Improving Software Security by Identifying and Securing Paths Linking Attack Surfaces to Attack Targets

By Dale Brenneman,
Vice President Software Quality Solutions
McCabe Software

Introduction

Software Security Analysis (SSA) typically includes the identification of attack surfaces, entry points into the system that a malicious user can exploit by providing malformed data to trigger deviant behavior; and of attack targets, areas of the system that can cause adverse critical impact if exploited. The task of the analyst is to review these entry points and critical impact areas, and assess their correctness and robustness. The challenge is that a complex piece of code typically has a large number of potential attack surfaces and attack targets, often far more than can be thoroughly analyzed in the time available.

Fortunately, not all of the potential attack targets need to be investigated in detail, rather only those that are connected to attack surfaces. Path-based methods can be used to quickly and accurately generate mappings (“attack maps”) that identify linkages between attack surfaces and attack targets, and therefore which attack targets are at risk and which are benign. This makes it possible for security analysts to concentrate their efforts on ensuring the robustness of the at risk attack targets. The result is that they are able to substantially improve the security of the code with reduced resources.

Security analysis challenges

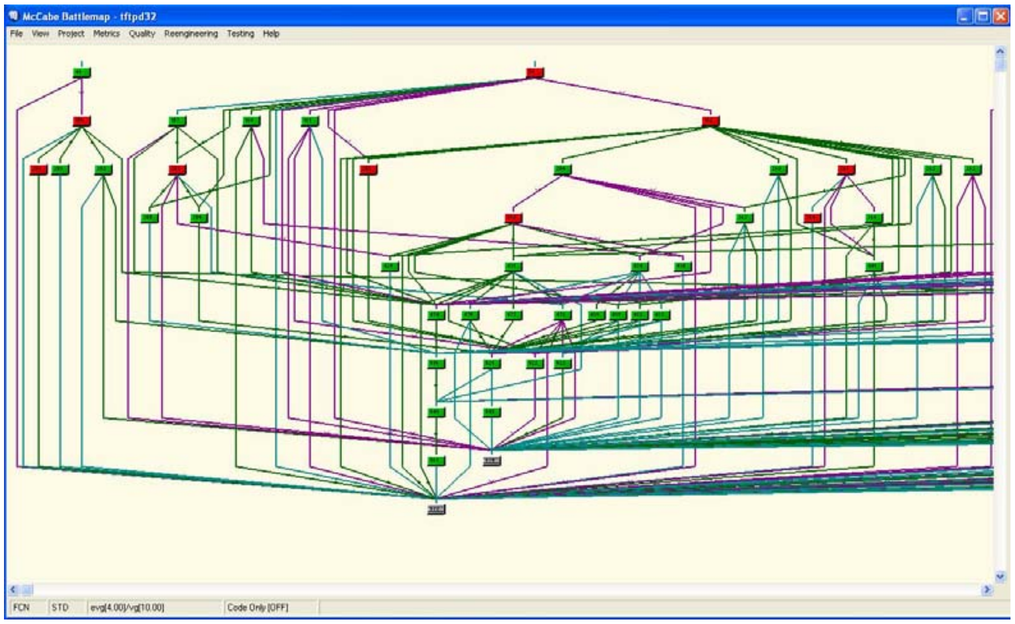
Today virtually every application connects to a network, and most to the Internet. So what used to be external interfaces are now potential points of attack. Microsoft, Carnegie Mellon University, and others pushed forward the concept of the attack surface for identifying areas of potential vulnerability. SSA provides a method of identifying the attack surfaces, which encompass code, interfaces, services and protocols.

From the standpoint of code analysis, a significant concern consists of areas where the system obtains external input, such as functions that accept data or read configuration files, environmental variables or registry entries that affect application behavior. Malicious attacks often originate from these entry points, so it is important to review them to assess their correctness and robustness. But today’s software can also have hundreds of potential attack targets inside the application, and there is rarely if ever enough time available to thoroughly check them all out. SSA typically ends when the team runs out of time, money or finds a specified number of bugs.

