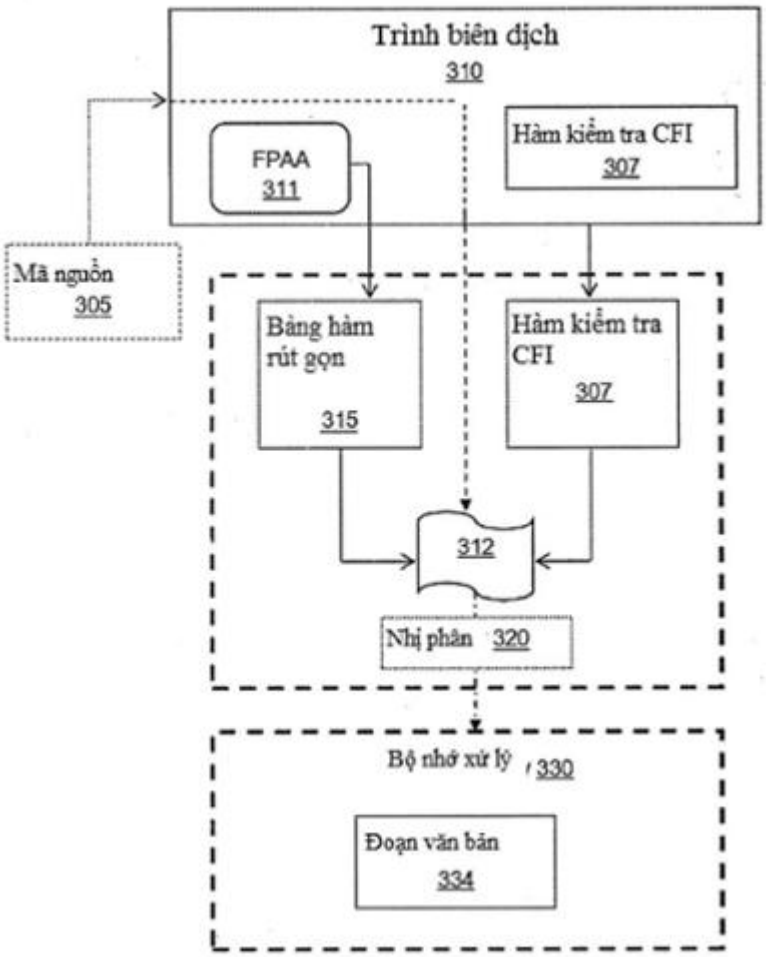
(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ
(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ
(11) **1-0037857**
(51)[19] **G06F 21/54**; G06F 21/51
(13) **B**

(21) 1-2019-05060
(22) 26/02/2018
(86) PCT/SG2018/050086 26/02/2018
(87) WO2018/156085 30/08/2018
(30) 10201701541S 27/02/2017 SG
(45) 25/12/2023 429
(43) 30/01/2020 382A
(73) HUAWEI INTERNATIONAL PTE. LTD. (SG)
51, Changi Business Park Central 2, #07-08, The Signature, Singapore 486066, Singapore
(72) DAI, Ting (CN); WU, Yongzheng (CN).
(74) Công ty Luật TNHH Phạm và Liên danh (PHAM & ASSOCIATES)

(54) THIẾT BỊ VÀ PHƯƠNG PHÁP CỦNG CỐ TÍNH TOÀN VẸN LUỒNG ĐIỀU KHIỂN CỦA ỨNG DỤNG PHẦN MỀM

(57) Sáng chế mô tả thiết bị và phương pháp để thiết bị củng cố tính toàn vẹn luồng điều khiển của ứng dụng phần mềm làm ứng dụng được thực thi trên thiết bị.



Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến thiết bị và phương pháp cho thiết bị củng cố tính toàn vẹn luồng điều khiển của ứng dụng phần mềm làm ứng dụng được thực thi trên thiết bị.

Tình trạng kỹ thuật của sáng chế

Phần lớn ứng dụng phần mềm dễ gặp phải các cuộc tấn công nguy hiểm nhằm cố gắng thay đổi hành vi của ứng dụng phần mềm bằng cách thay đổi luồng điều khiển của ứng dụng. Thông thường, các cuộc tấn công này xuất hiện khi ứng dụng phần mềm nhận và thực thi mã nhị phân độc hại được nhận trên các kênh truyền thông thông thường. Khi được thực thi, thì các mã nhị phân độc hại thay đổi luồng điều khiển của ứng dụng bằng cách khai thác các lỗ hổng phần mềm trước đó được chứa trong ứng dụng. Thông qua việc khai thác các lỗ hổng này, các kẻ tấn công cố ý sau đó có thể phá hoại ứng dụng. Khi ứng dụng bị phá hoại, thì các kẻ tấn công nguy hiểm này sẽ chiếm điều khiển trên hành vi của các ứng dụng phần mềm mà người dùng ban đầu không biết.

Cụ thể là, các loại sâu hoặc vi rút có thể tự chèn vào luồng thực thi của ứng dụng phần mềm. Khi điều này xuất hiện, thì luồng từ tiến trình được lên kế hoạch bằng phần mềm có thể được chệch hướng sang tiến trình được ra lệnh bởi sâu hoặc vi rút. Nếu ứng dụng phần mềm có lỗ hổng ở mức mã máy, tấn công bởi sâu hoặc vi rút có thể gây chệch hướng thực thi mã máy phần mềm, dẫn đến thực thi chuỗi lệnh mã máy không ngừa trước.

Các chuyên gia trong lĩnh vực đã cố gắng giải quyết vấn đề này bằng cách giảm hoặc nỗ lực giảm các nhiều lỗi hoặc lỗ hổng trong mã hóa

phần mềm nhất có thể. Bất lợi của cách tiếp cận này là tất cả các lỗi trong một đoạn phần mềm khó bị triệt tiêu hoàn toàn. Các cách di trú khác tùy thuộc vào các hệ thống máy tính chọn độc lập các số nguyên ngẫu nhiên từ tập lớn, và cố gắng bao quanh thông tin quan trọng hoặc dữ liệu luồng (mà kẻ tấn công có thể muốn chỉnh sửa) với các giá trị ngẫu nhiên đó — sao cho kẻ tấn công, đối với cuộc tấn công thành công, tận dụng các giá trị thực được chọn. Các cách này không thể đảm bảo rằng kẻ tấn công sẽ không nhận biết các giá trị được chọn, và có thể không hiệu quả nếu kẻ tấn công nhận thức hoặc đoán được giá trị được chọn. Rộng hơn, các cách này sẽ không thể ngăn được các cuộc tấn công khiến bộ nhớ của thiết bị bị thay đổi.

