TPT 17 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 # 34814
=============
TITLE: Compiler-Bug when compiling index-based array-accesses for declared
constants that are arrays or structs containing arrays.

ISSUE DETECTION:
25-February-2023

AFFECTED VERSIONS OF TPT:
TPT 17 to TPT18u2

PRECONDITIONS:
A constant must be declared which has an array type, matrix type or a struct type
that contains an array.
In the test model there is at least an index-based access to this array constant,
i.e. "myarray[index]" where "index" differs from "0".

DETAILS:
Assume that the above conditions hold in a TPT model. Then the TPT compiler has
produced faulty code, so that instead of the access
"myarray[index]" always "myarray[0]" was used. Note that this error only occurs
with declared constants and **not** with declared
channels or declared parameters.

EFFECT OF THE ISSUE:
The wrong values of declared constant arrays in the test model are accessed and, if
used, might result in wrong test results.

WORKAROUND:
Convert all declared array constants to local array parameters. The mentioned error
does not occur with declared array parameters or array channels.

RESOLVED IN:
TPT 17u5, TPT18u3

ISSUE # 33294
=============
TITLE: Inconsistent name resolving during test execution and assessment if a name
refers to a struct member and an enum value.

ISSUE DETECTION:
18-January-2022

AFFECTED VERSIONS OF TPT:
TPT 17

PRECONDITIONS:
A struct channel and a type with the same name with equaly named struct element and enum constant.

DETAILS:
Assuming we have a struct channel "foo" with and struct element "elem1" and a type named "foo" as well
with enum constant "elem1". When accessing the value "foo.elem1" the name can refer to the struct element
or the enum constant. After a change in TPT 17 to make enum constants acessible if a channel name clashes
with a type name the case of further name clashes of struct element names and enum constant names was
overlooked and lead to inconsistent name resolving in such cases:
During test execution the name was resolved to the enum constant and during assessment the name was
resolved to the struct member. After the resolving the issue the enum value is now prioritized during assessment
as well.

EFFECT OF THE ISSUE:
Not the value intended is used either during test execution or assessment.

WORKAROUND:
Ideally use unique names for channels and types but at least unique names for struct elements and
enum constants.

RESOLVED IN:
TPT 17u3, TPT 18

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, 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, 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 # 31827

=============
TITLE:
Data from a previous test run is loaded and reported, if an error occurs during platform initialization.

ISSUE DETECTION:
08-Oktober-2021

AFFECTED VERSIONS OF TPT:
TPT 17

PRECONDITIONS:
- Test data from a previous test run exists while starting a new test run.
- The new test run has an error at platform initialization (e.g. if the test frame is missing)

DETAILS:
If the platform initialization fails (e.g. due to a missing test frame) test execution and result data from
a previous test run is loaded. The report will then be built based on this data.

EFFECT OF THE ISSUE:
The test report is built based on the data of the previous test run.

WORKAROUND:
Make sure to delete old test data before running a new test execution.

RESOLVED IN:
TPT 17u1

ISSUE # 32294
=============
TITLE:
Inconsistent behaviour between TPT model and assessment execution when accessing a struct member/
enum constant with same name.

ISSUE DETECTION:
18-January-2022 identified as Severe 28-March-2022

AFFECTED VERSIONS OF TPT:
TPT 17 to TPT 17u2

PRECONDITIONS:
The TPT project defines a struct member and a enum constant with the same full qualified name.

DETAILS:
In TPT 16 it was impossible to access an enum constant if the name of the enum datatype was identical to a

signal name, even if the given full name was unambigious. E.g enum constant "foo.bar" was not accessable
if a channel "foo" exists but "foo.bar" was not a valid struct member of the channel. The fix introduced an
incosistent behaviour if the name "foo.bar" is the valid full name of an enum constant and a valid full name
of a struct member: In the TPT model the enum constant was accessed and in the assessment the value of
the struct member. This may lead to checks with an unintended value.
Since TPT 17u3 the behvaiour is consistent and in accessment an enum value will be accessed in favour of
a struct member.

EFFECT OF THE ISSUE:
Unintended struct member access by using the value ot the struct member instead of the enum constant
for checks in assessment.

WORKAROUND:
Ensure the names of struct members and enum constants are disjunct.

RESOLVED IN:
17u3