Path-based methods enable an approach that much more effectively leverages the available SSA resources. The attack targets are analyzed based on their connections to attack surfaces through call relationships. The value of this approach is that a large proportion of attack targets does not interact with attack surfaces, and therefore are completely benign. The identification of these trivial attack targets makes it possible for security analysts to focus their energies on the typically small proportion of attack targets that present a real danger.

Identifying the critical paths

The critical areas of code requiring analysis for potential security remediation are identified through the creation of an attack map that links attack surfaces to attack targets. The creation of the attack map begins with the identification of functions that characterize the attack surfaces and attack targets. For example, create a set called AttackSurface and add to its contents the functions from the input space that you wish to trace. In a network application, the recv() function, which receives data from a socket, may be of interest. Next, create a set called AttackTarget and specify target functions that identify critical areas of the code.


McCabe Battlemap - tftpd32
File View Project Metrics Quality Reengineering Testing Help
FCN
STD
evg[4.00]/vg[10.00]
Code Only [OFF]

Structure chart of attack map shows attack surfaces, attack targets, and functions that link them

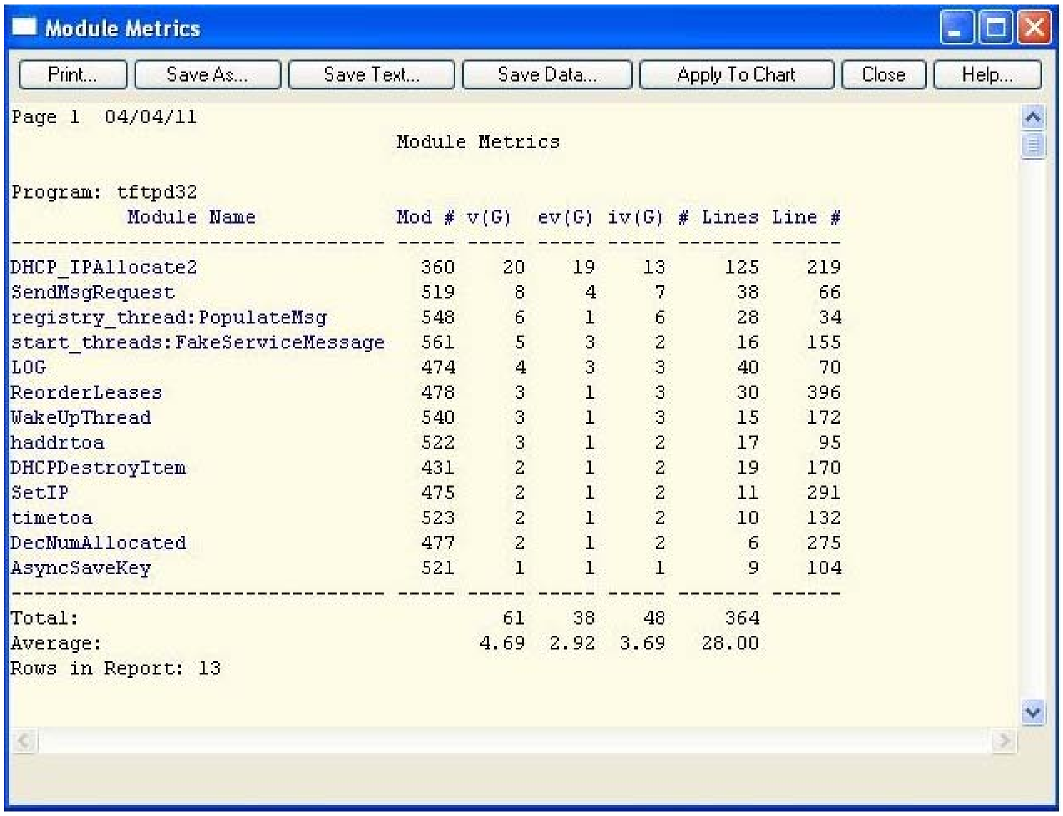
An attack map is created by requesting the tool to provide a reduced view, consisting only of functions in the attack surface set, the attack target set, and the functions that link them through call relationships. All other functions will be filtered from view. Further analysis and reporting can proceed from this focused view.

