Project security checking implementation on MPF

Primary scenario: The project file should be checked for security the same way as the language projects are doing.

1. Checking for security on the language projects means checking the followings:
2. Checking for dangerous targets. All targets defined within the LoadTimeSensitiveTargets are considered protected.
3. Checking for dangerous items. All items defined within the LoadTimeSensitiveItems are considered protected.
4. Checking for dangerous properties. All items defined within the LoadTimeSensitiveProperties are considered protected.
5. Checking for unsafe imports. All imports that are not defined in the <VSRegistrythive>\ MSBuild\SafeImports are unsafe. For a user project no imports are allowed.
6. Checking for unsafe using tasks. All using tasks defined in the project file are considered unsafe.
7. Checking for items considered to be defined on safe locations. The items defined within LoadTimeCheckItemLocation are checked whether they are defined in safe locations.
8. How it is implemented on MPF?

Security checking is done during project load, before any project state, but the project instance guid is set.

1. The package is asked to read the suo file to see if in any previous runs this project instance was marked as trusted. If yes the project loads.
2. If not, the project is asked to be checked for security.
3. Checking for security means that the project and possibly the associated user file if it exists are checked for security.
4. The files are fully checked, contrary to the language projects that just check the first failure and if the user considered the project trusted no more checks are done even if there are more violations.
5. First the project file is checked and if that is safe or it is chosen to be loaded than the user project is also checked for security.
6. If there are security violations a security warning dialog is shown that will give the user the option: to load the project for browsing, load it normally, or do not load it. If both the project and user file contain violations and the project is said to be loaded normally the second dialog for the user file will override any options made for the project file. In one word the user choices from the second dialog wins.
7. If the project is loaded normally then an entry is added to the suo file that this project instance should not be checked again.
8. Static class structure

The following describes the static class structure.



The design tried to make it possible that extenders can:

1. Extend projects security checking by plugging in their own, both for project and user files. ProjectSecurityCheckers are created through polymorphic factory methods in the project node.
2. Replace the UI by a new one by overriding the ShowSecurityDialogBox method.
3. Sequence diagram

The below describes in overall, without entering in more details what happens during project security checking.

