Cyclomatic Complexity

Cyclomatic Complexity quantifies the number of linearly independent pathways through a section of programming code. Code can be represented by a Control Flow Graph with each command represented by a node (N). These nodes are joined by edges (E) which represent the flow of the code. The code may exit at one or more points (P) or connected components. This can be defined by Mc Cabe's formula:

$M = E – N + 2P$

Question: Is Cyclomatic Complexity relevant today?

Cyclomatic Complexity was formalised in the 1976 when computing processing power was limited. It will often be the case that a more complex code will demand more processing power and while the measure of Cyclomatic Complexity is targeted at describing complexity, it may also be an indicator to efficiency. It could be argued that the increased processing power available today limits the relevance of Cyclomatic Complexity as machines are able to handle much more complex tasks with ease. However, fundamentally, there is a craft to coding and, regardless of processing power available, an efficient solution should always be preferred, as there is a parallel with elegance in mathematics. Furthermore, there are implications for the development cycle of secure software.

Question: What is the relevance of Cyclomatic Complexity in regard to software security?

Cyclomatic Complexity is extremely relevant and important with regard to secure software development. PEP 8 Guidelines highlight the importance of readable code. It is a fact that the more complex the code is, the harder is it to read. This is important when considering the pillars of software security.

Testing and Debugging

The more complex the code, the harder it will be to debug and fix any problems. This means that complex code is inherently less secure as vulnerabilities may be harder to identify and fix.

Maintenance

An important goal of secure software development is to ensure that the system can be maintained. It may be that there are new or emerging threats to a system that did not exist at the time of original development. To continue to maintain the high level of security to a system, updates will be required. Where the Cyclomatic Complexity of a system is high, updates will be harder to implement. Conversely, where there is Cyclomatic Complexity is kept to a minimum, updates will be easier to implement. This, in turn, will extend the overall lifecycle of the system.

Agile Development

The advent and growth of agile software development means that teams of programmer will be working on complex problems with a goal of quick and efficient development. It is always important that the speed of development does not compromise the security of the system. Minimising Cyclomatic Complexity will ensure that code can be shared across the team which will enhance the security.

Conclusion

Cyclomatic Complexity is still relevant today. While processing power, may well enable more complex, and sometimes, inefficient code, minimising Cyclomatic Complexity will promote and enhance security of a system. The advantages of minimising Cyclomatic Complexity are that:

1. Code is more readable
2. Code is easier to debug and test
3. Code can be maintained more easily
4. Code can be shared and re-used across the team or for future projects

These factors enable better threat testing and adaptability to any new or emerging threats.