FORTRA

beSTORM

EDSA Companion User Guide

202303230715

Table of Contents

Overview

On top of adding 6 new modules corresponding to the ISA EDSA tests, beSTORM also brings significant enhancements in usability and functionality in the GUI, engine, functionality, extensibility, interfacing, and monitoring aspects.

This EDSA Companion User Guide will shed light on these key points, and as such, it does not come to replace the official beSTORM User Guide, but rather to update and extend it where applicable.

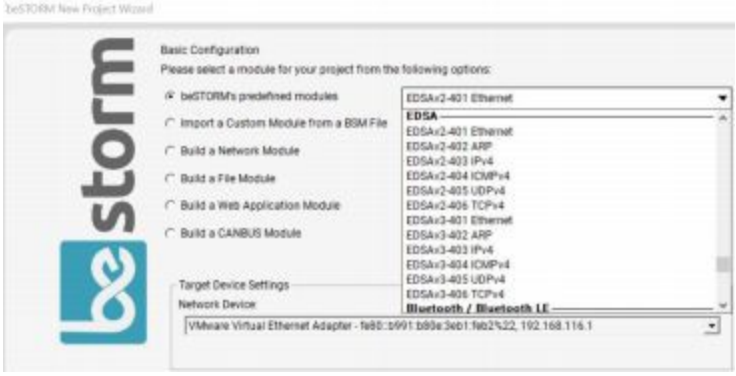
New EDSA modules

All ISASecure EDSA modules correspond by both index and name, and comply with the protocol reference standards cited in the CRT specifications listed in Section 2 of ISASecure EDSA-201-ISASecureCRTTool-Recognition Process (v1_21), specifically:

- ISASecure EDSA-310 ISA Security Compliance Institute Embedded Device Security Assurance - Common requirements for communication robustness testing of IP-based protocol implementations.
- [ISASecure EDSA-401] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of two common “Ethernet” protocols.
- [ISASecure EDSA-402] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of the IETF ARP protocol over IPv4.
- [ISASecure EDSA-403] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of the IETF IPv4 network protocol.
- [ISASecure EDSA-404] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of the IETF ICMPv4 network protocol.
- [ISASecure EDSA-405] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of the IETF UDP transport protocol over IPv4 or IPv6.
- [ISASecure EDSA-406] ISA Security Compliance Institute Embedded Device Security Assurance – Testing the robustness of implementations of the IETF TCP transport protocol over IPv4 or IPv6.

To access these another EDSA modules in beSTORM, do the following:

1. Open **beSTORM Client**.
2. Select **New Project Wizard**.
3. On the **Basic Configuration** page, select an EDSA module from the **beSTORM's predefined modules** list.



Each module has its own environment variables configurable through the Advanced Configuration screen. However, they also share some important common variables and modes of control.

Looking through the object-oriented class, each (new) project is an (object) instantiation of a certain module (class), predefined or imported (built-in or custom), where its environment-variables (members) take default values and can be overloaded.

In some cases, the values for some variables are fixed by the definition for a particular test. In those cases, the values set by the user for those variables, are not used by that test.

Payload Generation beSTORM is creating “from scratch” payloads that correspond and encapsulate all the necessary data applicable to the different tests and protocols. For example, creating a TCP Packet involves encapsulating the TCP data and its header within an IP packet, with the correctly calculated checksums including any pseudo-headers, all encapsulated within an Ethernet frame. It then “drops” these payloads onto the network interface selected for a specific project.

For example, all packets eventually “ride” within Ethernet frames, so all modules will include all the operational environment variables derived from the Ethernet module. Both TCP and UDP are using the same base socket information, etc.

Throughput Control

These streams of varied length payloads encoded along different protocols, which during testing the tool is generating, are known as “attack vectors”.

Since some of the protocols/tests require more than one transmission per session (e.g. TCP SYN, TCP-DATA, TCP-RST), an “attack vector” is not necessarily only a single “packet” of the tested protocol, but the total transmission of packets that takes place within a single traversal on the module tree.

