FAQ on Software and File Protection Techniques

Q1. What are various ways to protect software?
Mainly there are two techniques: 1. Authentication and 2. Code Porting

Q.2 What is Authentication Technique?
Authentication means verifying or validating a string or number or a combination. For example using static software license key or verifying hardware serial number of a USB flash or CPU while software is running is authentication. Similarly verifying or validating a key string stored inside the dongle is also authentication.

Q3. What code porting technique?
Software code consists of many procedures and functions. As a common practice several functions are stored in a library files [like .lib, .dll etc] In Code Porting one or more functions are stored inside the dongle. Software calls these functions from inside the dongle. Because there is no way to extract the function from inside the dongle, this technique is considered most safe and can even be used in open source software.

Q4. What are various methods used in protect software?
Though there are many methods, few listed below are important: 1. Verifying hardware id like CPU ID, Hard disc Number, USB flash number, Motherboard serial number or any other hardware. 2. During specially designed USB dongle for software protection 3. Using 3.5 mm head phone socket dongle 4. Using printer port dongle 5. Using static licensing keys 6. Using licensing keys based on hardware serial numbers and activating online by email or website. [*Net Activation from SoftwareFileProtection.com uses this technique*] 7. Using online portals and allowing user to work on software till he is logged in. This allows us to issue license for few days also for demo software.

Q5. What is reverse engineering? And How it affects software protection?
Reverse engineering means decompiling the software i.e. extracting the code again. By decompiling one can bypass the authentication part of the code and recompile the application again. And make it freely distributable application which will run without authentication. If you are using code porting, you are safe because important code or functions are stored inside dongle.

Q6. How to prevent reverse engineering?
There are no hundred percent safe methods. You can only make reversing little more difficult. **iBinder** envelope from SoftwareFlileProtection.com is one such solution. For .Net uses obfuscation is one such solution. You can implement multiple methods and make reversing more and more difficult.

Q7. What are limitations of using hardware serial number based software protection?
1. Software will work only on one system on which it is licensed 2. If the user changes hardware, either he has to buy new software or you need to give another authentication free. This imposes lot of restrictions on the customer in using your software.

Q8. What are risks involved in using hardware serial number based software protection?
1. There are techniques available to create the hardware with same serial number by changing the serial number of any existing hardware or same or similar configuration 2. We are simply saying this is possible, it does not mean everyone can do it.

Q9. What are limitations of using USB Dongles?
1. USB dongles do not impose restrictions but gives freedom to users to use it on any system. 2. He/she can even resale the license [if not restricted by terms of use].

Q10. What are risks involved in using Authentication Dongles?

Authentication dongles are available in many designs and hence it is difficult to provide risks specific to dongles unless there design is known.
Though some generic risks are as below:

1. In general authentication dongles have three keys or values:
    a. ID of the dongle which makes it specific to program of software developer. This can be called software ID, Developer ID, Vendor ID, Private Key etc. In all dongles assigning this key is first step.

        This can be given by you [software developer] or dongle maker or its distributor.
        If this is decided by you, you need to keep it secret from your programmers or create multiple levels in your administration and ensure that "how to assign this key" is known to some higher level persons only.

        If this is decided by the dongle maker or its distributor, you should preferably sign a Non-Disclosure-Agreement (NDA) with the dongle maker Or its distributor.

    b. Password to write dongle
    c. Password to read dongle
    d. Keys stored in the dongle

        These are always decided by you. Again create multiple levels in your administration and ensure that these data are known to persons at specific level only.

        The involved risk is that if the ID of the dongle is created by dongle maker, it is possible that he can create similar dongles and hence first level of security is breached.

        If due to faulty dongle design or by using some special tools a person gets hold of the ID of the dongle or how this ID was created, he can purchase dongles with ID similar to yours and hence first level of security is breached again.

        The second important risk is from your own employees who know too much about your software protection. The only way is to sign a NDA with employees and ensure that only minimum required things are known at each level of your software development administration.

    This is one of the known facts that human involvement is weakest link in the security chain.
    Though NDA do not stop a person from cheating. This is still considered best tool as it provides string legal safety.

Q11. What are risks involved in using Code Porting Dongles?
Code porting dongles are considered best available solution. Code porting means storing some functions of your software inside dongle. You need to write that function in C and port it as HEX file which will be stored inside the dongle. This requires use of software like KJEIL. You can call the function from your software. This is also good way to protect open source software, dot net software and others which are easy to reverse engineer. For example in the software that calculates material requirement for a building based on **AutoCAD** design will require mostly formula based calculations. These calculations can be ported inside the dongle making software almost impossible to crack.

Q12. What envelope protects from reverse engineering and what is level of safety?
As replied in a similar question previously, no technique is hundred percent safe. Envelope developed by SoftwareFileProtection.com adds a binary layer over the software application. It also creates the application in virtual environment when run. The binary layer ensures that normal de-compilers do not work. Though it should be tested for every specific use.

Q13. How a software license can be broken?
This is very important question and every programmer must understand this so as to decide what protection and licensing he/she should use. Please note following points: 1. Using easily available DEBUGGING software like WinDbg to bypass authentications functions and recompiling. 2. Using HEX editor to bypass authentications functions and just saving. 3. Using Cache or memory editor and editing code while the demo software is running and removing demo limitations and recompiling or saving 4. Creating a DLL of same name as used to connect dongle functions and bypassing hardware check and supplying required returns directly. To stop this, ensure to validate DLL using MD5 string. If programming language like C++, lib is integrated inside the EXE which makes it safer. 5. If connecting port of dongle like USB is not encrypted it is easy to know what is stored inside dongle this helps in making crack defined above. 6. If licensing is based on static keys, a key generator can be made easily. There is much more to cracking and reversing which cannot be explained here as that is not the objective of this document. Please do research for programming language used by you or consult an expert.