SourcePort	<div>162</div> <div>Port number of the local machine that sends packets. The default value is '162'.</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 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**.





## 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`

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

### Configuring the monitors for basic

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

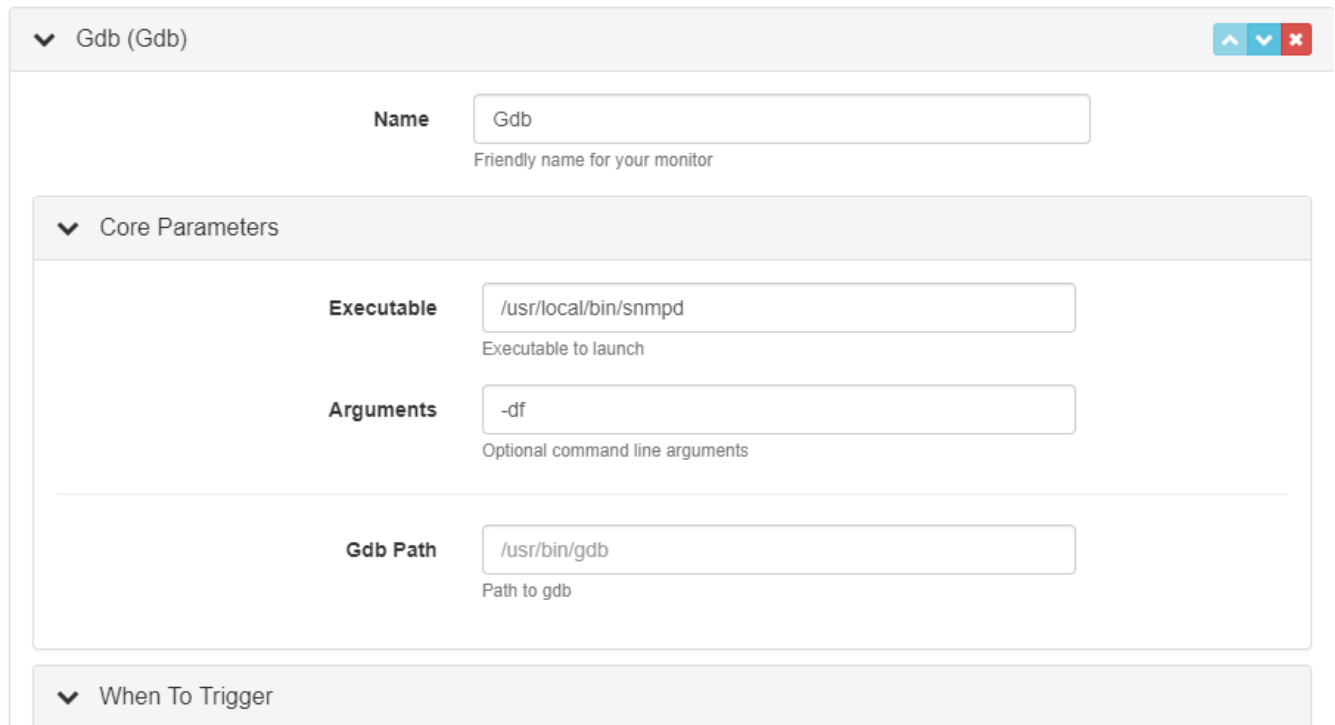
- **Gdb Monitor.** This will allow Peach Fuzzer to launch osnmpd from within GDB so that GDB attaches to osnmpd. Peach Fuzzer will monitor GDB and attempt to analyze any crashes that GDB detects based on receiving signals from the application being tested.
- **RunCommand Monitor** In this case, osnmpd takes a long time to load. It is therefore necessary to add a means to delay starting the test run until the application has fully loaded. To do this, the RunCommand Monitor can be used to run the bash "sleep" command. Peach Fuzzer must wait both at the start of each session and the start of each iteration in case a fault was found and the target had to be restarted. As a result, two monitors will be added with the same sleep command, one set to **OnStart** and the other set to **OnIterationStart**.
- **Network Capture monitor.** Since SNMP 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.
- **Syslog Monitor** The application writes entries to the syslog, which may be useful to capture for



debugging purposes. In some cases, may also be useful to trigger a fault if a particular syslog message is seen.

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 **/usr/local/bin/snmpd** which is the location of osnmpd'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 **-fd**.
4. Click **Save**.



The screenshot shows a configuration window for a Gdb monitor. The window has a title bar with a dropdown arrow, the text "Gdb (Gdb)", and three window control buttons (minimize, maximize, close). The main content area is divided into sections. The first section is labeled "Name" and contains a text input field with the value "Gdb", with a subtitle "Friendly name for your monitor" below it. The second section is titled "Core Parameters" and contains three fields: "Executable" with the value "/usr/local/bin/snmpd" and subtitle "Executable to launch", "Arguments" with the value "-df" and subtitle "Optional command line arguments", and "Gdb Path" with the value "/usr/bin/gdb" and subtitle "Path to gdb". The third section is titled "When To Trigger" and is currently collapsed.