Do vậy, các giải pháp hiện tại vẫn khiến ứng dụng phần mềm dễ bị tấn công từ các vi rút, sâu, và các cuộc tấn công khác không phụ thuộc vào tính bí mật của các giá trị được chọn tùy ý mà có thể chỉnh sửa bộ nhớ của thiết bị.

Lập trình hướng trả về (Return orient programming, ROP) là phương pháp tấn công cao cấp mà khai thác các lỗ hổng chương trình. Phương pháp ROP tận dụng các "gadget" (tiện ích), vốn là tập lệnh, để thực thi được mã tùy ý trên chương trình đích. Phương pháp tấn công cải tiến khác mà chuyên gia trong lĩnh vực đã biết là tấn công "trả về libc" (ret2libc). Theo phương pháp tấn công ret2libc, kẻ tấn công định hướng lại lời gọi hàm trong ứng dụng phần mềm đích về hàm không mong muốn khác. Các loại tấn công này thay đổi luồng điều khiển bình thường của việc thực thi chương trình.

Một trong các phương pháp tấn công truyền luồng điều khiển phổ biến nhất mà các kẻ tấn công sử dụng là tấn công lời gọi hàm gián tiếp trong mã nguồn. Đây là do kẻ tấn công có thể định hướng lại lời gọi hàm gián tiếp đến mã tùy ý để khởi tạo cuộc tấn công ROP hoặc tấn công ret2libc.

Để ngăn ngừa điều này xảy ra, các phương pháp tính toàn vẹn luồng điều khiển (CFI) dựa trên trình biên dịch hiện tại tập trung vào việc đảm bảo rằng các phiên truyền luồng điều khiển diễn ra ở lời gọi hàm gián tiếp chỉ xảy ra đối với các mục tiêu hàm hợp pháp. Ở các phương pháp này, thậm chí nếu kẻ tấn công phải thay đổi luồng điều khiển; lời gọi hàm gián tiếp có thể chỉ gọi các mục đích hợp pháp này, thay vì mã tùy ý trong chương trình. Do vậy, điều này cho phép các cuộc tấn công được di trú hoặc thậm chí được ngăn ngừa.

Phương pháp được đề xuất bởi các chuyên gia trong lĩnh vực để củng cố CFI của các chương trình bao gồm việc đưa vào phần bảo vệ luồng điều khiển vào trong mã nhị phân hoặc nguồn của chương trình. Phần bảo vệ luồng điều khiển chèn kiểm tra trước lời gọi hàm gián tiếp. Phép kiểm tra đảm bảo rằng đích của lời gọi hàm gián tiếp là mục của hàm, mà hiện được nạp vào không gian bộ nhớ của tiến trình. Trong phương pháp này, các phần bảo vệ luồng điều khiển được biểu diễn bằng cấu trúc ánh xạ bit trong bộ nhớ và cấu trúc ánh xạ bit chứa tất cả các mục hàm. Mặt trái đối với phương pháp là việc cấu trúc bảo vệ luồng điều khiển không đủ an toàn do giả thiết rằng các mục tiêu hợp pháp bao gồm tất cả các hàm được nạp vào bộ nhớ. Bằng cách khai triển yếu điểm này, kẻ tấn công có thể vẫn tận dụng tất cả các hàm trong quá trình để mở cuộc tấn công.

Đối với các nguyên nhân nêu trên, các chuyên gia trong lĩnh vực liên tục cố gắng phát minh ra thiết bị và phương pháp củng cố CFI của các ứng dụng phần mềm.

Bản chất kỹ thuật của sáng chế

Sáng chế đề xuất các thiết bị và các phương pháp cải tiến việc củng cố CFI của các ứng dụng phần mềm dưới đây theo các phương án thực hiện sáng chế.

Cải tiến thứ nhất theo các phương án thực hiện của các thiết bị và các phương pháp theo sáng chế là việc các phép kiểm tra CFI được thực hiện theo sáng chế được tạo chi tiết mịn và chỉ được áp dụng cho các mục tiêu hợp pháp của các phiên truyền luồng điều khiển. Ngoài ra, do số lượng các mục tiêu hợp pháp cho mỗi lời gọi hàm được giảm, điều này nghĩa là các kẻ tấn công nguy hiểm sẽ có ít hàm hơn để khai thác.

Cải tiến thứ hai theo các phương án thực hiện của các thiết bị và các phương pháp theo sáng chế là việc các củng có CFI được đề xuất các phương án thực hiện sáng chế có các chi phí bổ sung CPU nhỏ nhất và kết quả là, có tác động nhỏ nhất lên toàn bộ hiệu năng của CPU.

Cải tiến thứ ba theo các phương án thực hiện của các thiết bị và các phương pháp theo sáng chế là việc sáng chế không đòi hỏi các thay đổi cực đoan được thực hiện cho mã nguồn ban đầu và cho mã nhị phân biên dịch.

Các cải tiến nêu trên được đề xuất bởi các phương án thực hiện thiết bị theo sáng chế hoạt động theo cách thức sau.

Theo khía cạnh thứ nhất của sáng chế, thiết bị củng cố CFI của ứng dụng phần mềm được bộc lộ, thiết bị bao gồm bộ xử lý; và phương tiện lưu trữ bất biến đọc được bởi bộ xử lý, phương tiện lưu trữ bất biến lưu trữ các lệnh mà khi được thực thi bởi bộ xử lý, khiến bộ xử lý: tiếp nhận tệp tin mã nguồn thứ nhất cho ứng dụng phần mềm nhờ đó tệp tin mã nguồn thứ nhất bao gồm các hàm và các lời gọi hàm gián tiếp; biên dịch tệp tin mã nguồn thứ nhất đẻ tạo tệp tin nhị phân thứ nhất nhờ đó cho mỗi lời gọi hàm gián tiếp được biên dịch, liên kết hàm kiểm tra CFI với mỗi lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân thứ nhất sao cho hàm kiểm tra CFI liên kết sẽ được thực thi trước khi thực thi mỗi lời gọi hàm gián tiếp được biên dịch; và thêm hàm, được đề cập đến trong mỗi lời gọi hàm gián tiếp được biên dịch, vào bảng hàm rút gọn.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm các lệnh để ra lệnh bộ xử lý: thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; và thực thi lời gọi hàm gián tiếp liên kết khi xác định được rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm: các lệnh để ra lệnh bộ xử lý: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy có hàm được đề cập trong câu gọi hàm ngoài gián tiếp khi thực thi gán lời gọi hàm ngoài gián tiếp trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; và thực thi lời gọi hàm gián tiếp liên kết khi xác định được rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm: các lệnh để ra lệnh bộ xử lý: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy có hàm được đề cập trong câu gọi hàm ngoài gián tiếp khi thực thi gán lời gọi hàm ngoài gián tiếp trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong

bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, và khi tệp tin nhị phân thứ hai được xác định đã được biên dịch theo cách hợp pháp.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm các lệnh để ra lệnh bộ xử lý: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy có hàm được đề cập trong câu gọi hàm ngoài gián tiếp khi thực thi gán lời gọi hàm ngoài gián tiếp trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, khi tệp tin nhị phân thứ hai được xác định đã không được biên dịch theo cách hợp pháp, và khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn được liên kết với tệp tin nhị phân thứ hai.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm các lệnh để ra lệnh bộ xử lý: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy có hàm được đề cập trong câu gọi hàm

ngoài gián tiếp khi thực thi gán lời gọi hàm ngoài gián tiếp trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, khi tệp tin nhị phân thứ hai được xác định đã không được biên dịch theo cách hợp pháp, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn được liên kết với tệp tin nhị phân thứ hai, và khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, thiết bị còn bao gồm các lệnh để ra lệnh bộ xử lý: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy có hàm được đề cập trong câu gọi hàm ngoài gián tiếp khi thực thi gán lời gọi hàm ngoài gián tiếp trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân được biên dịch, nhờ đó đối với mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ

chứa trong bản đồ nhị phân, và khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, bảng hàm thời gian chạy được lưu trữ trong phương tiện lưu trữ bất biến được bảo vệ đọc được bởi bộ xử lý.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, bản đồ nhị phân được lưu trữ trong phương tiện lưu trữ bất biến được bảo vệ đọc được bởi bộ xử lý.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế các lời gọi hàm gián tiếp bao gồm các lời gọi hàm gián tiếp được gán như là một phần của cấu trúc dữ liệu.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, các lời gọi hàm gián tiếp còn bao gồm các lời gọi hàm gián tiếp được gán cho các hàm.

Dựa vào khía cạnh thứ nhất, theo các phương án thực hiện sáng chế, các lời gọi hàm gián tiếp còn bao gồm các lời gọi hàm gián tiếp được gán như là các lời gọi khối.

Theo khía cạnh thứ hai của sáng chế, phương pháp củng cố CFI của ứng dụng phần mềm bằng cách sử dụng thiết bị tính toán được bộc lộ, phương pháp bao gồm các bước: nhận tệp tin mã nguồn thứ nhất cho ứng dụng phần mềm nhờ đó tệp tin mã nguồn thứ nhất bao gồm các hàm và các lời gọi hàm gián tiếp; nhận tệp tin nhị phân thứ nhất mà được tạo từ việc biên dịch mã nguồn thứ nhất và lời gọi hàm gián tiếp được biên dịch, liên kết hàm kiểm tra CFI với mỗi lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân thứ nhất sao cho hàm kiểm tra CFI liên kết sẽ được thực thi trước khi thực thi mỗi lời gọi hàm gián tiếp được biên dịch; và thêm hàm, được đề cập đến trong mỗi lời gọi hàm gián tiếp được biên dịch, vào bảng hàm rút gọn.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; và thực thi lời gọi hàm gián tiếp liên kết khi xác định được rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập đến trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy với hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp khi việc gán lời gọi hàm ngoài gián tiếp được thực thi trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; và thực thi lời gọi hàm gián tiếp liên kết khi xác định được rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập đến trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy với hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp khi việc gán lời gọi hàm ngoài gián tiếp được thực thi trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên

kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, và khi tệp tin nhị phân thứ hai được xác định đã được biên dịch theo cách hợp pháp.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập đến trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy với hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp khi việc gán lời gọi hàm ngoài gián tiếp được thực thi trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, khi tệp tin nhị phân thứ hai được xác định đã không được biên dịch theo cách hợp pháp, và khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn được liên kết với tệp tin nhị phân thứ hai.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập đến trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy với hàm được đề cập đến trong lời gọi hàm

ngoài gián tiếp khi việc gán lời gọi hàm ngoài gián tiếp được thực thi trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, khi tệp tin nhị phân thứ hai được xác định đã không được biên dịch theo cách hợp pháp, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn được liên kết với tệp tin nhị phân thứ hai, và khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, phương pháp còn bao gồm các bước: nạp tệp tin nhị phân thứ nhất để thực thi; cập nhật bản đồ nhị phân có địa chỉ được đề cập đến trong lệnh nạp khi lệnh nạp trong tệp tin nhị phân thứ nhất được thực thi; cập nhật bảng hàm thời gian chạy với hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp khi việc gán lời gọi hàm ngoài gián tiếp được thực thi trong tệp tin nhị phân thứ nhất; thực thi tệp tin nhị phân thứ nhất được biên dịch, nhờ đó cho mỗi hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm rút gọn; thực thi lời gọi hàm gián tiếp liên kết, khi xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết không được chứa trong bảng hàm rút gọn, khi hàm được đề cập đến trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ chứa trong bản đồ nhị phân, và khi hàm được đề cập

đến trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, trong đó bảng hàm thời gian chạy được lưu trữ trong phương tiện lưu trữ bất biến được bảo vệ đọc được bởi bộ xử lý.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, trong đó bản đồ nhị phân được lưu trữ trong phương tiện lưu trữ bất biến được bảo vệ đọc được bởi bộ xử lý.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, trong đó các lời gọi hàm gián tiếp bao gồm các lời gọi hàm gián tiếp được gán như là một phần của cấu trúc dữ liệu.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, trong đó các lời gọi hàm gián tiếp còn bao gồm các lời gọi hàm gián tiếp được gán cho các hàm.

Dựa vào khía cạnh thứ hai, theo các phương án thực hiện sáng chế, trong đó các lời gọi hàm gián tiếp còn bao gồm các lời gọi hàm gián tiếp được gán như là các lời gọi khối.