The beSTORM tool is traversing the nodes of the tree-space derived from the optimized module description, and of which evaluation results in a set of actions/transmissions that represent a single attack. Therefore, the tool’s native unit of network throughput is given as attack-vectors per second (avps), and it includes a settable value called “SRT” (Saturation Rate Threshold) which sets the tool’s throughput upper-limit in avps units, with a maximum value of 250,000 avps.

Under these circumstances, for example, given that the shortest IP message is 20 bytes, and 84 bytes when encapsulated in a valid Ethernet, max SRT would be at worst equivalent to 250k IP/sec and 168Mbps, respectively.

Other than “capping” the tool’s avps throughput by using the SRT setting, the tool has two additional settings which allow network capping:

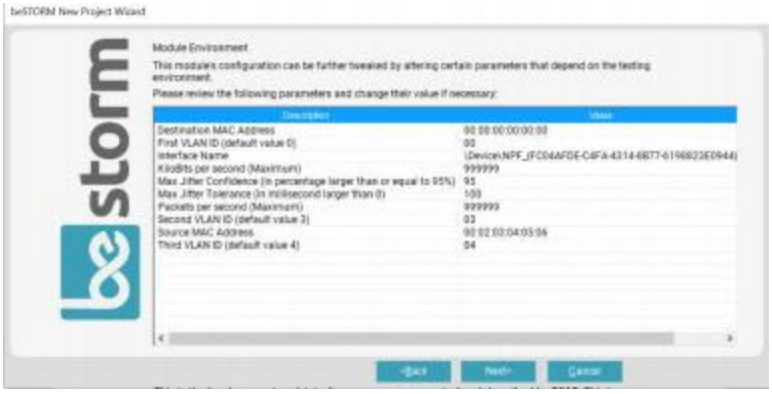
- **Packets Per Second** – while “attack vectors” are complete transmissions happening within a tree node, “packets” are the countable transmissions that happen within. Thus, regardless of protocol and test, the user can set a capping value based on the rate of packet transmission. For example, a cycle on 1 TCP attack-vector may include 3 packets.
- **Kilobits Per Second** – in accordance with SI, this will cap the throughput to the value specified in thousands of bits per second, where it is practically adjusted to the nearest multiple of 8 (1 Bytes).

These two settings are capping the network concurrently, so during testing, actual capping will occur at the lowest rate resulting from the values specified in them.

The only exception to that would be in the “Saturation” test, corresponding to the Phase-2 of the “Maintenance of Service under high load” of each module, where the “avps” setting is automatically and temporarily set to MAX, and any settings made to the “Packet Per Second” or “Kilobits Per Second” is temporarily disregarded – effectively disabling any capping on the throughput for that test/phase.

Modules

EDSA-401 Ethernet



Two variants of Ethernet (II and 802.3) are used throughout the tests as described.

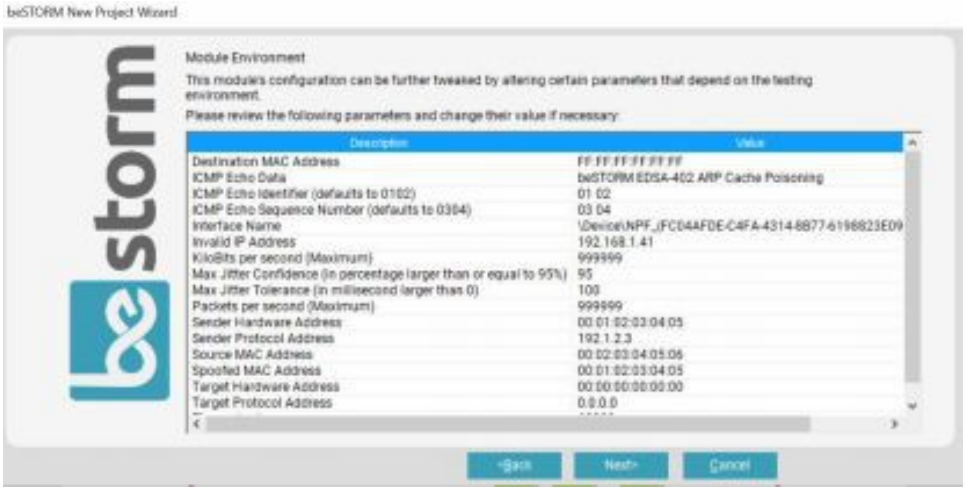
- **Destination MAC Address** - This must be a broadcast or the real MAC of the DUT, otherwise it won't even pick the sent attack vectors, and so no vulnerabilities would be found even if they exist. This environment variable is consistent throughout all the EDSA modules.
- **First/Second/Third VLAN ID** - These are the default or fixed values used for the tests that implement usage of the 802.1Q tag 10-bit VLAN ID, where a value of 0 means broadcast (equivalent to no-tag), 4095 is RESERVED, and 1 might be inherently used within network bridges.
- **Interface Name** - This is the hardware network interface name as enumerated and described by PCAP. This is where all the raw bytes eventually go and are expected to come from. This environment variable is consistent throughout all the EDSA modules.
- **Kilobits Per Second** - Capping using Kbps. Effective in all tests except "Saturation". This environment variable is consistent throughout all the EDSA modules.
- **Packets Per Second** - Capping using number of transmitted packets (of current module protocol). Effective in all tests except "Saturation". This environment variable is consistent throughout all the EDSA modules.
- **Source MAC Address** - This is supposed to be the MAC of the TD, which can (mostly) be spoofed if required. This environment variable is consistent throughout all the EDSA modules.

