

WHAT IS CLAIMED IS:

1. A computer-implemented method for detection of malicious code within runtime generated code executing within a computer, comprising executing on a processor of the computer the acts of:

receiving an indication of at least one of the creation and the execution of runtime generated code in a memory of a computer;

identifying a match between signature data associated with the runtime generated code and a template signature of a plurality of templates representing authorized source creation modules that created the runtime generated code, the templates stored in a repository on a storage device,

wherein the signature data includes at least one of: a predefined size of an area in a physical or virtual memory storing the runtime generated code, and at least one predefined code pattern within the runtime generated code; and

triggering a security process to handle malicious code in the runtime generated code when no match is found.

2. The method of claim 1, wherein the template signature represents an authorized just in time (JIT) compiler.

3. The method of claim 2, wherein identifying the match between the signature data and the template signature comprises at least one of:

identifying an association between a first executable module called by the runtime generated code to invoke an operating system function, and the template representing the authorized JIT compiler, and

identifying an association between a second executable module creating the runtime generated code and the template representing the authorized JIT compiler.

4. The method of claim 2, wherein the signature data comprises a designation of a memory region storing the runtime generated code as read-only or no-access.

5. The method of claim 1, wherein the at least one code pattern includes at least one member selected from the group consisting of: at least one predefined prolog at a start region of at least one function of the runtime generated code, at least one epilogue, and at least one magic operand value.

6. The method of claim 2, wherein the signature data comprises predefined control structures related to the JIT compiler at least one of at a start region and an end region of the runtime generated code.

7. The method of claim 6, wherein the predefined control structures include at least one of: a linked list at each of a plurality of different memory regions each storing a portion of the runtime generated code, and fields defining size and address of the respective memory region located after the respective linked list.

8. The method of claim 7, wherein the linked list is verified by traversing pointers of each memory region, and the fields are verified by correlating the values of the fields with operating system values.

9. The method of claim 2, wherein the signature data comprises an application associated with the runtime generated code to which the authorized JIT compiler is restricted.

10. The method of claim 1, wherein the template signature represents an authorized hook engine.

11. The method of claim 10, wherein the signature data includes identification that the runtime generated code is created by a hook engine, the identifying performed by at least one of:

emulating preexisting code at the prolog of a hooked module to reach outside code residing outside of the hooked module; and

analyzing a stack trace related to the outside code to identify the runtime generated code by locating the position of the runtime generated code as appearing in the stack trace before the authorized hook engine executable that installed the hook.

12. The method of claim 10, wherein the signature data includes at least one member selected from the group consisting of: predefined control structures at least at one of at a start portion and an end portion of the runtime generated code memory region, and an opcode signature calculated from assembly obtained by applying a disassemble program to the runtime generated code excluding mutable parameters.

13. The method of claim 12, wherein the at least one code pattern includes at least one member selected from the group consisting of: at least one predefined prolog at a start region of at least one function of the runtime generated code, at least one epilogue, and at least one magic operand value.

14. The method of claim 1, wherein the template signature represents an authorized executable compressor, wherein the runtime generated code comprises a decompressed program.

15. The method of claim 14, wherein the signature data further includes at least one member selected from the group consisting of: a cryptographic hash function calculated over immutable portions of the executable file structure and code, and permissions on memory pages where the decompressed executable file resides.

16. The method of claim 15, further comprising verifying that contents of the memory at the base of the memory allocation is according to the format of the decompressed executable file by parsing contents of the memory allocation according to the format of the decompressed executable file, and checking that field values are logical and conform to the format.

17. A system for detection of runtime generated code containing malicious code, comprising:

a memory for storing code;

a storage device for storing a repository of templates representing authorized; source creation modules that create runtime generated code;

a program store storing code; and

a processor coupled to the memory, the storage device, and the program store for implementing the stored code, the stored code comprising:

stored code to receive an indication of at least one of the creation and the execution of runtime generated code in the memory, identify a match between signature data associated with the runtime generated code and a template signature of the repository, wherein the signature data includes at least one of: a predefined size of an area in a physical or virtual memory storing the runtime generated code, and at least one predefined code pattern within the runtime generated code; and trigger a security process to handle malicious code in the runtime generated code when no match is found.

18. A computer program product comprising a non-transitory computer readable storage medium storing program code thereon for implementation by a processor of a system for detection of runtime generated code containing malicious code, the program code comprising:

instructions to receive an indication of at least one of the creation and the execution of runtime generated code in a memory of a computer;

instructions to identify a match between signature data associated with the runtime generated code and a template signature of a set of templates representing authorized source creation modules that create runtime generated code,

wherein the signature data includes at least one of: a predefined size of an area in a physical or virtual memory storing the runtime generated code, and at least one predefined code pattern within the runtime generated code; and

instructions to trigger a security process to handle malicious code in the runtime generated code when no match is found.