Mô tả vắn tắt các hình vẽ

Các ưu điểm và các dấu hiệu trên theo sáng chế được mô tả trong phần mô tả chi tiết sau và được thể hiện ở các hình vẽ sau:

Fig.1 là hình vẽ minh họa sơ đồ khối lấy làm ví dụ của hệ thống để hỗ trợ củng cố CFI của ứng dụng phần mềm theo các phương án thực hiện sáng chế;

Fig.2 là hình vẽ minh họa sơ đồ khối đại diện các thành phần trong thiết bị điện tử để triển khai các phương án thực hiện theo các phương án thực hiện sáng chế;

Fig.3 là hình vẽ minh họa sơ đồ khối của hệ thống mà biên dịch mã nguồn để tạo mã nhị phân mà sau đó được thực thi theo các phương án thực hiện sáng chế;

Fig.4 là hình vẽ minh họa sơ đồ khối của hệ thống mà biên dịch hai mã nguồn khác nhau để tạo một mã nhị phân mà sau đó được thực thi theo các phương án thực hiện sáng chế;

Fig.5 là hình vẽ minh họa sơ đồ luồng thể hiện quá trình mà được thực hiện bởi thiết bị trong suốt thời gian chạy của mã nguồn đã được biên dịch theo các phương án thực hiện sáng chế;

Fig.6 là hình vẽ minh họa sơ đồ luồng thể hiện quá trình mà được thực hiện bởi thiết bị trong suốt thời gian chạy của mã nguồn theo các phương án thực hiện sáng chế nhờ đó mã nguồn bao gồm các lời gọi hàm gián tiếp; và

Fig.7 là hình vẽ minh họa sơ đồ luồng thể hiện quá trình mà được thực hiện bởi thiết bị được mô tả trên Fig.6 khi thiết bị thực thi lệnh trong mã nguồn theo các phương án thực hiện sáng chế.

Mô tả chi tiết sáng chế

Sáng chế đề xuất thiết bị và phương pháp cho thiết bị củng cố CFI của ứng dụng phần mềm làm ứng dụng được thực thi trên thiết bị.

Cụ thể là, khi mã nguồn được biên dịch, sáng chế sẽ tạo bảng hàm rút gọn dựa trên các lời gọi hàm gián tiếp đã được gán cho các con trỏ hàm được chứa trong tệp tin mã nguồn. Sau đó, trình biên dịch này sẽ biên dịch hàm kiểm tra CFI và sau đó hàm kiểm tra CFI này được liên kết với mỗi lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân sao cho khi tệp tin nhị phân được thực thi, hàm lời gọi hàm gián tiếp liên kết sẽ được thực thi trước khi thực thi mỗi lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân. Nên lưu ý rằng thuật ngữ chương trình và ứng

dụng phần mềm có thể được sử dụng qua lại trong suốt bản mô tả mà không xa rời sáng chế.

Ngoài ra, nếu lời gọi hàm gián tiếp trong tệp tin mã nguồn gọi hàm ngoài mà được chứa trong tệp tin mã nguồn khác, bảng hàm thời gian chạy sẽ được tạo dựa trên tệp tin được biên dịch của tệp tin mã nguồn còn lại. Nói cách khác, bảng hàm thời gian chạy được tạo dựa trên tệp tin nhị phân của tệp tin mã nguồn còn lại. Sau đó bảng hàm thời gian chạy này được tận dụng để kiểm tra nếu các lời gọi hàm ngoài là các lời gọi hợp pháp nhờ đó chặn các lời gọi bất hợp pháp. Lớp kiểm tra bổ sung này còn tăng cường việc củng cố CFI của ứng dụng phần mềm.

Chuyên gia trong lĩnh vực sẽ nhận ra rằng nhiều khối chức năng trong phần mô tả đã được dán nhãn như là các môđun trong suốt bản mô tả. Chuyên gia trong lĩnh vực cũng sẽ nhận ra rằng môđun có thể được triển khai như là các mạch điện tử, các vi mạch lôgic hoặc tổ hợp bất kỳ của các linh kiện điện và/hoặc điện tử rời rạc trong thiết bị. Ngoài ra, một chuyên gia trong lĩnh vực cũng sẽ nhận ra rằng môđun có thể được triển khai trong phần mềm sau đó có thể được thực thi bởi nhiều bộ xử lý. Theo các phương án thực hiện sáng chế, môđun cũng có thể bao gồm các lệnh máy tính hoặc mã thực thi được mà có thể ra lệnh bộ xử lý máy tính thực thi chuỗi sự kiện dựa trên các lệnh nhận được. Sự lựa chọn triển khai các môđun được dành lại như là lựa chọn thiết kế cho chuyên gia trong lĩnh vực và không giới hạn phạm vi sáng chế theo cách bất kỳ.

Các chuyên gia trong lĩnh vực hiểu rằng, khi mã của ứng dụng phần mềm được biên dịch, điều này khiến mã được dịch thành mã máy mà sau đó được thực thi trực tiếp bởi bộ xử lý hoặc máy ảo. Theo các phương án thực hiện sáng chế, mã máy này có thể bao gồm mã nhị phân được lưu trữ trong bộ nhớ của thiết bị.

Fig.1 minh họa sơ đồ khối lấy làm ví dụ của hệ thống để hỗ trợ củng cố CFI của các ứng dụng phần mềm theo các phương án thực hiện sáng chế. Hệ thống được minh họa trên Fig.1 bao gồm thiết bị 100 được tạo cấu hình để nhận các mã nguồn từ các nguồn khác nhau. Thiết bị 100 có thể bao gồm, nhưng không bị giới hạn ở, thiết bị bất kỳ có thể thực hiện các hàm tính toán không dây hoặc qua các kết nối có dây chẳng hạn các máy chủ, máy tính, máy tính tablet, máy tính di động, hoặc các netbook hoặc loại thiết bị tính toán bất kỳ có thể được tạo cấu hình để nhận và thực thi các lệnh. Các thiết bị này cũng có các môđun thực hiện các hàm tính toán và xử lý khác nhau.

Nói chung, thiết bị 100 có môđun trình biên dịch 110. Môđun trình biên dịch 110 được tạo cấu hình để nhận mã nguồn 105 và để biên dịch mã nguồn 105 để tạo tệp tin nhị phân 112. Sau đó, tệp tin nhị phân 112 được lưu trữ trong bộ nhớ 220 trong đó sau đó nó được thực thi bởi thiết bị 100.