You must select the following required parameters for this module:

- **Interface**
- **Destination MAC address**
- **Kilobits Per Second / Packets Per Second** - At least one of these based upon the known limited rate of the destination device (see EDSA-310).

Other values can remain at their defaults.

EDSA-402 ARP



ICMP Echo Data, Identifier, Sequence Number, Spoofed (Sender) MAC Address and Invalid IP (Sender Protocol) Address, as well as Timeout Value These are specifically used for the ARP.T01 DUT Cache Poisoning test, in which we use ICMP messages to provoke the DUT into resolving the Invalid IP Address we start with. We are waiting for applicable responses during the specified timeout given in milliseconds.

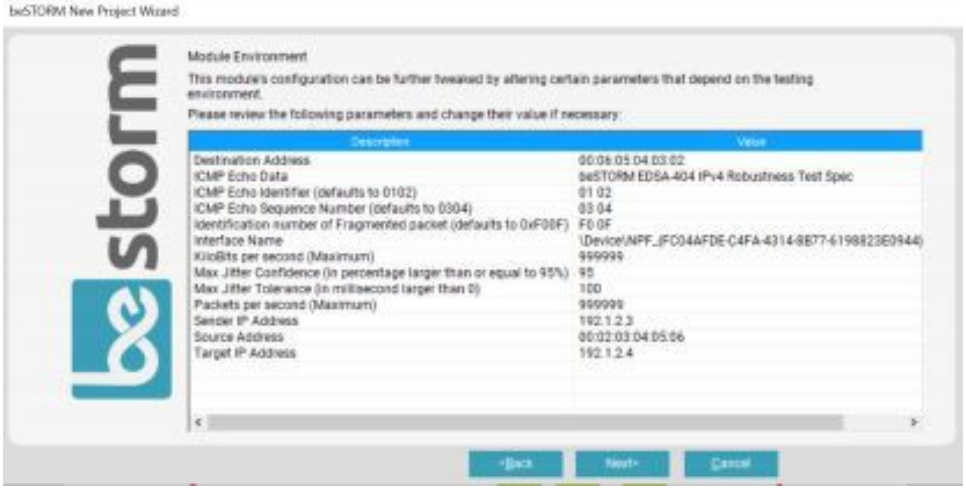
- **Sender Hardware Address** - In the context of an ARP request, this is supposedly equivalent to the Ethernet Sender's MAC (and therefore also much ignored by receiving parties).
- **Sender Protocol Address** - In the context of an ARP request, this is the IP address of the sender of the request.
- **Target Protocol Address** - In the context of an ARP request, this is the IP address to be resolved into a MAC.
- **Target Hardware Address** - In the context of an ARP request, this makes no predefined sense and is usually ignored.

You must select the following required parameters for this module:

- **Interface**
- **Kilobits Per Second / Packets Per Second** - at least one of these based upon the known limited rate of the destination device (see EDSA-310)
- **Sender protocol address**
- **Target protocol address**

Other values can remain at their defaults.

