TPT 16 Severe Issues

Introduction

The following document contains a list of known severe issues of TPT. By severe issues we mean issues/bugs
in particular versions of TPT that:

1. might cause malfunctions in the behavior of TPT
2. are hard or even impossible to find by the TPT user herself/himself
3. cause the risk that bugs/defects in a SUT (system under test) are not detected by TPT in cases where
   TPT would have been able to reveal these bugs/defects in the SUT without the aforementioned
   malfunction in the behavior of TPT.

Usually these severe issues address the situations where the problem might appear and have well-defined workarounds.

ISSUE # P90208485-43040

TITLE: Enum import with duplicate constant can lead to an incorrect value for some of the constants to be imported to TPT

ISSUE DETECTION:
06/18/2024

AFFECTED VERSIONS OF TPT:
TPT 16 to TPT 20

PRECONDITIONS:
Enumeration data type with multiple constants using the same value present in C/C++ example or other data-source for interface import capable of importing enumeration data types to TPT.

DETAILS:
On interface import of an enumeration data type with more than one constant for the same value, the duplicate constants may get imported with an incorrect value.

EFFECT OF THE ISSUE:
The declared enumeration data type in TPT has incorrect values for the affected constants.

WORKAROUND:
Ensure that this use-case does not occur within the data source of the interface import before import the interface to TPT or manually review the constants for imported enumeration data types.

RESOLVED IN:
TPT 2024.12

ISSUE # 33413
==========
TITLE:
Assessment function REQUIREMENTS.checked() does not work correctly, if the second argument is a signal
or type boolean or int and not a verdict.

ISSUE DETECTION:
07-June-2022

AFFECTED VERSIONS OF TPT:
TPT 13 to TPT 18

PRECONDITIONS:
TPT Script Assessment contains REQUIREMENTS.checked("REQ-1",check_sig) and the second argument is a
signal and not the result direct via verdict.

DETAILS:
If inside a script Assesslet the function
REQUIREMENTS.checked("REQ-1",check_sig) is used with a signal
with a result, the function does not use the result of signal check_sig, instead the function uses the value
of this signal.

```
check_sig  = TPT.UInt8X()
check_sig := 0
check_sig.setResult(TPT.PASSED)
REQUIREMENTS.checked("REQ-1",check_sig)
```
In this case "REQ-1" is marked as FAILED, instead of PASSED.

EFFECT OF THE ISSUE:
Requirements get the wrong result.

WORKAROUND:
Use instead
```
REQUIREMENTS.checked("REQ-1",check_sig.getResult())
```

RESOLVED IN:
TPT 17u4, TPT 18u1

ISSUE # 33174
==========
TITLE:
TPT trigger rule, implausible interval discarding, when using "t" in 'Abort'/'Ignore intervals if' conditions.

ISSUE DETECTION:
02-May-2022

AFFECTED VERSIONS OF TPT:

TPT 13 to TPT 18

PRECONDITIONS:
The TPT project contains Trigger Rule assesslets using "t" in 'Abort'/'Ignore intervals if' conditions.

DETAILS:
In general, abort conditions influence whether an interval is used for analyzing THEN/ELSE-checks or not.
If an 'Abort'/'Ignore intervals if' condition is true within such an interval, the interval is discarded and
no THEN/ELSE checks take place in this particular interval.
The inconsistency arises from the use of "t" in the 'Abort condition' of "Trigger condition"-checks because
"t" refers for THEN-intervals to the *global* time (time elapsed since the trigger rule evaluation started),
but for ELSE-intervals to the *local* time (time since the beginning of the respective interval).
Furthermore, in the 'Ignore intervals if' condition of "While condition is true"-checks "t" refers for both,
THEN- and ELSE-intervals, to the *local* time (time since the beginning of the respective interval).
Since "t" in the START/STOP conditions always refer to the global time, this should also be uniformly
corrected to the *global* time in all 'Abort'/'Ignore intervals if' conditions. To avoid incompatiblity for
existing models, this behavior shall be able to explicitly be turned on/off in the TPT Tool Preferences
(settings per model).

EFFECT OF THE ISSUE:
The intervals discarded according to 'Abort'/'Ignore intervals if' conditions containing "t" might be
discarded or retained under implausible (but deterministic and well-defined) conditions.

WORKAROUND:
There is no real workaround, since the old behavior is not wrong! It is simply implausible or inconsistent
and should therefore be adjusted to avoid misunderstandings. For this reason, the behavior was only
adjusted via a preference option (configurable via TPT Tool Preferences): As long as this option is not
selected in old TPT models, the behavior remains unchanged even after the fix to maintain compatibility.
In new TPT models, however, the TPT Tool Preference option is automatically preset.

RESOLVED IN:
TPT 17u4, TPT 18u1

ISSUE # 31924
==========

TITLE:
RMI API is vulnerable to "CWE - 502 : Deserialization of Untrusted Data" due to used libraries.
Execution of arbitrary code is possible.