Theo các phương án thực hiện sáng chế, thiết bị 100 cũng có thể có môđun phát triển (không được thể hiện trên hình vẽ) có thể vận hành được để tạo môi trường phát triển tích hợp (integrated development environment, IDE) để phát triển các ứng dụng phần mềm. Môđun phát triển có thể được tạo cấu hình để hỗ trợ thiết kế, phát triển, kiểm thử, và/hoặc khai triển các ứng dụng phần mềm trong thiết bị 100. Môđun phát triển cũng được tạo cấu hình để hỗ trợ các ngôn ngữ lập trình khác và để tích hợp các môđun khác nhau để đơn giản hóa việc phát triển các ứng dụng phần mềm. Ngoài ra, môđun phát triển cũng được tạo cấu hình sao cho nhà phát triển có thể tạo và chỉnh sửa mã nguồn cho dự án và biên dịch mã nguồn để thực thi ứng dụng phần mềm.

Thiết bị 100 có thể được tạo cấu hình để truyền thông không dây với các thiết bị khác hoặc các nguồn khác qua mạng Internet hoặc các mạng

không dây chẳng hạn, nhưng không bị giới hạn ở, các mạng tế bào, các mạng vệ tinh, các mạng viễn thông, hoặc các mạng diện rộng (Wide Area Network, WAN). Truyền thông giữa thiết bị 100 và các thiết bị ngoài khác cũng có thể diễn ra qua phương tiện hữu tuyến. Qua các phương tiện truyền thông này, các người dùng ngoài có thể truy nhập từ xa các môđun chứa trong thiết bị 100 để thực hiện phát triển các ứng dụng phần mềm hoặc chạy bảo mật các ứng dụng phần mềm chứa trong thiết bị 100.

Fig.2 minh họa sơ đồ khối đại diện các thành phần của môđun 200 có thể nằm trong thiết bị 100 để triển khai các phương án thực hiện theo các phương án thực hiện sáng chế. Chuyên gia trong lĩnh vực sẽ nhận ra rằng cấu hình chính xác của mỗi thiết bị không dây nằm trong các thiết bị hoặc điểm truy nhập (access point, AP) có thể khác nhau và cấu hình chính xác của môđun 200 có thể thay đổi và Fig.2 được đưa ra chỉ nhờ ví dụ.

Theo các phương án thực hiện sáng chế, môđun 200 bao gồm bộ điều khiển 201 và giao diện người dùng 202. Giao diện người dùng 202 được bố trí để cho phép các tương tác thủ công giữa người dùng và môđun 200 và nhằm mục đích này bao gồm các thành phần nhập/xuất (input/output, I/O) được yêu cầu cho người dùng để nhập lệnh vào môđun điều khiển 200. Chuyên gia trong lĩnh vực sẽ nhận ra rằng các thành phần của giao diện người dùng 202 có thể thay đổi từ phương án thực hiện này sang phương án thực hiện khác nhưng đặc trưng sẽ bao gồm một hoặc nhiều trong số phần hiển thị 240, bàn phím 235 và bảng di chuột cảm ứng 236.

Bộ điều khiển 201 đang truyền thông dữ liệu với giao diện người dùng 202 qua đường truyền 215 và bao gồm bộ nhớ 220, bộ xử lý 205 được lắp trên bảng mạch mà xử lý lệnh và dữ liệu để thực hiện phương pháp theo phương án thực hiện, hệ điều hành 206, giao diện I/O 230 để truyền thông với giao diện người dùng 202 và giao diện truyền thông, theo phương án thực hiện ở dạng thẻ mạng 250. Thẻ mạng 250 có thể, chẳng hạn, được

tận dụng để gửi dữ liệu từ thiết bị điện tử 200 qua mạng hữu tuyến hoặc không dây đến các thiết bị xử lý khác hoặc để nhận dữ liệu qua mạng hữu tuyến hoặc không dây. Các mạng không dây mà có thể được tận dụng bởi thẻ mạng 250 bao gồm, nhưng không bị giới hạn ở, Wi-Fi, Bluetooth, truyền thông trường gần (Near Field Communication, NFC), các mạng tế bào, các mạng vệ tinh, các mạng viễn thông, WAN, v.v..

Bộ nhớ 220 và hệ điều hành 206 đang truyền thông dữ liệu với CPU 205 qua đường truyền 210. Các thành phần bộ nhớ bao gồm cả bộ nhớ khả biến lẫn bất biến và nhiều hơn một trong mỗi loại bộ nhớ, bao gồm bộ nhớ truy nhập ngẫu nhiên (Random Access Memory, RAM) 220, bộ nhớ chỉ đọc (Read Only Memory, ROM) 225 và thiết bị lưu trữ khối 245, cuối cùng bao gồm một hoặc nhiều ổ trạng thái rắn (solid-state drive, SSD) hoặc loại ổ khác bất kỳ. Bộ nhớ 220 cũng bao gồm bộ lưu trữ bảo mật 246 để lưu trữ bảo mật dữ liệu bảo mật. Nên lưu ý rằng các nội dung trong bộ lưu trữ bảo mật 246 có thể chỉ được truy nhập bởi siêu người dùng hoặc quản trị viên của môđun 200 và có thể không được truy nhập bởi người dùng bất kỳ của môđun 200. Chuyên gia trong lĩnh vực sẽ nhận ra rằng các thành phần bộ nhớ nêu trên bao gồm các phương tiện máy tính đọc được bất biến và được xem xét để bao gồm tất cả các phương tiện máy tính đọc được cho tín hiệu lan truyền khả biến. Thông thường, các lệnh được lưu trữ như mà mã chương trình trong các thành phần bộ nhớ nhưng cũng có thể được nối cứng. Bộ nhớ 220 có thể bao gồm kernel và/hoặc các môđun chương trình chẳng hạn ứng dụng phần mềm có thể được lưu trữ trong bộ nhớ khả biến hoặc bất biến.

Ở đây, thuật ngữ "bộ xử lý" được sử dụng để đề cập chung đến thiết bị hoặc linh kiện bất kỳ có thể xử lý các lệnh này và có thể bao gồm: bộ vi xử lý, bộ vi điều khiển, thiết bị lôgic lập trình được hoặc thiết bị tính toán khác. Tức là, bộ xử lý 205 có thể được cung cấp bởi hệ mạch lô gic thích

hợp bất kỳ để nhận các đầu vào, xử lý chúng theo các lệnh được lưu trữ trong bộ nhớ và tạo đầu ra (chẳng hạn đến các thành phần bộ nhớ hoặc trên phần hiển thị 240). Theo phương án thực hiện, bộ xử lý 205 có thể bộ xử lý một lõi hoặc nhiều lõi với không gian bộ nhớ có thể đánh địa chỉ. Trong một ví dụ, bộ xử lý 205 có thể là nhiều lõi, bao gồm—chẳng hạn—CPU 8 lõi.