EDSA-403 IPv4



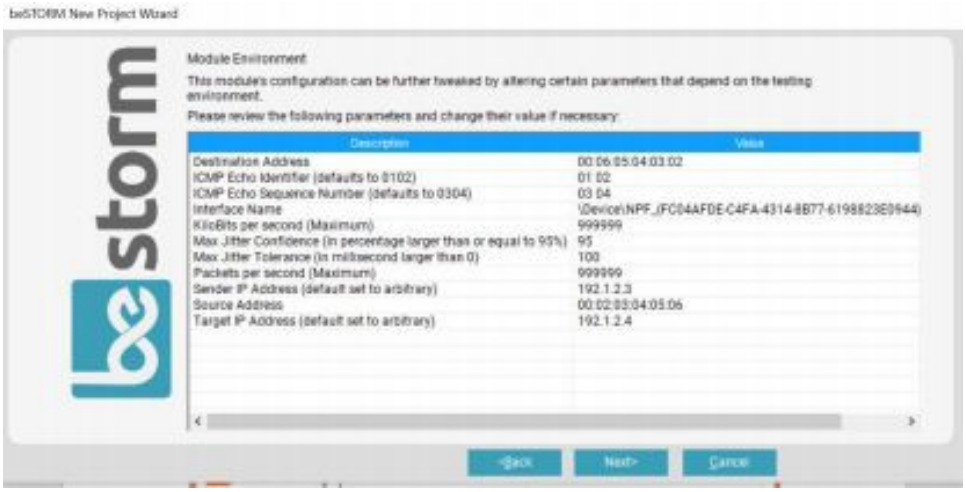
- **ICMP Echo Data, Identifier, Sequence Number** - Throughout the IPv4 tests, ICMP is used as the payload. These are the default ICMP values used where applicable. Identification number of Fragmented packet Used in the reassembly tests.
- **Identification number of Unreassembled packets** - Used in the high load reassembly test.
- **Sender and Target IP address** - To be specified by default in the IP headers. Can be set and/or spoofed as applicable. These environment variables are consistent throughout all IPv4 based modules, i.e. ICMPv4, UDPv4 and TCPv4.

You must select the following required parameters for this module:

- **Interface**
- **Kilobits Per Second / Packets Per Second** - at least one of these based upon the known limited rate of the destination device (see EDSA-310)
- **Sender IP address**
- **Target IP address**

Other values can remain at their defaults.

EDSA-404 ICMPv4



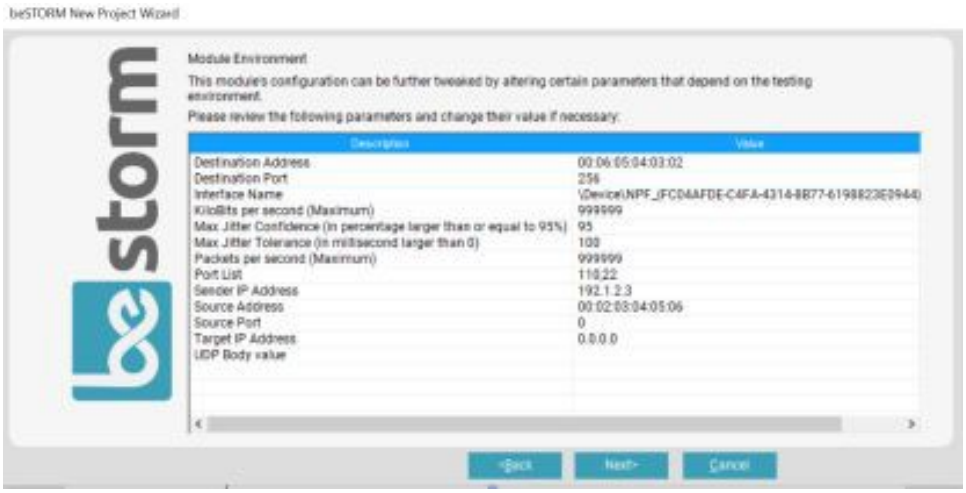
ICMP Echo Data, Identifier, Sequence Number - The default value of the ICMP fields to be fuzzed.

