MODULE 8 CODE OBFUSCATION ATTACK SCENARIOS

Introduction –

Obfuscation is a distinctive mechanism equivalent to hiding, often applied by security developers, to harden or protect the source code (which is deemed as intellectual property of the vendor) from reversing. The goal of such an approach is to transform the source code into new encrypted byzantine source code symbols which have the same computational effect as the original program. By applying effective obfuscation over the source code, it is difficult for a vicious-intentioned person to analyze or subvert the unique functionality of software as per his requirements. Vendors typically seem to be safe by ensuring obfuscation over their intellectual property, but unfortunately, software code is not safe from being modified even after applying obfuscation; it still can be cracked. However, this phenomenon can be illustrated by applying sort of rare tactics to bypass the obfuscation mechanism in order to reverse engineer or alter the inherent functionality of software.

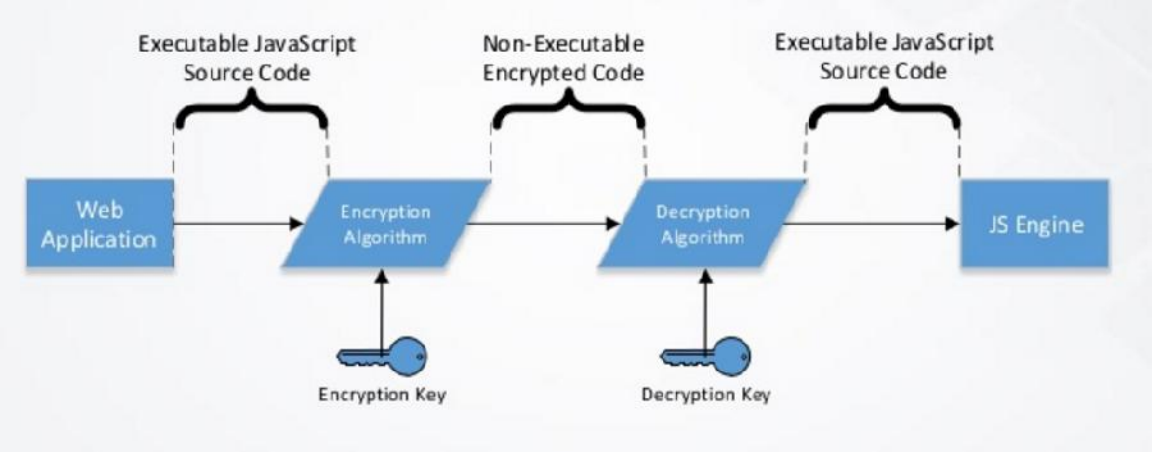
Obfuscation Analysis

It is a very difficult and often time-consuming process to reverse engineer a compiler-generated code, especially as things gets even worse when machine code is in encrypted or obfuscated form. Such compiler-generated code is deliberately constructed in encrypted form to resist analysis from reverse engineers. Some examples of situations in which obfuscation might be applied are as follows:

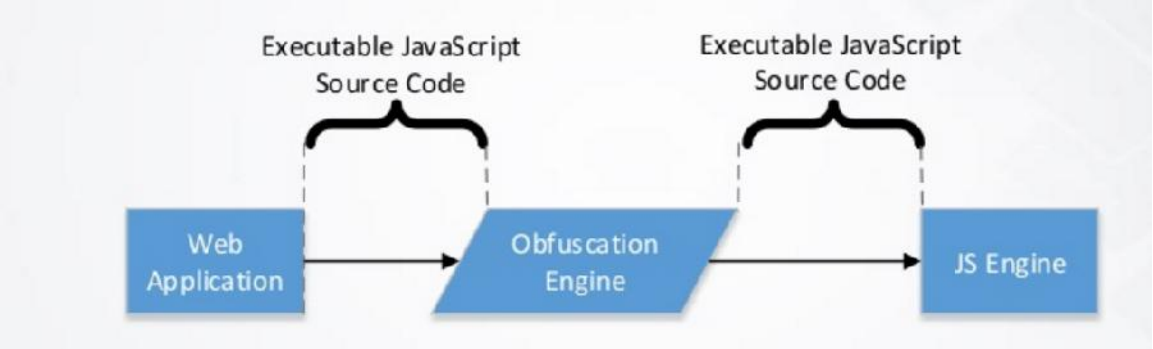
- **Protection of intellectual property**—Commercial software typically has protection against unauthorized duplication by employing further obfuscation for the purpose of obscuring the implementation particulars of certain crucial segments of the mechanism.
- **Digital Rights Management**— leading contemporary applications are often obfuscated by employing DRM schemes, which commonly protect certain crucial pieces of information (e.g., protocols and cryptographic keys) using obfuscation.
- **Malware**— Hackers and reverse engineer criminals practice obfuscation for avoiding the detection of malware signature from anti-virus search engines.

Obfuscation not equal to Encryption –

There is always a misconception that obfuscation is the encryption of the code, but it is not. The encrypted code is not executable by the browser as it will not have a key to decrypt it and render into the browser. The below diagram explains the process in a nutshell.



JavaScript obfuscated code is still valid and ready to execute with the browser, the obfuscation doesn't require any de obfuscation technique for the revival. The below diagram explains the process.



Differences of using obfuscation –

The Good part –

1. It prevents code theft and reuse
   Its very easy to stop your competitor from using any of the proprietary code and start his own.
2. Protect intellectual property
   Obfuscation can hide algorithms.
   Hide data – Many a times developers use some hard coded data into the code due to system limitations.
3. Enforce license agreements –
   For example you ca lock the domain with some lock code inherited in to the obfuscated code.
4. Extra security layer

Attacker has to put some extra effort to find vulnerabilities on the client side, and also you can test the strength of the network monitoring tools like IDS/IPS/WAF etc.

The bad part –

What if the obfuscated code bypasses the network monitoring tools by getting inserted with some malicious code by making the code look like less harmless.

Attack on Obfuscation through Reverse Engineering

Reverse engineering is the process of extracting the knowledge or design blueprints from anything man-made. Reversing is used extensively in both ends of the malicious software chain. Developers of malicious software often use reversing to locate vulnerabilities in operating systems and other software.

Reversing Crytpo –

Cryptographic algorithms can be roughly divided into two groups: restricted algorithms and key-based algorithms. Restricted algorithms are secret once the algorithm is leaked, its exposed. And on the other hand key based algorithms are well knows, every one knows the technology behind the encryption, but only the key is kept secret. To reverse engineer this you have to either:

1. Obtain the key
2. Try all possible combinations until you get to the key
3. Look for a flaw in the algorithm that can be employed to extract the key or the original message

Static Code Analysis –

The one way of attacking obfuscation is using a static code analyzer, by following steps:

Parsing the code – Line by line code parsing which can be done manually or by using automated tools. There are lot of tools in market which can be used for source code analysis.

Dynamic Code Analysis –

The analysis is performed by executing with the code, we need to retrieve the CFG (Control Flow Graph) of the code executed and retrieve values of some expressions.

You need to understand one instance of program at a time and retrieve the values of expression. It’s something similar to find a needle in the haystack by finding some encryption key within the code. The dynamic analysis also bypasses the dead code within the application.