Signatures Technical Details

There are various signature databases, one for each platform. This speeds up the scanning process, since it makes no sense scanning for a x64 signature in a x86 portable executable. Nevertheless, there's also a multiplatform database called "PLATFORM_INDEPENDENT". This database contains signatures that are valid on all platforms. For instance, a .NET obfuscator could (it's not a rule, just a possibility) have the same signature on x86, on x64, on IA64 and on ARM (basically on every platform which supports .NET): keep in mind that .NET executables are not always x86 executables, they can also be compiled specifically for x64, IA64 or ARM. All other databases are named by the platform they represent based on the Machine field in the PE File Header, e.g.:

- IMAGE_FILE_MACHINE_I386 (x86)
- IMAGE_FILE_MACHINE_AMD64 (x64)
- IMAGE_FILE_MACHINE_IA64 (IA64)
- etc.

The database format is XML. I chose this format instead of the old ini-like format for various reasons:

1. The parsing is faster.
2. Adding new fields is easy, so changing the database schema isn't drastic.
3. The format is more transparent than a proprietary (created by me) format would be.

The third point implies that all signatures are public, it doesn't make much sense either that every program has its own internal database. The XML schema of a database looks something like this:

```
<?xml version="1.0"?>
<SIGNATURES>
   <ENTRY>
      <NAME>Name of the signature</NAME>
      <COMMENTS>Eventual technical comments</COMMENTS>
      <ENTRYPOINT>1A2B??3C4D</ENTRYPOINT>
      <ENTIREPE>5F6E7A8B</ENTIREPE>
   </ENTRY>
</SIGNATURES>
```

SIGNATURES is the XML root element, there's only one element with this name.

ENTRY is the root element for a signature. If a database contains 10 signatures, there will be 10 ENTRY elements, one for each signature.

NAME is the name of the current signature.

COMMENTS is a field designed to contain (only) technical information about the protection / packer / crypter / compiler referred by the current signature. Please, don't use this field to include other kind of information.

ENTRYPOINT and **ENTIREPE**, these two are the actual signatures of the current entry. One of this two fields can be empty. In already existing signature

databases there's only one signature and a following parameter which tells the scanner if that signature is to confront only with the entry point or to search in the entire executable. I could have done things in the same way, instead I chose to put two distinct tags. The reason is that the same entry could contain an entry point signature but also a signature which could be anywhere in the portable executable. In the case of a deep scan, the matches of the signature to search in the entire executable will be added to the entry point signature matches. This means that if an entry point signature is 20 bytes long and produces 20 matches and an entire PE signature is 40 bytes long and produces 38 matches, the final result will be of 58 matches. 58 matches for just one entry is better than writing two separate entries, one that scores 20 and the other that scores 38. It's just more effective.

As you may have already noticed, the signature length does not always equal the number of matches. That's because wildcards in signatures don't increase the score. Let's take for instance the signature: AABB??DD. This signature is 4 bytes long, but only produces 3 matches. That's also the reason why signatures shouldn't end with wildcards, since they are meaningless at the end of a signature, e.g.: AABBCCDDEE?? should be written AABBCCDDEE, since wildcards exist only to link static bytes, without a static byte following they are meaningless. The software deletes wildcards at the end of a signature automatically, so there's no need to stress further on this point.

The convention I adopted is to store signatures without spaces (to reduce the database size) and in upper-case (to give a style unity and to give the chance to speed up the conversion from string to numbers).

My only regret about the database format is that the scanning process would have been way faster (or let's say optimized) if in addition to the entry point and entire PE signature, I had added specific location tags like: <CODESECTION>, <ENTRYPOINTSECTION>, <IMPORTEDDLL> etc. The reason I couldn't do this is that it would have increased too much the database complexity and it would have made it impossible to look after collisions (too much computing time necessary).

Daniel Pistelli