Fig.3 minh họa các sơ đồ khối của hệ thống trong thiết bị 100 được tạo cấu hình để biên dịch mã nguồn để tạo mã nhị phân mà sau đó được thực thi theo các phương án thực hiện sáng chế. Ngoài ra, Fig.3 cũng minh họa sơ đồ luồng thể hiện tương tác của mã nguồn 305 với môđun trình biên dịch 310 trước khi tệp tin nhị phân 320 thu được được cấp cho bản đồ bộ nhớ 330 trong đó nó được thực thi tiếp sau đó.

Khi hoạt động, trước hết mã nguồn 305 được cấp cho môđun trình biên dịch 310. Nên lưu ý rằng theo các phương án thực hiện sáng chế, hàm kiểm tra CFI 307 được tạo bởi trình biên dịch 310 như là mã nhị phân khi trình biên dịch 310 bắt đầu biên dịch mã nguồn 305. Thực tế, hàm kiểm tra CFI 307 có thể được gọi bởi trình biên dịch 310.

Hàm kiểm tra CFI 307 chủ yếu là hàm mà khi được gọi, kiểm tra liệu hàm đối tượng được chứa trong tệp tin đích. Theo các phương án thực hiện sáng chế, khi hàm kiểm tra CFI 307 được gọi, hàm kiểm tra CFI 307 sẽ xác định liệu hàm đối tượng được chứa trong bảng hàm rút gọn 315. Nếu hàm đối tượng không được chứa trong bảng hàm rút gọn 315, hàm kiểm tra CFI 307 sẽ đánh dấu câu gọi hàm đối tượng như là vi phạm và sau đó sẽ ngăn không cho hàm đối tượng thực thi. Việc tạo bảng hàm rút gọn 315 sẽ được mô tả chi tiết dưới đây.

Như đối với mã nguồn 305, mã nguồn này có thể bao gồm mã được viết bằng cách sử dụng ngôn ngữ lập trình bất kỳ mà các chuyên gia trong lĩnh vực đã quen thuộc chẳng hạn ngôn ngữ lập trình C++.

Theo các phương án thực hiện sáng chế, môđun trình biên dịch 310 cũng có trình biên dịch phân tích gán con trỏ hàm (function pointer assignment analysis, FPAA) 311. Trình biên dịch FPAA 311 được tạo cấu hình để tạo bảng hàm rút gọn 315 dựa trên mã nguồn 305 khi mã nguồn 305 đang được biên dịch bởi trình biên dịch 310. Các hoạt động bên trong của trình biên dịch FPAA 311 được mô tả tốt nhất bằng cách sử dụng mẫu của mã nguồn 305 như được minh họa dưới đây. Chuyên gia trong lĩnh vực sẽ nhận ra rằng mã nguồn này được thể hiện chỉ như là ví dụ và không giới hạn sáng chế theo cách bất kỳ.

```
typedef void (*ftype) (void);
void func1 (void) { printf("func1\n");}
void func2 (void) { printf("func2\n");}
void func3 (void) { printf("func3\n");}

struct S {
        ftype s_fp;
} s;
s.s_fp = func1;
int main (void) {
        ftype fp = func2;
        _asm_ _volatile_("mov1 $func1, %eax\n", call *%eax");
        return 0;
        }
```

Khi mã nguồn 305 được biên dịch bởi trình biên dịch 310, trình biên dịch FPAA 311 được tạo cấu hình để dò và truy xuất tất cả các lời gọi hàm gián tiếp sao cho, nhưng không giới hạn ở, các phép gán lời gọi nhờ

đó các hàm được gán cho các con trỏ hàm, từ mã nguồn 305. Theo các phương án thực hiện sáng chế, trình biên dịch FPAA 311 được tạo cấu hình để dò các lời gọi hàm gián tiếp được gán như là một phần của cấu trúc dữ liệu, các lời gọi hàm gián tiếp được gán cho các hàm hoặc các lời gọi hàm gián tiếp được gán như là các lời gọi hàm khối (assembly, ASM). Các lời gọi hàm gián tiếp lấy làm ví dụ như được trích rút từ mã nguồn nêu trên được nêu dưới đây.

```
s.s_fp = func1;  ← Gán cấu trúc dữ liệu
ftype fp = func2; ← Gán trực tiếp
_("mov1 $func1, %eax\n", ← Gán ASM
```

Một chuyên gia trong lĩnh vực sẽ nhận ra từ đây, tham chiếu bất kỳ được thực hiện với lời gọi hàm gián tiếp có thể liên quan đến con trỏ hàm hoặc lời gọi hàm gián tiếp bất kỳ của loại nêu trên.

Do các loại khác nhau của các lời gọi hàm gián tiếp được dò thấy, nên các hàm được đề cập bởi các lời gọi hàm gián tiếp này hoặc trong các phép gán con trỏ hàm này sau đó được thêm vào bảng hàm rút gọn 315. Trong mã nguồn ví dụ nêu trên, điều này nghĩa là các hàm sau cuối cùng đều được thêm vào bảng hàm rút gọn 315.

```
func1;
func2;
```

Do trình biên dịch 310 biên dịch mã nguồn 305, trình biên dịch 310 có thể đồng thời tạo hàm kiểm tra CFI 307.

Khi đang hoạt động, mã nguồn 305 sẽ được biên dịch thông thường bởi trình biên dịch 310 để tạo mã nguồn được biên dịch 312. Tuy nhiên, khi trình biên dịch 310 gặp lời gọi hàm gián tiếp trong mã nguồn 305, trình biên dịch 310 sẽ biên dịch lời gọi hàm gián tiếp và sau đó liên kết lời gọi hàm gián tiếp được biên dịch với hàm kiểm tra CFI 307 trong mã

nguồn được biên dịch 312 sao cho khi mã nguồn được biên dịch 312 được thực thi, hàm kiểm tra CFI được liên kết 307 sẽ được thực thi hoặc sẽ được chạy trước khi lời gọi hàm gián tiếp được biên dịch có thể được thực thi. Sau đó, hàm được đề cập bởi lời gọi hàm gián tiếp cũng sẽ được thêm vào bảng hàm rút gọn 315.