The attack map clearly identifies the functions and paths that are potentially exercised in the execution flow of an attack, and therefore should be carefully analyzed and tested. Less attention needs to be paid to functions that are filtered out. This approach can substantially improve the security of an application by enabling security analysts to focus their attention on areas of code that could potentially be utilized in an attack.

Securing the critical paths

The paths of execution control identified on the attack map are potentially critical to application security, and therefore should be carefully analyzed and tested using a path-based approach.

A program metrics report can be produced to show the program integration complexity for the subset of functions of interest. The program integration complexity represents the number of unique, linearly independent paths exercising all calls through the functions shown in the chart. This can be used to help estimate the effort of further analysis.



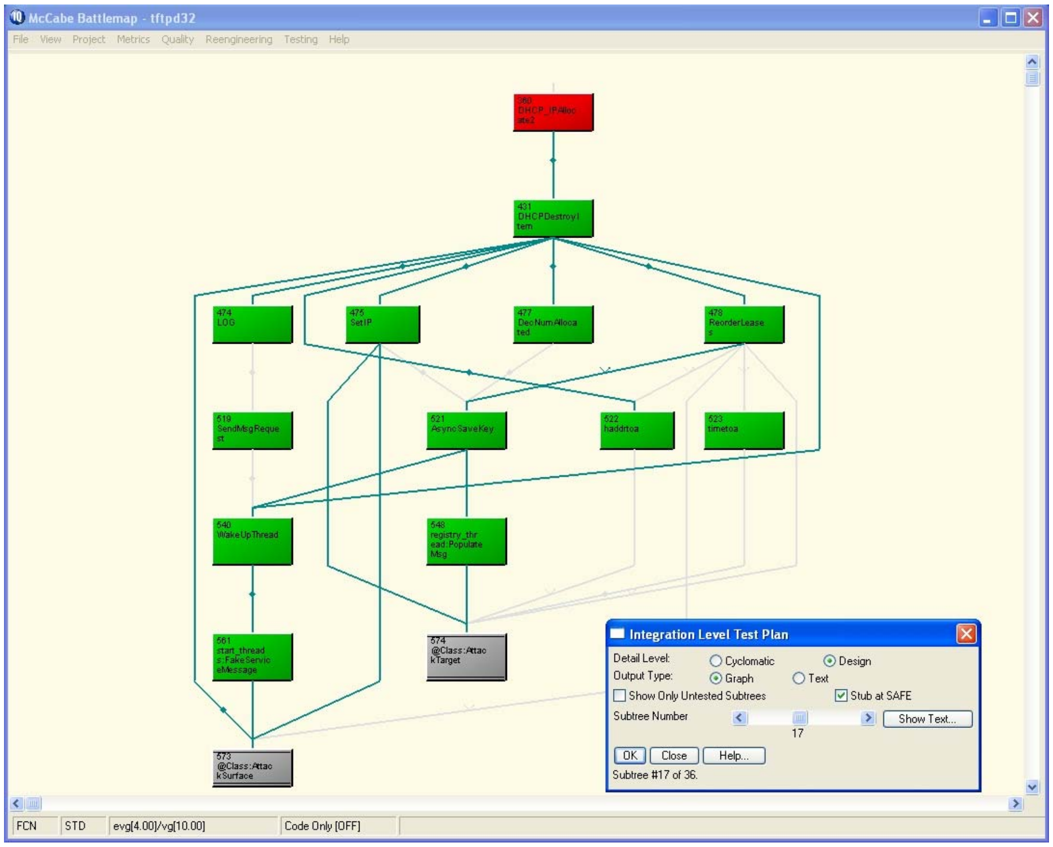
```
Page 1  04/04/11
                                 Module Metrics

Program: tftpd32
          Module Name              Mod # v(G)  ev(G) iv(G) # Lines Line #
-------------------------------- ----- ----- ----- ----- ------- ------
DHCP_IPAllocate2                   360    20    19    13     125    219
SendMsgRequest                     519     8     4     7      38     66
registry_thread:PopulateMsg        548     6     1     6      28     34
start_threads:FakeServiceMessage   561     5     3     2      16    155
LOG                                474     4     3     3      40     70
ReorderLeases                      478     3     1     3      30    396
WakeUpThread                       540     3     1     3      15    172
haddrtoa                           522     3     1     2      17     95
DHCPDestroyItem                    431     2     1     2      19    170
SetIP                              475     2     1     2      11    291
timetoa                            523     2     1     2      10    132
DecNumAllocated                    477     2     1     2       6    275
AsyncSaveKey                       521     1     1     1       9    104
-------------------------------- ----- ----- ----- ----- ------- ------
Total:                                   61    38    48     364
Average:                               4.69  2.92  3.69   28.00
Rows in Report: 13
```