You must select the following required parameters for this module:

- **Interface**
- **Kilobits Per Second / Packets Per Second** - at least one of these based upon the known limited rate of the destination device (see EDSA-310)
- **Sender IP address**
- **Target IP address**

Other values can remain at their defaults.

EDSA-405 UDPv4



- **Destination Port** - This is the default DUT's port number used to communicate with. For tests that expect the DUT to reply, an actual, adequate service must be established and bound to that port locally on the DUT. These environment variables are also consistent with the TCPv4 socket based module.
- **Source Port** - This is the default TD's port number used to communicate from. 0 traditionally means "auto assign" by the system. These environment variables are also consistent with the TCPv4 socket based module.
- **UDP Body Value** - This is the default data payload sent with the datagrams.
- **Port List** - This is the list of ports expected to be open (and servicing) on the DUT, per the UDP conveyed application test.

You must select the following required parameters for this module:

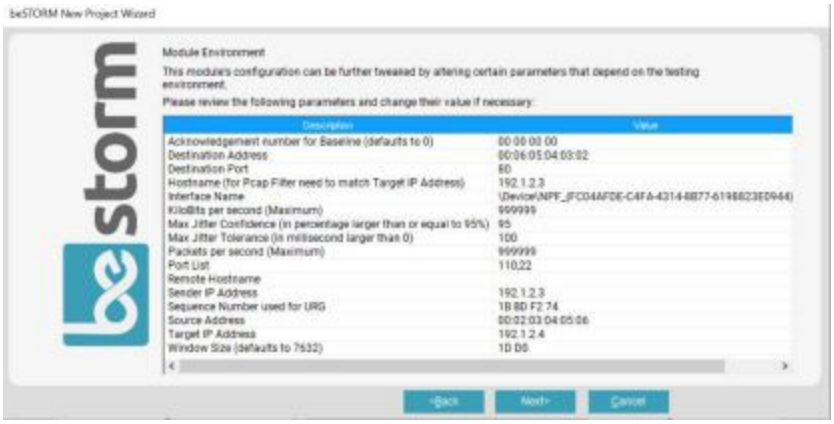
- **Interface**
- **Kilobits Per Second / Packets Per Second** - at least one of these based upon the known limited rate of the destination device (see EDSA-310)
- **Sender IP address**
- **Target IP address**
- **Destination port**
- **Port list**

Other values can remain at their defaults.

EDSA-406 TCPv4

For this module, numbered test T01 is handled using an external tool (a python script), which is distributed as a package named SPA (SYN Probability Analysis) and includes its own documentation.

Please note that when using the script you should set the target host argument -t as the device to be tested and accept the SPA default parameters for sample size -c and test duration -cto. Other SPA parameters that control data collection can be set as per your preference. Please refer to that documentation for performing and evaluating that test.



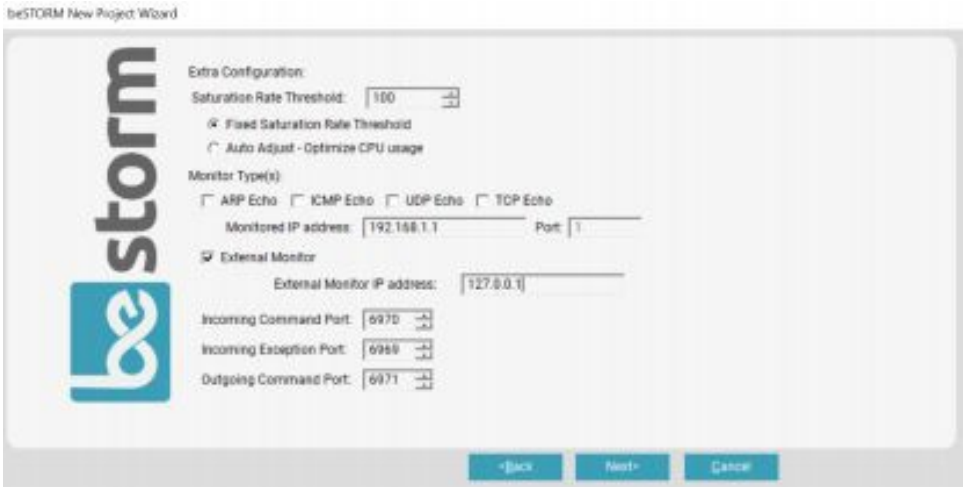
- **Hostname (for PCAP…)** - This is the DUT's IP address that the TD is using to filter incoming packets with (SYN+ACK, etc…)
- **Sequence Number used for URG** - Used throughout the tests as prescribed and applicable.
- **Window Size** - Used throughout the tests as prescribed and applicable. Fuzzed in the Spoofed Flags test.
- **Port List** - This is the list of ports expected to be open (and servicing) on the DUT, per the TCP conveyed application test.
- **Remote Hostname** - This is the remote hostname value used within the HTTP-GET and SMTP HELO parts of the conveyed application test.

