WHAT IS CLAIMED IS:

1. A method comprising:

generating output code based on input code using a compiler or translator;

providing the output code to one or more platforms for execution;

receiving feedback associated with the execution of the output code, the feedback identifying at least one of: one or more failures during the execution of the output code and one or more performance characteristics of the execution of the output code;

modifying the compiler or translator based on the feedback; and

generating additional output code using the modified compiler or translator, wherein generating the additional output code comprises using binary diversity to randomize an amount of stack space allocated to each of one or more functions associated with the additional output code.

2. The method of Claim 1, wherein modifying the compiler or translator comprises modifying the compiler or translator to at least one of:

reduce or eliminate the one or more failures during execution of the additional output code; and

improve the one or more performance characteristics of the execution of the additional output code relative to the execution of the output code.

3. The method of Claim 1, wherein the feedback comprises metrics associated with memory stack allocations, control flow graphs, and memory heap interactions during the execution of the output code.

4. The method of Claim 1, further comprising:

receiving information identifying one or more cyber-security vulnerabilities; and

modifying the compiler or translator based on the information;

wherein modifying the compiler or translator comprises modifying the compiler or translator to partially or completely avoid creation of the one or more vulnerabilities in the additional output code.

5. The method of Claim 1, further comprising:

receiving information identifying one or more portions of the input code or additional input code to be converted using the compiler or translator;

wherein modifying the compiler or translator causes the compiler or translator to only convert the one or more portions of the input code or the additional input code.

6. The method of Claim 1, wherein generating the additional output code further comprises using the binary diversity to at least one of:

randomize an order of at least some functional code blocks associated with the additional output code;

randomize an order of data used by the additional output code;

randomize heap memory allocations associated with the additional output code; and

insert instructions into the additional output code that do not alter results generated by the additional output code.

7. The method of Claim 1, wherein the compiler or translator comprises a WebAssembly compiler or translator.

8. The method of Claim 1, wherein modifying the compiler or translator based on the feedback comprises performing, using a machine learning model and based on telemetry identifying at least one of: the one or more failures and the one or more performance characteristics, multiple types of modifications to the compiler or translator.

9. An apparatus comprising:

at least one processor configured to:

generate output code based on input code using a compiler or translator;

provide the output code to one or more platforms for execution;

receive feedback associated with the execution of the output code, the feedback identifying at least one of: one or more failures during the execution of the output code and one or more performance characteristics of the execution of the output code;

modify the compiler or translator based on the feedback; and

generate additional output code using the modified compiler or translator,

wherein the at least one processor is configured to use binary diversity to randomize an amount of stack space allocated to each of one or more functions associated with the additional output code.

10. The apparatus of Claim 9, wherein the at least one processor is configured to modify the compiler or translator to at least one of:

reduce or eliminate the one or more failures during execution of the additional output code; and

improve the one or more performance characteristics of the execution of the additional output code relative to the execution of the output code.

11. The apparatus of Claim 9, wherein the feedback comprises metrics associated with memory stack allocations, control flow graphs, and memory heap interactions during the execution of the output code.

12. The apparatus of Claim 9, wherein the at least one processor is further configured to:

receive information identifying one or more cyber-security vulnerabilities; and

modify the compiler or translator based on the information to partially or completely avoid creation of the one or more vulnerabilities in the additional output code.

13. The apparatus of Claim 9, wherein:

the at least one processor is further configured to receive information identifying one or more portions of the input code or additional input code to be converted using the compiler or translator; and

the at least one processor is configured to modify the compiler or translator to only convert the one or more portions of the input code or the additional input code.

14. The apparatus of Claim 9, wherein the at least one processor is further configured to use the binary diversity to at least one of:

randomize an order of at least some functional code blocks associated with the additional output code;

randomize an order of data used by the additional output code;

randomize heap memory allocations associated with the additional output code; and

insert instructions into the additional output code that do not alter results generated by the additional output code.

15. The apparatus of Claim 9, wherein the compiler or translator comprises a WebAssembly compiler or translator.

16. The apparatus of Claim 9, wherein, to modify the compiler or translator based on the feedback, the at least one processor is configured to perform, using a machine learning model and based on telemetry identifying at least one of: the one or more failures and the one or more performance characteristics, multiple types of modifications to the compiler or translator.

17. A non-transitory computer readable medium containing instructions that when executed cause at least one processor to:

generate output code based on input code using a compiler or translator;

provide the output code to one or more platforms for execution;

receive feedback associated with the execution of the output code, the feedback identifying at least one of: one or more failures during the execution of the output code and one or more performance characteristics of the execution of the output code;

modify the compiler or translator based on the feedback; and

generate additional output code using the modified compiler or translator,

wherein the instructions that when executed cause the at least one processor to generate the additional output code comprise instructions that when executed cause the at least one processor to use binary diversity to randomize an amount of stack space allocated to each of one or more functions associated with the additional output code.

18. The non-transitory computer readable medium of Claim 17, wherein the instructions that when executed cause the at least one processor to modify the compiler or translator comprise instructions that when executed cause the at least one processor modify the compiler or translator to at least one of:

reduce or eliminate the one or more failures during execution of the additional output code; and

improve the one or more performance characteristics of the execution of the additional output code relative to the execution of the output code.

19. The non-transitory computer readable medium of Claim 17, wherein:

the instructions when executed further cause the at least one processor to receive information identifying one or more cyber-security vulnerabilities; and

the instructions that when executed cause the at least one processor to modify the compiler or translator comprise instructions that when executed cause the at least one processor modify the compiler or translator to partially or completely avoid creation of the one or more vulnerabilities in the additional output code.

20. The non-transitory computer readable medium of Claim 17, wherein:

the instructions when executed further cause the at least one processor to receive information identifying one or more portions of the input code or additional input code to be converted using the compiler or translator; and

the instructions that when executed cause the at least one processor to modify the compiler or translator comprise instructions that when executed cause the at least one processor modify the compiler or translator to only convert the one or more portions of the input code or the additional input code.

21. The non-transitory computer readable medium of Claim 17, wherein the instructions that when executed cause the at least one processor to generate the additional output code comprise instructions that when executed cause the at least one processor to use the binary diversity to at least one of:

randomize an order of at least some functional code blocks associated with the additional output code;

randomize an order of data used by the additional output code;

randomize heap memory allocations associated with the additional output code; and

insert instructions into the additional output code that do not alter results generated by the additional output code.

22. The non-transitory computer readable medium of Claim 17, wherein the compiler or translator comprises a WebAssembly compiler or translator.

23. The non-transitory computer readable medium of Claim 17, wherein the instructions that when executed cause the at least one processor to modify the compiler or translator based on the feedback comprise instructions that when executed cause the at least

one processor to perform, using a machine learning model and based on telemetry identifying at least one of: the one or more failures and the one or more performance characteristics, multiple types of modifications to the compiler or translator.