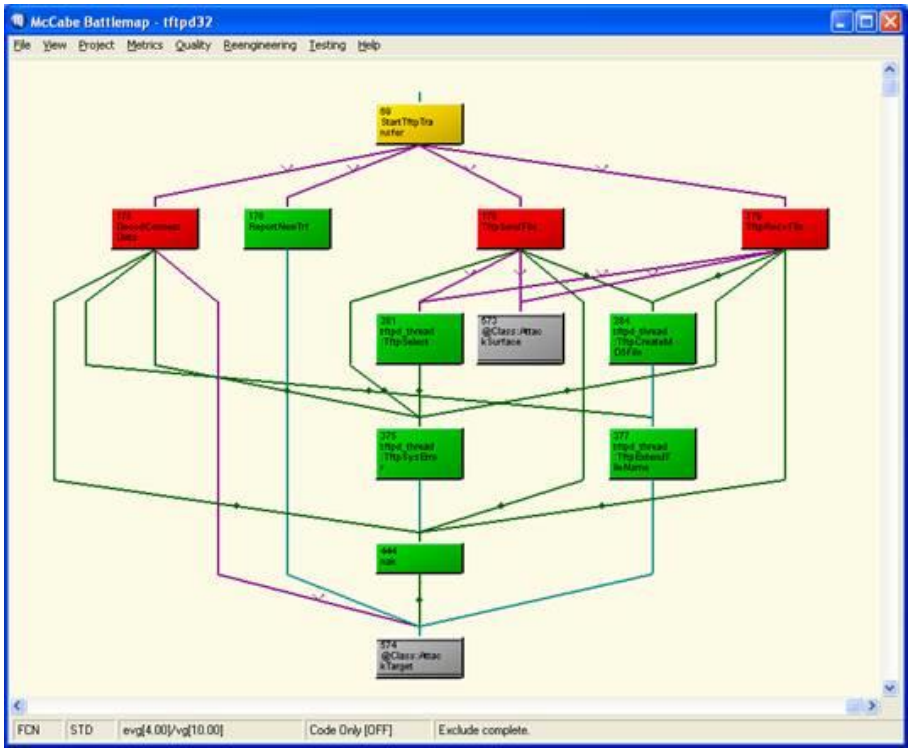
Security Risk Identification:
12 Application Architecture Categories to Review

Static code analysis provides an excellent technique to quickly and easily identify potential security vulnerabilities and weaknesses in applications. However, as static code analysis tools have improved, organizations are becoming overwhelmed with lists of potential vulnerabilities, often without having the resources to address more than a fraction of them. In reality, a large proportion of these so-called vulnerabilities do not present a real security risk because of their nonstrategic location in the application architecture yet traditional static analysis for security vulnerabilities do not provide a way to identify the vulnerabilities that present a real security risk.

Therefore, many organizations are seeking automated methods to facilitate analyzing the architecture from a security standpoint. Utilization of automated methods would optimize the allocation of scarce resources who have been tasked with analyzing identified potential vulnerabilities, while having the greatest positive impact on security risk. McCabe IQ can expedite this process by decomposing the application into detailed, color-coded flow maps that simplify the task of evaluating the application's architecture from a security risk standpoint, thereby reducing the time required to assess potential vulnerabilities by an order of magnitude or more when compared to working with raw code. As an example, see the McCabe IQ flow map (structure chart) below depicting how certain code represented by the grey boxes can interact with each other.



The 12 most critical categories of risks to an application are shown in the chart below. This article explains how the first seven of these categories can be analyzed with McCabe's flow maps. Note that similar techniques can be applied to the other five categories.