ISSUE DETECTION:
02-November-2021

AFFECTED VERSIONS OF TPT:
TPT 7 to TPT 17u1

PRECONDITIONS:
- RMI/Remote API is activated in preferences and library commons-collections-3.2.1.jar is present
  in your TPT installation directory in "lib\commons-collections-3.2.1.jar".
- firewall allows access of RMI API from other computers

DETAILS:
If RMI API is enabled and exposed by your firewall an unauthenticated attacker could execute arbitrary
code on the remote machine. Java cannot verify that serialized objects are well formed. In some cases
an attacker can use modified data to execute code during deserialization. There are some known "gadgets"
that can be used by an attacker. One gadget is the libray commons-collections version 3.2.1. The attack
can be prevented by restricting deserialization by using deserialization filters or upgrading the libraries
to unvulnerable versions.

EFFECT OF THE ISSUE:
An unauthenticated attacker could execute arbitrary code on the remote machine.

WORKAROUND:
Disable RMI API or replace "lib\commons-collections-3.2.1.jar" by commons collection version 3.2.2.

RESOLVED IN:
TPT 15u5, TPT 16u4, TPT 17u2

ISSUE # 31722
==========
TITLE:
If a test case containing a “Wait for Value” step with the „with assessment“-option being set or
a “Compare” step is executed with the FUSION platform with real-time option, it can happen that the
assessment fails although the “Wait for Value” step was executed properly or that the “Compare” step
is not calculated at all. Using FUSION with real-time behavior and test cases that contain
"Wait for value" steps with the „with assessment“-option might fail during assessment even if the step

has been executed successfully.

ISSUE DETECTION:
17-September-2021

AFFECTED VERSIONS OF TPT:
TPT 16, TPT16u1, TPT16u2, TPT 17

PRECONDITIONS:
- FUSION platform in use (or derived platforms that base on FUSION under the hood)
- Real-time node in use
- Test case contains step lists with either “Wait for Value” steps (with “with assessment” option being set)
  or “Compare steps” with a duration of just one sample.

DETAILS:
If a test case containing a “Wait for Value” step with the „with assessment“-option being set or a “Compare” step
is executed with the FUSION platform with real-time option, it can happen that the assessment fails although the
“Wait for Value” step was executed properly or that the “Compare” step is not calculated at all.
The problem occurs if the duration of the underlying "Wait for value"/“Compare” step is shorter than one sample
in "step size" (in terms of real-time).

EFFECT OF THE ISSUE:
The problem appears sporadically.
The TPT assessment dejitters all real-time test data before running the assessment. The sample rate for
dejittering is the configured sample rate of the FUSION platform. If a sample is shorter (in real-time) than
"step size", the sample will be omitted caused by the dejitter undersampling. Thus, the assessment does not see
the step being performed in the first place. The fix now that TPT will handle samples that are important for
such "Wait for value" / “Compare” steps for later assessment in a special manner to avoid being omitted while dejittering.
The bug does NOT affect the test execution itself, but the assessment only.

WORKAROUND:
Set the “Maximum increase in percent of one simulation step” Option in FUSION Real-time node config to “0%”.
However, since this setting makes the problem occur less often, but cannot avoid it 100%, it is best to apply
the update to TPT16u3 or TPT17u1.

RESOLVED IN:
TPT 16u3, TPT 17u1

ISSUE # 31641

==========
TITLE:
If an execution error occurs during the call to a TPT-Server function from within the SUT, the execution of the
function will stop but the error is not reported and the test continues.

ISSUE DETECTION:
03-September-2021

AFFECTED VERSIONS OF TPT:
TPT 16

PRECONDITIONS:
In the MATLAB/Simulink Platform the "Stub Simulink functions called from within the SUT" feature is used
and a runtime error occurs while the function is being called from Simulink.

DETAILS:
The "Stub Simulink functions called from within the SUT" feature enables the user to function called from
function caller blocks in Simulink with a specific implementation in TPT. In case such a function is called
from Simulink and the implementation causes a runtime error within the TPT VM (this may e.g. be an illeagal
array access) this error is not reported. Instead running the function will be aborted and the
test execution will continue.

EFFECT OF THE ISSUE:
In case a function stubbed by TPT is called from Simulink and causes a runtime error there is no error reported.
Any steps within the concerned function definition that occur after the error are not executed. In spite of
this error the test case result may still be passed.

WORKAROUND:
When using the "Stub simulink functions called from within the SUT" feature in the MATLAB/Simulink Platform
make sure the implementation does not cause any runtime error.

RESOLVED IN:
TPT 16u3

ISSUE # 31080
==========
TITLE:
When using the C/C++ Platform to connect a bit-field for which data type in TPT differs from the C-Code
the generated test frame might read or write the data incorrectly.

ISSUE DETECTION:
09-July-2021

AFFECTED VERSIONS OF TPT:
TPT 15 - TPT 16

PRECONDITIONS:
The C/C++ Platform is used to connect a C-Variable with a bit-field. The declared data type for the
bit field differs from the data type used for the corresponding struct element in the TPT declaration.