5. Click **Add Monitor**. In the pop-up, scroll down and select **RunCommand**.
6. Under **Command**, enter **/bin/bash**.
7. Under **Arguments**, enter **-c "sleep 1"**. This will call the bash "sleep" command to sleep for 1 second.
8. Under **When To Trigger**, set **When** to **OnStart** so that this will execute when the test session starts.
9. Click **Save**

RunCommand (Run Command)

Name: Run Command  
Friendly name for your monitor

Core Parameters

Command: /bin/bash  
Command line command to run

Arguments: -c "sleep 1"  
Optional command line arguments

Working Directory:   
Working directory to set when running command

When To Trigger

When: OnStart  
Period \_When the command should be ran (OnCall, OnStart, OnEnd, OnIterationStart, OnIterationEnd, OnFault, OnIterationStartAfterFault)

Start On Call:   
Run when signaled by the state machine

10. Click **Add Monitor**. In the pop-up, scroll down and select **RunCommand**.
11. The previous RunCommand monitor will have automatically used the name "Run Command" when it was created unless it was changed. You will need to give this RunCommand Monitor a different name. Here, **Run Command On Iteration Start** is used.
12. Under **Command**, enter **/bin/bash**.
13. Under **Arguments**, enter **-c "sleep 1"**. This will call the bash "sleep" command to sleep for 1 second.
14. Under **When To Trigger**, set **When** to **OnIterationStart** so that this executes when each iteration starts. This is important because if a fault is found, Peach Fuzzer will re-start the target application and therefore needs to delay running the next iteration to give the target application enough time to load.
15. Click **Save**

RunCommand (Run Command On Iteration Start)

Name

Run Command On Iteration Start

Friendly name for your monitor

Command

/bin/bash

Command line command to run

Arguments

-c "sleep 1"

Optional command line arguments

Working Directory

Working directory to set when running command

When To Trigger

When

OnIterationStart

Period \_ When the command should be ran (OnCall, OnStart, OnEnd, OnIterationStart, OnIterationEnd, OnFault, OnIterationStartAfterFault)

Start On Call

Run when signaled by the state machine

16. Click **Add Monitor**. In the pop-up, scroll down and select **Network Capture** under the **Data Collection** section. Click **Ok**.
17. Under **Device**, enter **eth0** (osnmpd is listening on the target VM's eth0 interface)
18. Under **Filter**, enter **port ##TargetPort## or port ##SourcePort##** to capture all traffic going to or from the port you specified when you configured the variables in the previous section.
19. Click **Save**.

18

▼ NetworkCapture (Network Capture)

**Name**   
Friendly name for your monitor

▼ Core Parameters

**Device**   
Device name for capturing on

**Filter**   
PCAP Style filter

20. Click **Add Monitor**. In the pop-up, scroll down and select **Syslog**. Click **Ok**.
21. Leave the default values for **Interface** and **Port**.
22. Click **Save**

▼ Syslog (Syslog)

**Name**   
Friendly name for your monitor

▼ Core Parameters

**Interface**   
Interface to listen on

**Port**   
Port number to listen on

**Fault Regex**   
Fault when regular expression matches

▼ When To Trigger

## Configuring the monitors for advanced

The advanced configuration is targeting osnmpd 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 osnmpd. Because osnmpd 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 SNMP 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.
- **Syslog Monitor** The application writes entries to the syslog, which may be useful to capture for debugging purposes. In some cases, may also be useful to trigger a fault if a particular syslog message is seen.

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/osnmpd/bin/snmpd` which is the location of osnmpd's launcher. This will allow the monitor to launch the application when fuzzing starts.
3. Under **Arguments**, enter `-fd`.
4. Do not change any of the other settings for this monitor.
5. Click **Save**.

The screenshot shows the 'Add Monitor' dialog box with the 'Process' monitor selected. The dialog has a title bar with a dropdown arrow, the text 'Process (Process)', and three window control buttons (minimize, maximize, close). The main content area is divided into sections. The first section is 'Name', with a text input field containing 'Process' and a subtitle 'Friendly name for your monitor'. The second section is 'Core Parameters', which contains two fields: 'Executable' with the value '/var/targets/advanced/osnmpd/bin/snmpd' and subtitle 'Executable to launch', and 'Arguments' with the value '-fd' and subtitle 'Optional command line arguments'. The third section is 'When To Trigger', which is currently collapsed.

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` (osnmpd is listening on the target VM's eth0 interface)
8. Under **Filter**, enter `port ##TargetPort## or port ##SourcePort##` to capture all traffic going to or from the port you specified when you configured the variables in the previous section.

