TPT 13 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 # 33413
=============
TITLE:
Assessment function REQUIREMENTS.checked() does not work correctly, if the secound 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 secound 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 # 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 # 29261
=============
TITLE:
When executing tests using the FUSION platform,the mechanism "Read parameters from FUSION only once (before first test case)" does not ensure that the first test case has completed the parameter exchange before the following test cases use the parameters.

ISSUE DETECTION:
4–September-2020

AFFECTED VERSIONS OF TPT:
TPT 12, TPT 13, TPT 14, and TPT 15

PRECONDITIONS:
The checkbox "Read parameters from FUSION only once (before first test case)" must be selected in the FUSION platform and the number of cores in the Execution Configuration dialog must be greater than 1.

DETAILS:
When running tests in multicore mode using the FUSION platform and the option "Read parameters from FUSION only once (before first test case)" is selected, the test cases are executed immediately without waiting until the parameters have been exchanged.

EFFECT OF THE ISSUE:
For tests executed in parallel (i.e. all but the first test case), not the parameter values determined during a parameter exchange are used, but the default values from the Declaration Editor (nondeterministic behavior).

WORKAROUND:
Deselect the "Read parameters from FUSION only once (before first test case)" checkbox in the FUSION platform configuration, or do not run tests in multicore mode.

RESOLVED IN:
TPT14u3, TPT15u2

ISSUE # 26884
==============
TITLE:
Local search and replace in step lists always replaced all occurrences in table steps at once and so may replace entries unnoticeably the user did not intend to change.
ISSUE DETECTION:
15-Jul-19

AFFECTED VERSIONS OF TPT:
TPT 12 – TPT 13

PRECONDITIONS:
A step list including a table step with multiple occurrences
of a search term and trying to replace one of these.

DETAILS:
If a search term occurs multiple times in a table step trying
to replace one of them using the search dialog always replaces
all occurrences in the table step. Generally, the issue
affects all tables embedded in other tables but the table step
is the only one where this condition applies.

EFFECT OF THE ISSUE:
Due the unexpected behavior of search and replace more changes
than intended may be done to table steps leading to unintended
TPT model behavior.

WORKAROUND:
Using global search and replace to replace occurrences of a
search term in table steps. Changes done by local search and
replace affecting table steps have to be reviewed carefully.

RESOLVED IN:
TPT13u2

ISSUE # 26796
==============
TITLE:
If explicit array-of-struct values are being used in TPT
models (e.g. "mychannel := [{2,3},{4,param+5}]") these
expressions can compute wrong values at runtime if the arrayof-struct expression refers to non-constant
channels or
parameters.

ISSUE DETECTION:
01-Jul-19

AFFECTED VERSIONS OF TPT:
TPT 12 – TPT 13

PRECONDITIONS:
TPT computes wrong values at runtime under the following
conditions: an explicit array-of-struct value is assigned to a
channel orparameter (e.g. in a step list) AND the array-ofstruct value itself depends on channels/parameters
AND•these
channels/parameters are not constant (at runtime) AND these
channels/parameters have values that differ from their default
values (as specified in the declaration editor).

DETAILS:
In the special case of explicit array-of-struct expressions
(e.g. “mychannel := [{2,3},{4,param+5}]”) any reference to
channels/parameters inside this expression will not be
computed dynamically (at runtime), but computed just once when
initializing the expression (before runtime). Therefore, if
the value of channels/parameters that are referred inside this
expression changes at runtime, the array-of-struct expression
will not update its value.The fix now explicitly prohibits the
usage of (a) references to all channels (b) references to all
parameters that are modified dynamically in the test model
inside any explicit array-of-struct expression. Otherwise the
TPT compiler will raise a compile error.

EFFECT OF THE ISSUE:
In the special case of explicit array-of-struct expressions
(e.g. “mychannel := [{2,3},{4,param+5}]”) with references to
non-constant channels or parameters, the value of the arrayof-struct expression gets stuck with its initial value for
all points in time even if the referred channels/parameters
change. After the fix related to this issue, those expressions
are denied by the compiler.

WORKAROUND:
Instead of assigning the whole array-of-struct use individual
assignments for non-constant elements.

RESOLVED IN:
TPT13u2