Report shows functions in attack map, including their cyclomatic complexity

A metrics report helps determine which functions of an attack map or subset have a high risk due to their complexity. The report above shows that one of the functions in this rooted subset of the attack map has a cyclomatic complexity (v(G)) greater than 15. Refactoring could be used to break down complex functions in the attack map into smaller functions, many of which will likely be outside the attack path of execution, reducing the security risk.

The next step is investigating each root level function and examining the integration paths that reach the attack target. The structure chart shows a hierarchical call tree of functions. (Alternatively, a chart showing class relationships can be displayed.) To focus analysis on an individual call tree, filter the structure chart to show only the functions in a call tree rooted at the specified function.



Detail Level: [ ] Cyclomatic [✓ Design]
Output Type: [✓ Graph] [ ] Text
[ ] Show Only Untested Subtrees [✓ Stub at SAFE]
Subtree Number 17
Subtree #17 of 36.

Structure chart rooted at a function shows calls exercised for each call tree sequence

Analyze specific control flow paths by highlighting them and by displaying the sequences of decision outcomes needed to exercise them thoroughly. The structure chart can show the linearly independent integration paths through the functions in the current chart. The user can step through the various subtrees and the chart will highlight lines connecting the boxes that represent the calls exercised for the selected call tree sequence. Special attention should be paid to integration subtrees that highlight calls to the both the attack target and attack surface. The chart can also be used to generate a test report on any design subtree listing the calling and called functions and the test conditions needed to exercise the subtree.

For further code and path analysis, drill down into the details of a function by right-clicking on a box on the structure chart or on a function name in a report, to bring up a context menu for selection. Selectable details include a graphical representation of the function's control flow and the source code for the function. The user can also step through sets of integration of cyclomatic paths through the function. The flowgraph, annotated source coding listing and test path details can be used to carry out a thorough analysis of each function in the attack map to ensure the control flow paths shown are valid, consistent with requirements and secure.

Path based code coverage analysis should also be applied to all code in an attack map, to further reduce your security risk. At a minimum, testing should be required through all integration paths in an attack map; that is, with a goal of 100% integration path coverage, with exceptions only where certain path combinations cannot be attained. If resources permit, it is suggested to apply code coverage analysis at the cyclomatic path level for functions in an attack map.

Reduced security risk

The path-based methods described here can make a substantial improvement to SSA by identifying potential vulnerable code based on linkages between attack surfaces and attack targets. Path-based methods can also be used to assist in securing the critical paths by providing a methodical approach to analyzing integration paths linking attack surfaces and attack targets. The end result is an improvement in application security while working within the existing SSA resources.

For more information, contact: McCabe Software, Inc. | 41 Sharpe Drive, Cranston, Rhode Island 02920 USA | Ph: 800-638-6316 or 401-572-3100 | Fax: (401) 572-3351 | Email: info@mccabe.com | Web: http://security.mccabe.com