DETAILS:
When TPT generates the get/set functions to access the bit field data via the public TPT-VM-API function
"tpt_vmapi_bindSignalGetSet" or "tpt_vmapi_bindSignalGetSet_v2", TPT misstakenly used the type specified
in TPT when accessing the incoming pointer. Instead the data type passed to "tpt_vmapi_bindSignalGetSet"/"tpt_vmapi_bindSignalGetSet_v2" as fixed point data type shall be used.

EFFECT OF THE ISSUE:
At test execution with the C/C++ platform the data for a bit field value may be written to or read
from the SUT incorrectly. If the bit-size of the data type in TPT and the type of the bit field
does not match, the memory with the SUT or the TPT VM may be corrupted at test runtime.

WORKAROUND:
Manually check and adjust the code generated by C/C++ Platform in case of bit-field data types to use
the correct data type within get/set function used with "tpt_vmapi_bindSignalGetSet"
or "tpt_vmapi_bindSignalGetSet_v2".

RESOLVED IN:
TPT 16u3

ISSUE # 30102
==========
TITLE:
When comparing INT64 signals using Min/Max or Signal Comparison assesslet with values larger than 2^53 or smaller than -2^53, the computation can incorrectly compare the signal with the reference(s) which might lead to PASSED results even if the specified bounds are exceeded.

ISSUE DETECTION:
11-January-2021

AFFECTED VERSIONS OF TPT:
TPT 8 - TPT 16

PRECONDITIONS:
The Min/Max or Signal Comparison assesslet is used with INT64 signals with values larger than 2^53 or smaller than -2^53.

DETAILS:
If signal or reference values are larger than 2^53 or smaller than -2^53, the difference can be missed because the values are converted and compared as double values. (Since values larger/smaller than 2^53/-2^53) cannot be converted to double without losing precision, floating point precision problems might occur in such cases.)

EFFECT OF THE ISSUE:
Values outside of the specified bounds might be overlooked by TPT leading to PASSED results, but should be FAILED.

WORKAROUND:
Avoid using INT64 signals in Min/Max or Signal Comparison assesslets if the values are larger than 2^53 or smaller than -2^53.

RESOLVED IN:
TPT 15u4, TPT 16u1
ISSUE # 30063
==========
TITLE:
When iterating in assessment scripts via an inlined loop and applying signal processing functions like TPT.average(), TPT.min(), ... with changing argument in each loop, the result of the first iteration is incorrectly used for the following iterations.

ISSUE DETECTION:
16-December-2020

AFFECTED VERSIONS OF TPT:
TPT 8 - TPT 16

PRECONDITIONS:
The usage of the signal processing function must be applied on an expression or a signal that is being changed while iterating through an inlined loop or list comprehension.

DETAILS:
For faster computation of timed expressions of the form
 foo(t) := TPT.average(...)
results of signal processing functions are being cached.
The cached result is invalidated as soon as a new line is reached.
When iterating through an inlined loop, the same expression is evaluated multiple times.
If during this inlined iteration a variable of the expression is changed, the cached result of the signal processing function is used instead of a newly calculated.

Affected are inlined expressions of the form
 for x in range(n): print TPT.min(...+x)
 while x < n: x=x+1;print TPT.min(...+x)
as well as list comprehension:
 my_list = [TPT.min(...+x) for x in range(n)].

EFFECT OF THE ISSUE:
Old cached values are used instead of recalculating the value in every iteration,
which results in wrong computation results.

WORKAROUND:
Avoid using inlined loops or list comperhensions and use loops with indentation in multiple code lines instead.

RESOLVED IN:
TPT 15u4, TPT 16u1

ISSUE # 30064
==========
TITLE:
When running assesslets in parallel using the multicore feature and both assesslets
access the same signals imported from the same signal file, manipulation of such a signal
can be seen in the other parallel test run.

ISSUE DETECTION:
16-December-2020

AFFECTED VERSIONS OF TPT:
TPT 16

PRECONDITIONS:
The number of cores in the Execution Configuration dialog must be greater than 1 and
some assesslet executed in parallel must use the same signals imported from the same
signal file, e.g. via TPT.readRecord() or Import Measurements Assesslet and at least one
assesslet must manipulate such a signal. These assesslets may be the same assesslet linked
to multiple test cases.

DETAILS:
For faster execution imported signal files are cached. The cache is invalidated as soon as
a signal is manipulated. When running assesslets in parallel multiple assesslets can get a
reference to the same signal before it is manipulated. So any manipulation of the signal is
shared between these assesslets.

EFFECT OF THE ISSUE:
Assesslets see values and time intervals of signals set and created by other parallel running assesslets.

WORKAROUND:

When manipulation of imported signals (from external measurement files) in the assessment is
needed, do not run those test cases with multiple cores.

RESOLVED IN:
TPT 16u1