

CLAIMS

1. A method for detecting high-level functionality of an application executing on a computing device, the method comprising:

storing, in a secure memory on a computing device, an application-specific virtual address mapping table for an application, the application-specific virtual address mapping table comprising a plurality of virtual address offsets in application binary code mapped to corresponding target application functionalities in source code of the application;

in response to launching the application, generating a process-specific virtual address mapping table for an instance of an application process to be executed, the process-specific virtual address mapping table defining actual virtual addresses corresponding to the target application functionalities using the virtual address offsets in the application-specific virtual address mapping table;

during execution of the application binary code for the instance of the application process, detecting one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the process-specific virtual address mapping table; and

providing the executed target application functionalities as detected from the actual virtual addresses in the process-specific virtual address mapping table into an exception handling module, the exception handling module configured to detect one or more exceptions or behaviors associated with execution of the application binary code for the instance of the application process, wherein the exception handling module comprises a malicious code detection algorithm.

2. The method of claim 1, further comprising:

generating another process-specific virtual address mapping table for another instance of the application process to be simultaneously executed with the other instance of the application process;

during execution of the application binary code for the another instance of the application process, detecting when one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the another process-specific virtual address mapping table.

3. The method of claim 1, wherein the actual virtual addresses are determined using a base virtual address for the instance of the application process and the virtual address offsets in the application-specific virtual address mapping table.

4. The method of claim 1, wherein the secure memory resides in a trusted zone in a high-level operating system (HLOS).

5. The method of claim 1, wherein the application comprises one of a secure web application and a web browser.

6. The method of claim 1, wherein the application binary code is executed as native binary code.

7. A system for detecting high-level functionality of an application executing on a computing device, the system comprising:

means for storing on a computing device an application-specific virtual address mapping table for an application, the application-specific virtual address mapping table comprising a plurality of virtual address offsets in application binary code mapped to corresponding target application functionalities in source code of the application;

means for generating, in response to launching the application, a process-specific virtual address mapping table for an instance of an application process to be executed, the process-specific virtual address mapping table defining actual virtual addresses corresponding to the target application functionalities using the virtual address offsets in the application-specific virtual address mapping table;

means for detecting, during execution of the application binary code for the instance of the application process, one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the process-specific virtual address mapping table; and

means for providing the executed target application functionalities as detected from the actual virtual addresses in the process-specific virtual address mapping table into an exception handling module, the exception handling module configured to detect one or more exceptions or behaviors associated with execution of the application binary code for the instance of the application process, wherein the exception handling module comprises a malicious code

detection algorithm.

8. The system of claim 7, further comprising:

means for generating another process-specific virtual address mapping table for another instance of the application process to be simultaneously executed with the other instance of the application process;

means for detecting, during execution of the application binary code for the another instance of the application process, when one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the another process-specific virtual address mapping table.

9. The system of claim 7, wherein the actual virtual addresses are determined using a base virtual address for the instance of the application process and the virtual address offsets in the application-specific virtual address mapping table.

10. The system of claim 7, wherein the means for storing resides in a trusted zone in a high-level operating system (HLOS).

11. The system of claim 7, wherein the application comprises one of a secure web application and a web browser.

12. The system of claim 7, wherein the application binary code is executed as native binary code.

13. A computer program embodied in a memory and comprising a non-transitory computer-readable medium having computer-readable program code embodied therein executable by a processor for detecting high-level functionality of an application executing on a computing device, the computer program comprising logic configured to:

store, in a secure memory on a computing device, an application-specific virtual address mapping table for an application, the application-specific virtual address mapping table comprising a plurality of virtual address offsets in application binary code mapped to corresponding target application functionalities in source code of the application;

generate, in response to launching the application, a process-specific virtual address mapping table for an instance of an application process to be executed, the process-specific

virtual address mapping table defining actual virtual addresses corresponding to the target application functionalities using the virtual address offsets in the application-specific virtual address mapping table;

detect, during execution of the application binary code for the instance of the application process, one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the process-specific virtual address mapping table; and

provide the executed target application functionalities as detected from the actual virtual addresses in the process-specific virtual address mapping table into an exception handling module, the exception handling module configured to detect one or more exceptions or behaviors associated with execution of the application binary code for the instance of the application process, wherein the exception handling module comprises a malicious code detection algorithm.

14. The computer program of claim 13, further comprising logic configured to:

generate another process-specific virtual address mapping table for another instance of the application process to be simultaneously executed with the other instance of the application process;

during simultaneous execution of the application binary code for both instances of the application process, detect when one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the another process-specific virtual address mapping table.

15. The computer program of claim 13, wherein the actual virtual addresses are determined using a base virtual address for the instance of the application process and the virtual address offsets in the application-specific virtual address mapping table.

16. The computer program of claim 13, wherein the secure memory resides in a trusted zone in a high-level operating system (HLOS).

17. The computer program of claim 13, wherein the application comprises one of a secure web application and a web browser.

18. The computer program of claim 13, wherein the application binary code is executed as native binary code.

19. A system for detecting high-level functionality of an executing application, the system comprising:

- a processing device configured to execute application binary code; and

- a high-level operating system (HLOS) comprising:

 - an application-specific virtual address mapping table comprising a plurality of virtual address offsets in application binary code mapped to corresponding target application functionalities in source code of the application; and

 - a kernel module configured to generate, in response to launching the application, a process-specific virtual address mapping table for an instance of an application process to be executed, the process-specific virtual address mapping table defining actual virtual addresses corresponding to the target application functionalities using the virtual address offsets in the application-specific virtual address mapping table;

 - the HLOS configured to detect, during execution of the application binary code for the instance of the application process, one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the process-specific virtual address mapping table; and

 - the HLOS providing the executed target application functionalities as detected from the actual virtual addresses in the process-specific virtual address mapping table into an exception handling module, the exception handling module configured to detect one or more exceptions or behaviors associated with execution of the application binary code for the instance of the application process, wherein the exception handling module comprises a malicious code detection algorithm.

20. The system of claim 19, wherein the HLOS is further configured to:

- generate another process-specific virtual address mapping table for another instance of the application process to be simultaneously executed with the other instance of the application process; and

- detect, during simultaneous execution of the application binary code for both instances of the application process, when one or more of the actual virtual addresses corresponding to the target application functionalities are executed based on the another process-specific virtual address mapping table.

21. The system of claim 20, wherein the actual virtual addresses are determined using a base virtual address for the instance of the application process and the virtual address offsets in the application-specific virtual address mapping table.

22. The system of claim 20, wherein the application-specific virtual address mapping table is stored in a trusted zone in the HLOS.