Nói theo cách khác, việc liên kết lời gọi hàm gián tiếp được biên dịch với hàm kiểm tra CFI 307 bao gồm hàm kiểm tra CFI 307 được thêm vào mã nguồn được biên dịch 312 ở dòng trước khi nhị phân của lời gọi hàm gián tiếp được biên dịch sao cho hàm kiểm tra CFI 307 sẽ được thực thi bởi hệ điều hành trước khi hệ điều hành chạy lời gọi hàm gián tiếp được biên dịch.

Quá trình liên kết hàm kiểm tra CFI 307 này với các lời gọi hàm gián tiếp được biên dịch và thêm các hàm được đề cập đến bởi các lời gọi hàm gián tiếp vào bảng hàm rút gọn 315 tự lặp lại cho đến khi mỗi lời gọi hàm gián tiếp được biên dịch trong mã nguồn 305 đã được liên kết với hàm kiểm tra CFI 307 và bảng hàm rút gọn 315 đã được tạo theo đó. Ở giai đoạn này, nhị phân 320 sau đó sẽ bao gồm bảng hàm rút gọn 315, hàm kiểm tra CFI 307 và mã nguồn được biên dịch 312 nhờ đó mã nguồn được biên dịch 312 sẽ bao gồm tất cả các liên kết liên quan hoặc chèn vào hàm kiểm tra CFI 307.

Sau đó, nhị phân 320 được cấp để xử lý bộ nhớ 330 mà được chứa trong bộ nhớ của thiết bị 100. Sau đó, mã nguồn được biên dịch 312, bảng hàm rút gọn 315 và hàm kiểm tra CFI 307 được trích rút từ nhị phân 320 và được lưu trữ ở đoạn văn bản 334 mà được đặt trong bộ nhớ xử lý 330.

Sau đó, thiết bị 100 tiếp tục thực thi mã nguồn được biên dịch 312 như được lưu trữ trong đoạn văn bản 334. Mã nguồn được biên dịch 312 được chạy bình thường khi gặp hàm kiểm tra CFI 307 đã được chèn vào trước

đó trong mã 312. Khi điều này xảy ra, thiết bị 100 sẽ kiểm tra bảng hàm rút gọn 315 để xác định nếu hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI 307 được chứa trong bảng hàm rút gọn 315. Nếu thiết bị 100 xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI 307 được chứa trong bảng hàm rút gọn 315, thì thiết bị 100 sẽ thực thi liên kết lời gọi hàm gián tiếp như bình thường. Sau đó, thiết bị 100 tiếp tục thực thi mã nguồn được biên dịch 312 như bình thường cho đến khi hàm kiểm tra CFI 307 khác được dò thấy và quá trình nêu trên được lặp lại.

Ngược lại, nếu ở giai đoạn bất kỳ thiết bị 100 xác định rằng lời gọi hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI 307 không được chứa trong bảng hàm rút gọn 315, thì thiết bị 100 sẽ hủy bỏ thực thi liên kết lời gọi hàm gián tiếp do điều này nghĩa là vi phạm đã xảy ra. Ở giai đoạn này, thiết bị 100 có thể được tạo cấu hình để bỏ qua lời gọi hàm gián tiếp bị ảnh hưởng và tiếp tục với việc thực thi nhị phân 320, tăng cảnh báo thông báo cho quản trị viên rằng vi phạm đã xảy ra hoặc thiết bị 100 có thể được tạo cấu hình để hủy bỏ việc chạy nhị phân 320 hoàn toàn.

Fig.4 minh họa các sơ đồ khối của hệ thống trong thiết bị 100 được tạo cấu hình để biên dịch hai mã nguồn khác nhau để tạo mã nhị phân mà sau đó được thực thi theo các phương án thực hiện sáng chế.

Khi hoạt động, mã nguồn 305 và mã nguồn 405 được bố trí giống nhau cho môđun trình biên dịch 310 trong khi hàm kiểm tra CFI 307 được tạo bởi trình biên dịch 310. Các hàm của môđun trình biên dịch FPAA 311, trình biên dịch 310 và hàm kiểm tra CFI 307 như được mô tả trước đó liên quan đến Fig.3. Tuy nhiên, theo phương án thực hiện được minh họa trên Fig.4, mã nguồn 305 chứa các lời gọi hàm gián tiếp đến

các hàm được tìm thấy trong mã nguồn 405. Nói cách khác, mã nguồn 405 chứa các lời gọi hàm ngoài xuất phát từ mã nguồn 305.

Ví dụ khác của mã nguồn 305 được nêu dưới đây cùng với ví dụ của mã nguồn 405. Chuyên gia trong lĩnh vực sẽ nhận ra rằng các mã nguồn này được thể hiện dưới dạng chỉ các ví dụ và không giới hạn sáng chế theo cách bất kỳ.

305

```
void funcA2 (void) { printf("funcA2\n");}
void funcA3 (void) { printf("funcA3\n");}
funcA1( ) {
    load 405;
    f1 = dlsym(405, "funcB");
    f1( );
    f2 = funcA2;
    f2( ); }
```

405

```
void funcB (void) { printf("funcB\n");}
```

Ở ví dụ nêu trên, khi trình biên dịch 310 nhận và bắt đầu biên dịch mã nguồn 305, trình biên dịch FPAA 311 sẽ dò thấy các lời gọi hàm gián tiếp được chứa trong trình biên dịch 310. Các chức năng được đề cập đến bởi các lời gọi hàm gián tiếp này sau đó sẽ được thêm vào bảng hàm rút gọn. Do trình biên dịch 310 bắt đầu biên dịch mã nguồn 305, trình biên dịch 310 có thể đồng thời tạo hàm kiểm tra CFI 307.

Tương tự, mã nguồn 305 sẽ được biên dịch như bình thường bởi trình biên dịch 310 để tạo mã nguồn được biên dịch. Khi trình biên dịch 310

bắt gặp lời gọi hàm gián tiếp trong mã nguồn 305, trình biên dịch 310 sẽ biên dịch lời gọi hàm gián tiếp và tiếp theo liên kết lời gọi hàm gián tiếp được biên dịch với hàm kiểm tra CFI trong mã nguồn được biên dịch sao cho khi mã nguồn được biên dịch được thực thi bởi hệ điều hành, hàm kiểm tra CFI được liên kết sẽ được thực thi hoặc sẽ được chạy trước khi lời gọi hàm gián tiếp được biên dịch có thể được thực thi. Sau đó, hàm được đề cập đến bởi lời gọi hàm gián tiếp sẽ được thêm vào bảng hàm rút gọn.

