This is a speclet for the F# Type Provider Security Dialog feature.  We iterated on the design a bit, so I’m just authoring a new document from scratch that summarizes the feature’s intended behavior.

**Goals**

Goal: The goal of the feature is to protect the user from having projects/scripts run code from arbitrary type provider DLLs merely by viewing or building the code in VS.  The user must first give consent before any code from the referenced TP DLL will run inside the VS process.  The dialog informs the user of both the security and privacy risks of allowing the type provider to run.

Non-goals: protection from command-line build with fsc.exe or msbuild.exe, or F# Interactive code running (either command-line fsi.exe or inside VS F# Interactive tool window)

**Basic model of behavior**

Type providers can become ‘trusted’ by the user approving the use of a particular filename.  A given filename path is always in one of three states:

* TRUSTED – VS has shown the user a dialog at some point in the past asking if the user trusts this provider, and the user explicitly enabled it
* NOT\_TRUSTED – VS has shown the user a dialog at some point in the past asking if the user trusts this provider, and the user explicitly disabled it or otherwise canceled the dialog
* UNKNOWN – VS has not recorded any trust decisions about this, and will prompt the user

The TRUSTED and NOT\_TRUSTED states are recorded in a file under the user’s local AppData, e.g.

C:\Users\brianmcn\AppData\Local\Microsoft\FSharp\3.0\type-providers.txt

which is a simple text file where each line has one trust decision and a filename:

TRUSTED C:\VSPro\_FSharp\binaries\x86chk\SuiteBin\FSharp\FSharp.Data.TypeProviders.dll

NOT\_TRUSTED C:\VSPro\_FSharp\binaries\x86chk\SuiteBin\FSharp\Samples.CSVTypeProvider.dll

Any filename not mentioned in this file is implicitly in the UNKNOWN state.

If the user does some action in VS that needs to use the TP, namely:

* Opens a script that references a TP (the language service needs the TP to get IntelliSense), or
* Opens a code file in a project that references a TP (the language service needs the TP to get IntelliSense), or
* Builds a project that references a TP (the compiler needs the TP to build)

then the referenced TP will be checked against the recorded state in the AppData file, and

* If that TP is TRUSTED, then fine, it just works
* If that TP is NOT\_TRUSTED, then the TP is not loaded by the language service or compiler, and a warning like “Type provider assembly 'C:\foo.dll' is not trusted and will not be loaded for security reasons. This may cause subsequent build errors. See the 'F# Tools' section of Visual Studio options for more information.” is issued.
* If that TP is UNKNOWN, then the modal security dialog appears on the screen (screenshots further below) and the user must make a decision, which will put that file into either the TRUSTED or NOT\_TRUSTED state, at which point we behave like one or the other of the previous two bullets

If the user wants to make changes to prior decisions, they can visit the VS Tools\Options (screenshots further below) and toggle the trust mode for a filename, or delete all record of a decision for a filename (which puts it back in the UNKNOWN state).  (Note: there’s no explicit way to add a new file to the TP list in Tools\Options, but you can always just open an F# script file and say #r @”filepath” to make the dialog pop up, at which point a decision is recorded and the file will now appear in Tools\Options.)

**Some behavior details**

VS handles any updates to the TP states ‘transactionally’, in that any time the dialog or Tools\Options page wants to record a change, it takes a lock on the AppData file.  There is an intrinsic race if you have multiple instances of VS open and go through approval UI on both at the same time.  I think the behavior is something like “last VS to write the file ‘wins’ with its own view of the world”, but the exact specific behavior is not too important (this is very uncommon, and users are unlikely to encounter this), the only thing that is important is that we should never end up saying a file is TRUSTED when the user did not explicitly trust it.  Stated another way, when multiple instances of VS are trying to “simultaneously” write the file, we may lose some of the user’s recorded settings, but that is merely a usability annoyance (rather than a security bug) for this corner case scenario.

Filenames are normalized via “(new System.IO.FIleInfo(filename)).FullName” and compared case-insensitively, which means that if you have approved “C:\FOO.DLL”, then also “C:\\FOO.DLL” (extra slash) and also “C:\foo.dll” (different casing) are also treated as approved and do not introduce a different entry in the AppData file or the Tools\Options page.

If at any point the AppData file becomes “damaged” (cannot be read/parsed, or has inconsistent data, see below), then VS may delete the file and start again.  Most importantly, if we cannot successfully read the file, we behave as though all TPs are either NOT\_TRUSTED or UNKNOWN (the point being, error conditions should not spuriously cause a file to become TRUSTED when the user did not explicitly approve it).  A couple ways the AppData file may become damaged are: (1) the user could hand-edit the AppData file, and add garbage that does not parse, or add inconsistent data (e.g. say that FOO.DLL was both TRUSTED and NOT\_TRUSTED on two different lines of the AppData file), or (2) the user could have two instances of VS both pop the dialog up for the same UNKNOWN file and ‘enable’ in one instance and ‘disable’ in the other, which may record the inconsistent data to the file, at which point VS will see the file as ‘damaged’ the next time it reads it.  These scenarios should be very rare; by deleting the file and starting over, we may create a minor usability annoyance, but do so in order to ensure that security is always preserved.

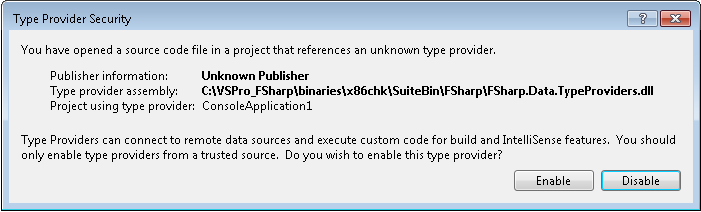
Doing a ‘build’ in VS (which calls fsc.exe out-of-process) is protected by having F# projects built from VS always pass the –validate-type-providers flag to the F# compiler.

**Screenshots**

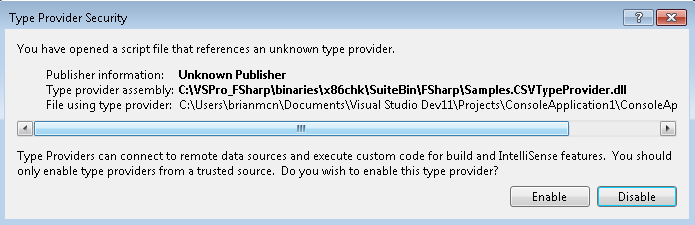
(Screenshots here are informative/suggestive; the actual final product wording/layout may be a little different)

(If a TP DLL is authenticode-signed with publisher info, the publisher info would be presented in the dialog, else it will show ‘Unknown publisher’ as seen below.)

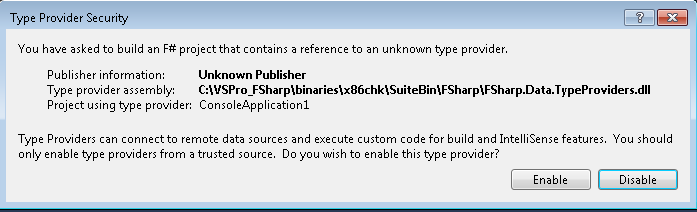
Opening source code in a project that references an UNKNOWN provider yields



Opening a script that references and UNKNOWN provider yields



Building a project that references an unknown provider yields



The Tools\Options page allows users to inspect which files are approved, toggle approvals, or remove files (revert them to UNKNOWN state):

