2011 18th Working Conference on Reverse Engineering

Function Insight

Highlighting Suspicious Sections in Binary Run Traces

Michelle Cheatham, Jason Raber
Cyber Research Lab
Riverside Research
Beavercreek, OH, USA
mcheatham, jraber@riversideresearch.org

***Abstract*— Function Insight is a tool for visualizing run traces at the functional level. A key feature is the ability to bring the user's attention to particularly important sections of code based on rule-, machine learning- and data mining-based heuristics or a user-defined "interest metric."**

Run trace analysis; profiling; visualization; differential analysis; software reverse engineering framework

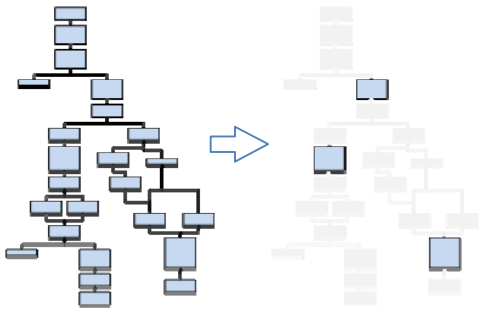
I. MOTIVATION

Traditionally, much of software reverse engineering involves sitting down with an executable for which no source code is available, identifying the original entry point, and slogging line-by-line through assembly code. In our experience analyzing malware and red teaming (assessing the strength of) application protections, insight often comes more quickly and easily by raising the level of abstraction. For instance, rather than examining straight assembly code, looking at the control flow and timing information of a protected or infected versus a clean executable at the functional level can give valuable information about the nature of the protection or malware. This becomes even more helpful if the tool can quickly direct the user's attention to particularly suspicious sections of code. It was these motivations that led us to develop Function Insight – we want to provide an easy-to-use tool that leverages rule-based, machine learning, and data mining techniques to aid non-experts in analyzing anomalous sections of executables.

II. CAPABILITIES

Function Insight shows a run trace at the functional level. It currently uses data generated by our custom profiling tool Data Code Miner [1] but could easily show information from other profiling tools, such as gprof, Detours, or WinAPIOverride. The information about each function call includes the function name, return address, parameter types and values, and other details. There can easily be hundreds of thousands or more function calls in a single run trace. Function Insight provides visualizations and commands to allow users to make sense of this data. Users can do things typical of code profiling tools, including viewing the call tree at various levels of abstraction (all the way down to the value of registers at the time of a function call, if they desire), finding all occurrences of a particular function, and considering the time spent in various functions. In addition, Function Insight has a diff mode that allows users to compare two executions side-by-side, facilitating the kind of analysis of clean versus modified code discussed above.

We think the most useful aspect of this tool is the ability to automatically highlight suspicious sections of code, which allows an analyst to quickly zero in on his target (see Figure 1). When a run trace is loaded, the user can select from a list of available heuristics. How each heuristic operates is completely unconstrained – it simply needs to produce a value between 0 and 1 in the end to indicate the level of interest of each line in the run trace. Function Insight then combines all of these values into a single interest score and shades that line in the run trace in proportion with that score. By default the values from the heuristics are simply averaged, but the user can specify whatever function they would like by writing a small piece of Java code. Based on our practical red teaming experience, we have developed a basic set of heuristics that simply flag API calls for socket communications, file access, and anti-debugging protections. We are currently beginning development of several more advanced heuristics based on machine learning and data mining techniques, one of which is discussed in Section IV.



Function Insight is part of Hydra, a software reverse engineering tool suite being developed through an internal research and development effort at Riverside Research. We hope that academic researchers will consider using Hydra as a test bed for their own heuristic algorithms and run trace visualizations. Researchers can develop their own heuristic algorithms in Java using our open API. Once the code is

1095-1350/11 $26.00 © 2011 IEEE
DOI 10.1109/WCRE.2011.63

IEEE computer society

compiled into a DLL and placed in the plugins directory, it is automatically included in the list of available heuristics that is presented to the user when they open a run trace. Similarly, users who want to develop their own visualizations for run trace information can write them in Java, package them in a jar file, and place them in the plugins directory. They will then be automatically available within the application and can make use of the heuristics and other functionality through our API.

III. Related Work

There are several existing tools similar to Function Insight. For instance, KProf [2] allows users to visualize the output from common profilers, such as gprof and Function Check, but it does not have the ability to identify or highlight interesting code segments. WinAPIOverride [3] allows users to compare two traces and view standard profiler statistics; its trace comparison system is relatively basic, however, and it does not provide any identification or highlighting features. Process Stalker [4] has the ability to generate detailed run trace graphs and automatically highlight "interesting" sections of graphs; however, it does not provide any differential analysis facilities for comparing traces and the definition of what qualifies as interesting is hard-coded. Function Insight seeks to provide an easy-to-use run trace comparison capability, along with the ability to apply built-in or user-created heuristics to indentify and highlight interesting code segments.

IV. Future Research

One of the research directions we would like to pursue based on the tool presented here is the development of more advanced plug-ins for filtering and assigning interest metric values to sections of code in binary run traces for which no source code is available. One possibility is to use Sequential Pattern Mining [5] on a run trace to determine which function calls normally lead to certain other function calls. Cases in which these patterns are violated could then be given a higher interest value to catch the analyst's attention. This technique could be applied either within a single run trace or to compare two different traces. For instance, assume we perform sequential pattern mining on a run trace of an executable that is known to be clean. We could then automatically flag areas of run traces from binaries suspected to be compromised where the discovered patterns are violated. The higher the support and confidence levels of the violated rule, the more suspicious it is when the rule is broken. Another potential research thread is to improve the way comparisons are done between run traces. Current approaches are based on simple equality between trace components (function names/addresses, parameter values, register values, etc.), but we have found that these lead to many false positives because of normal variations between execution runs. We would like to develop alternative similarity metrics that avoid distracting the user with these inconsequential fluctuations. Both of these ideas could be evaluated within our existing Function Insight framework.

V. Summary

Function Insight makes the following contributions to the field of software reverse engineering:

- Provides a polished, easy-to-use, and easily understandable tool to visualize run traces, including a diff mode for comparing two different traces
- Quickly draws the user's attention by highlighting important sections of code based on included or user-defined filters and interest metric value
- Facilitates research and development of more advanced heuristics and interest metric assignments by serving as a test bed for such algorithms

References

[1] J. Raber and E. Laspe, "Emulated breakpoint debugger and data mining using Detours," 14th Working Conference on Reverse Engineering (WCRE), pp.271-272 , 2007.

[2] KProf online documentation, http://kprof.sourceforge.net/.

[3] WinAPIOverride online documentation, http://jacquelin.potier.free.fr/winapioverride32/documentation.php.

[4] Process Stalker online documentation, http://pedram.redhive.com/process_stalking_manual/.

[5] B. Goethals, "Survey on frequent pattern mining," Technical report, Helsinki Institute for Information Technology, 2003