Architecture Risk Review Categories
1. Authentication
2. Session Management
3. Cryptography
4. Error Handling and Logging
5. Data Protection
6. Malicious Code or Controls
7. Communications Security
8. Access Control
9. Input Validation
10. HTTP Security
11. Business Logic
12. Files and Resources

1. Authentication

A holistic examination of the authentication process architecture without having to sift through the syntax of the code helps an analyst understand very quickly how the user is being authenticated. Are single factor, or multi-factor authentication used and does each application use the same or different methods? Multi-factor authentication is more secure because with single-factor authentication a potential intruder need only obtain the username and password, for example, to gain access. Assigning tokens or answering secret questions in combination with a username and password provides for greater protection. Likewise application functions that display sensitive data should be checked to ensure that the user is authorized to view that data.

2. Session Management

Web and mobile applications are normally structured so that the client consumes a service over the internet so session management is critical to security. From an architectural perspective, the first step typically is to make sure the application is properly managing the session. It can usually be quickly identified from the visual representation provided by McCabe IQ. Then take a deeper dive into how the session is managed, such as by making sure the session ID is changed each time the client communicates back to the server, all information is encrypted before being sent (HTTPS), etc.

3. Cryptography

Most applications have a cryptographic function that is responsible for protecting data at rest and in transit. A flow chart can be used to verify that cryptographic modules fail securely rather than leaving the data open where it might be compromised. The specific cryptographic algorithm should be identified and assessed. The random number generator plays an important role in most every encryption algorithm by generating a key or passphrase. McCabe IQ makes it easy to drill down to see what type of random number function is being used and whether it is coded from scratch or called from a library. In addition, the use of an outdated or custom cryptographic algorithm can lead to data being left unprotected as many custom cryptographic algorithms tend to be weak. Outdated cryptographic algorithms are algorithms that have already been proven to be compromised.

4. Error handling and logging

Error handling and logging are very important from a security perspective because, for one reason, by identifying abnormal behavior they often provide the first indicator that an attack is underway. By examining a visual representation of how the application is architected, you can quickly identify the central error handling and logging capability and analyze how it works. The application should be investigated to validate it always fails in a secure state to ensure that a hacker does not intentionally cause the application to fail in order to gain access.

Logging should not be limited to simply recording errors but should also create an audit trail that can be used to provide a warning of unusual behavior. For example, you should be able to see what credentials a user logged in with, see the records they accessed, see what data they added, what they deleted, etc. With that capability in place, an alert can easily be issued when, for example, a user that normally accesses certain functionality within the application suddenly uses other functional areas indicating that his or her credentials may have been compromised. The McCabe IQ chart can be used to make sure that all critical areas of the application have sufficient error handling and that logging is concentrated in the most important areas of the application.

5. Data Protection

Many applications perform some type of caching to improve performance so it's important to make sure that the cache is deleted when the operation is completed so that it cannot be compromised. The McCabe IQ chart can be used to check for the existence of and examine the operation of the central mechanism used to clean up the cache once the data is used. The database connections are critical. If a user opens a database connection to the database repository and reads data, when he or she is done the application should immediately close the connection and remove the path to the data source. The same principle applies to an object that consumes data. When the object has completed its task, it should immediately be emptied of data, set equal to null and removed from memory. By the same token, when the application stores data in a temporary file, the file should be deleted when the process is complete.

6. Malicious code or controls

One of the most dangerous threats to an application is the existence of malicious code designed to take control and provide a back door entry. Malicious code can usually be identified in a McCabe IQ chart by looking for objects outside the normal call stack that access system level binaries or components to open up a behind the scenes communications channel.

7. Communications security

The McCabe IQ chart can also be used to take a closer look at the transport layers to ensure secure data communications. Make sure that ports are opened securely, data is encrypted during transit and that the communications port is closed as soon as the data transfer is completed. Digital certification should also be used to validate that each side of the communications channel is actually communicating with the intended party.

McCabe IQ provides the right tool for validating the security of an application from an architectural perspective. Its interactive, visual environment provides a fast picture of what's going on in your architecture and it provides tools to quickly dive into the inner workings of the program, even down to the code level when needed. McCabe IQ facilitates the fast analysis of the impact of your identified potential security vulnerabilities, enabling precise targeting of your development and testing resources.