You must select the following required parameters for this module:

- **Interface**
- **Kilobits Per Second / Packets Per Second** - at least one of these based upon the known limited rate of the destination device (see EDSA-310)
- **Sender IP address**
- **Target IP address**
- **Destination port**
- **Port list**

Other values can remain at their defaults.

Internal Monitoring



Four internal network monitors can be inclusively switched to periodically “sniff” the DUT’s network liveliness. Any failure will trigger an exception.

- **ARP Echo** – resolve the Monitored IP address into a MAC.
- **ICMP Echo** – send the Monitored IP address an echo request and expect to receive an echo reply from it.
- **UDP Echo** – send a UDP datagram to the Monitored IP address and Port and verify no related ICMP errors are received. Note that using this obviously requires turning off all firewalls, as ICMP errors are not propagated.
- **TCP Echo** – try establishing a connection with the Monitored IP address and Port.

Set the ARP, ICMP, and UDP monitors when creating an EDSA project for the ARP, ICMP, and UDP EDSA modules, respectively. Set the TCP monitor for the Ethernet, IPv4 and TCP EDSA modules.

External Monitoring beSTORM’s monitoring concept is very simple. To evoke an exception, all that must be done is sending a UDP message to the specified Incoming Exception Port (6969), which is always open.

The text that consists of the body of that message is associated with the currently tested attack-vector, and both are encapsulated as a reproducible and reportable exception, with the potential of being a vulnerability.

DUT Waveform Monitoring

With EDSA, there's a python script that monitors a Labjack which interfaces the waveforms coming from the DUT's control-side. In case that python script recognizes an excessive jitter in the waveform's timing, it would evoke an exception by sending a message with all the relevant information to the same machine's (usually localhost) port 6969.

This script, which must be manually run in the parallel by the user, is a part of a package called EDW that comes with its own documentation.

The user sets the following EDW parameters in accordance with the specified jitter (see EDSA-310) and analog signal characteristics of the target device: MAXJITTER, MINCONFIDENCE, COMPOSITEJITTER, ANALOGMIN, ANALOGMAX. All other EDW parameters use default values for EDSA testing.

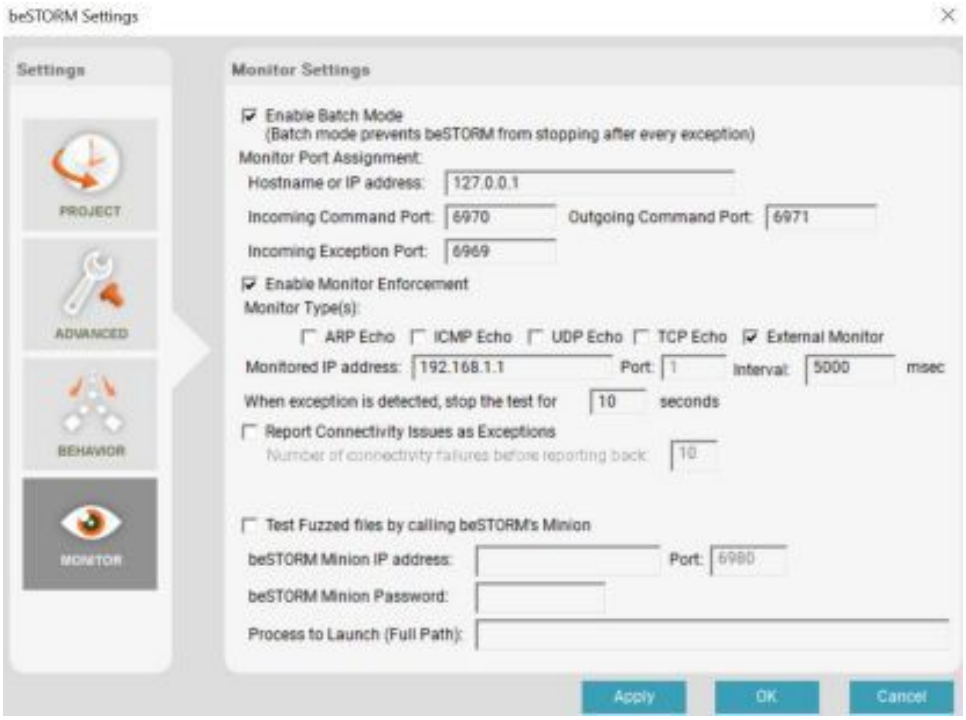
Please refer to that documentation for operating the waveform monitor.