9. Click **Save**.

The screenshot shows a configuration window titled "NetworkCapture (Network Capture)". It has a "Name" field with the value "Network Capture" and a subtitle "Friendly name for your monitor". Below this is a section titled "Core Parameters". Inside this section, there is a "Device" field with the value "eth0" and a subtitle "Device name for capturing on". There is also a "Filter" field with the value "port ##TargetPort## or port ##SourcePort##" and a subtitle "PCAP Style filter".

10. Click **Add Monitor**. In the pop-up, scroll down and select **Syslog**. Click **Ok**.

11. Leave the default values for **Interface** and **Port**.

12. Click **Save**

The screenshot shows a configuration window titled "Syslog (Syslog)". It has a "Name" field with the value "Syslog" and a subtitle "Friendly name for your monitor". Below this is a section titled "Core Parameters". Inside this section, there is an "Interface" field with the value "0.0.0.0" and a subtitle "Interface to listen on". There is also a "Port" field with the value "514" and a subtitle "Port number to listen on". There is a "Fault Regex" field with an empty value and a subtitle "Fault when regular expression matches". Below the "Core Parameters" section is a section titled "When To Trigger".

## 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**.

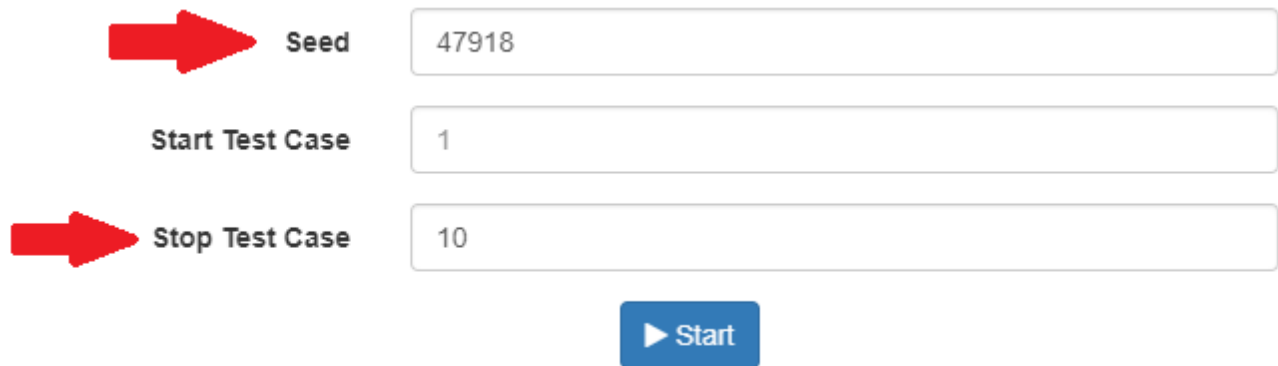
You can now fuzz the application.

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-SNMPv3\_Server-Basic**.
2. Enter a seed value of **47918**.
3. Enter a Stop Test Case value of **10**.
4. Click **Start**.



The screenshot shows a configuration form for a test. It has three input fields and a button. The first field is labeled 'Seed' with a red arrow pointing to it, containing the value '47918'. 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 '10'. Below these fields is a blue button with a white play icon and the text 'Start'.

Seed	47918
Start Test Case	1
Stop Test Case	10

**Start**

To run the pre-configured advanced test:

1. Click **Example-SNMPv3\_Server-Advanced**.
2. Enter a seed value of **47918**.
3. Enter a Stop Test Case value of **10**
4. Click **Start**.



**Seed**

47918

**Start Test Case**

1



**Stop Test Case**

10

▶ **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.

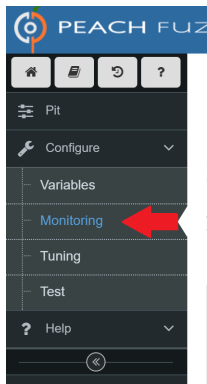


It is typically not necessary to configure multiple agents for the same machine. A single agent is capable of running multiple different monitors.

### A.1.1. Add a local agent

To add a local agent, follow these steps:

1. Click Monitoring



2. Click Add Agent

#### 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)

 **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)

**Name**

Friendly name for your agent

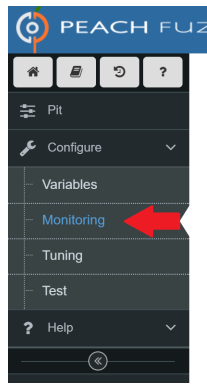
**Location**

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

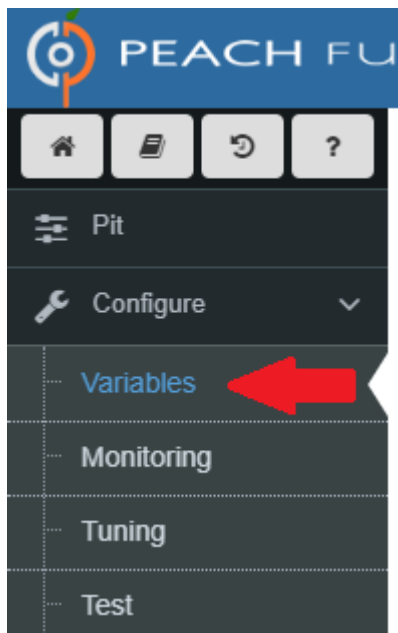
tcp://192.168.17.145

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

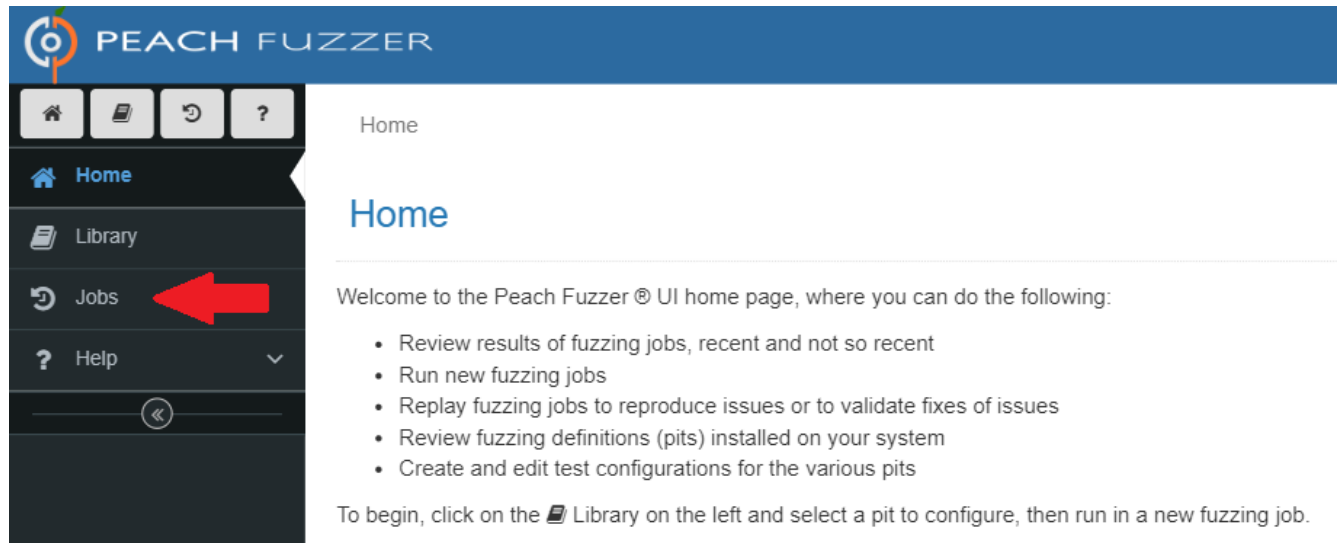
**Figure 2. The TargetPort variable is used here to set a Network Capture Monitor to capture all traffic for this configuration. In this example, traffic to and from port 20000 will be captured.**

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

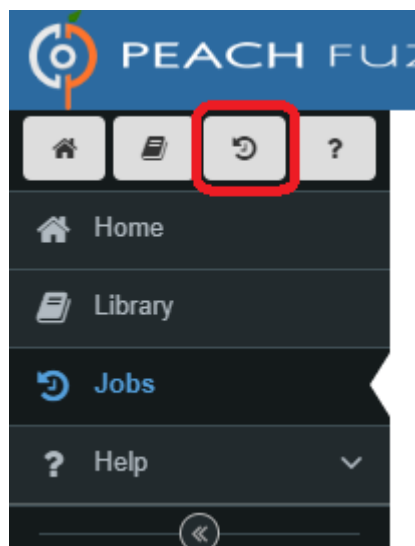
## A.3. Examining faults

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

Jobs

Example-SNMPv3 Server-Advanced

Faults

Test Case: 9

Dashboard

Faults5

Metrics

Help

Test Case: 9

Fault Details

Test Case9

Assets ArchiveDownload all fault assets

ReproducibleYes

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

When6/25/18 5:12 PM

SourceProcess

Riskheap-use-after-free

Major Bucket1D70A0F1

Minor BucketAE468585

Tested Fields

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	Num
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 [0x6140000fe40,0x6140000f
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 fd fd fd fd fd fd
 0x0c287fff9ff8: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
 0x0c287fff9ff0: fa fa fa fa fa fa fa fa fa fa 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	2.6
stderr.log	2.6

Other Assets

Name	
fault.json	

Original Fault Assets

32

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>