Monitoring for the Monitor's Heartbeats

To make sure that the monitor itself is alive, it is possible to listen to incoming "heartbeats" in the form of 4-byte NOOP messages received at the Incoming Command Port (6970). Checking the "beSTORM Monitor" checkbox does two things:

1. It opens this port to any incoming command, including the NOOP command.
2. It enforces a mandatory incoming heartbeat "watchdog", with a reset cycle that expects a 4-byte NOOP message to be received within 10 seconds at the Incoming Command Port (6970).

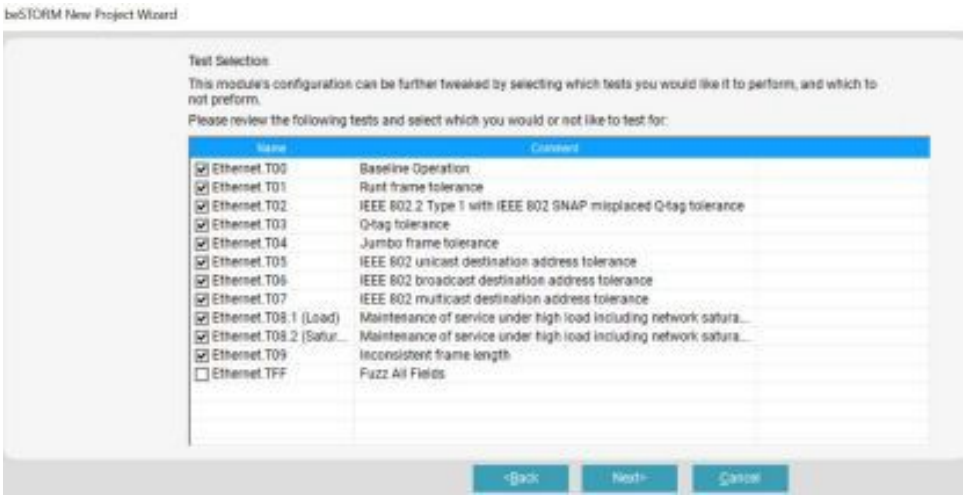
This fact can be verified and separately controlled by visiting the Monitor Settings screen.



Selection of Tests

The EDSA modules wizard allows predefined selection of tests to be included in a newly created project. The list of tests is corresponding to the numbered tests that appear in each module's specification, except for the "Maintenance of Service under high load" two phases being "split" to two separate tests (allowing inclusion or exclusion of a single phase) and the TFF test described below.

Exhaustive fuzzing



Under the hood, each protocol is described as the tree-space derived from the applicable optimized combinatory operations performed on its fields.

As such, each EDSA test can be described as a subset of that space, where fields or sets are locked to a certain value or put under certain restrictions.

Given that the combined subsets of the numbered tests might not cover all the optimized protocol-space, each module contains a last TFF (full fuzzing) test which then fully and "regularly" fuzzes all the fields in the protocol, much in the same way that most of the other "regular" modules are being fuzzed. This test covers aspects of requirements in the common requirements specification EDSA-310 that are not explicitly covered in the numbered tests. Note that this mode of fuzzing may need a very long time to run, and it is recommended to run it after all numbered tests have passed.

To run only the numbered tests required by the EDSA, make sure the TFF test is unchecked.