Theo phương án thực hiện sáng chế, khi trình biên dịch 310 gặp lời gọi hàm ngoài gián tiếp chẳng hạn hàm dlsym( ), trình biên dịch 310 sẽ biên dịch lời gọi hàm ngoài gián tiếp này và sau đó liên kết lời gọi hàm ngoài gián tiếp được biên dịch với hàm kiểm tra CFI trong mã nguồn được biên dịch sao cho khi mã nguồn được biên dịch được thực thi bởi hệ điều hành, hàm kiểm tra CFI được liên kết sẽ được thực thi hoặc sẽ được chạy trước khi lời gọi hàm ngoài gián tiếp được biên dịch có thể được thực thi. Tuy nhiên, lời gọi hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp (chẳng hạn dlsym( )) sẽ không được thêm vào bảng hàm rút gọn.

Quá trình liên kết hàm kiểm tra CFI này với các lời gọi hàm gián tiếp biên dịch, các lời gọi hàm ngoài gián tiếp và việc thêm các chức năng được đề cập đến bởi các lời gọi hàm gián tiếp cụ thể vào bảng hàm rút gọn tự lặp lại cho đến khi mỗi trong số các lời gọi hàm gián tiếp được biên dịch và các lời gọi hàm ngoài gián tiếp trong mã nguồn 305 đã được liên kết với hàm kiểm tra CFI và bảng hàm rút gọn đã được tạo theo đó. Để hoàn thiện, ở ví dụ nêu trên, điều này nghĩa là bảng hàm rút gọn sẽ bao gồm chỉ "*funcA2*".

Ở giai đoạn này, nhị phân 320 sẽ bao gồm bảng hàm rút gọn, hàm kiểm tra CFI và mã nguồn được biên dịch nhờ đó mã nguồn được biên

dịch sẽ bao gồm tất cả các liên kết liên quan hoặc chèm thêm hàm kiểm tra CFI.

Một cách đồng thời hoặc sau khi nhị phân 320 đã được tạo, trình biên dịch 310 sẽ tiến hành biên dịch mã nguồn 405 một cách tương tự được mô tả trên đây nhờ đó bảng hàm rút gọn được tạo cho mã nguồn 405 và các hàm kiểm tra CFI được liên kết với các lời gọi hàm gián tiếp và/hoặc các lời gọi hàm ngoài gián tiếp trong mã nguồn 405.

Ở giai đoạn này, nhị phân 420 sẽ bao gồm bảng hàm rút gọn được liên kết với mã nguồn 405, hàm kiểm tra CFI và mã nguồn được biên dịch 405 nhờ đó mã nguồn được biên dịch 405 sẽ bao gồm tất cả các liên kết hoặc chèn thêm liên quan của hàm kiểm tra CFI. Nên lưu ý rằng theo các phương án thực hiện khác của sáng chế, trình biên dịch FPAA 311 có thể được tạo cấu hình để biên dịch mã nguồn 405 một cách bình thường để tạo nhị phân 420, tức là không tạo bảng hàm rút gọn liên kết và không liên kết các hàm kiểm tra CFI với các lời gọi hàm gián tiếp trong mã nguồn được biên dịch 405.

Sau đó, nhị phân 320 và nhị phân 420 được cấp cho bộ nhớ xử lý 330 mà được chứa trong bộ nhớ của thiết bị 100. Sau đó, bảng hàm rút gọn liên kết, hàm kiểm tra CFI và mã nguồn được biên dịch 305 được trích rút khỏi nhị phân 320 và được lưu trữ ở đoạn văn bản 334. Một cách tương tự, bảng hàm rút gọn và hàm kiểm tra CFI được liên kết với mã nguồn được biên dịch 405 cùng với mã nguồn được biên dịch 405 sau đó được trích rút từ nhị phân 420 và được lưu trữ ở đoạn văn bản 335.

Ngoài ra, như được minh họa trên Fig.4, thư viện chia sẻ 430 cũng được cấp cho bộ nhớ xử lý 330 sao cho khi các chương trình được biên dịch được thực thi bởi hệ điều hành, các chương trình được biên dịch có thể chia sẻ các thư viện được chia sẻ và/hoặc các môđun được lưu trữ trong thư viện chia sẻ 430.

Cụ thể là, được chứa trong thư viện chia sẻ 430 là môđun thời gian chạy FPAA 422. Môđun thời gian chạy FPAA 422 được tạo cấu hình để dò và cập nhật bảng hàm thời gian chạy 425 với các hàm ngoài được đề cập đến trong các lời gọi hàm ngoài gián tiếp được khai báo trong khi chạy chương trình được biên dịch. Nên lưu ý rằng bảng hàm thời gian chạy 425 được tạo bởi môđun thời gian chạy FPAA 422 và được lưu trữ trong bộ nhớ bảo vệ trong bộ nhớ xử lý 330. Theo các phương án thực hiện sáng chế, môđun thời gian chạy FPAA 422 sẽ được kích hoạt khi hệ điều hành gặp phép gán lời gọi hàm ngoài gián tiếp chẳng hạn hàm dlsym( ) trong mã nguồn được biên dịch mà được thực thi. Sau đó, hàm ngoài được đề cập đến trong hàm dlsym( ) được thêm vào bảng hàm thời gian chạy 425. Hoạt động chi tiết của môđun thời gian chạy FPAA 422 được mô tả chi tiết trong phần sau dưới đây. Ngoài ra, nên lưu ý rằng hàm dlsym( ) được đề cập đến trong phần mô tả này đã được chỉnh sửa từ hàm dlsym( ) có thể được tìm thấy trong thư viện chuẩn Stdlib. Hàm dlsym( ) đã được chỉnh sửa sao cho hàm dlsym( ) này sẽ kích hoạt môđun thời gian chạy FPAA 422 mỗi lần hàm dlsym( ) này được sử dụng trong khi gán lời gọi hàm ngoài gián tiếp.

Khi vận hành, khi hệ điều hành thực thi mã nguồn được biên dịch 305 như được lưu trữ trong bộ nhớ xử lý 330, mã nguồn được biên dịch 305 sẽ chạy như bình thường cho đến khi gặp hoạt động hoặc lệnh "nạp" trong mã nguồn được biên dịch 305. Hoạt động "nạp" này chủ yếu là lệnh nạp hoặc đọc mã nguồn được biên dịch khác trong bộ nhớ xử lý 330 và/hoặc để nạp thư viện được chia sẻ như được chứa trong thư viện chia sẻ 430. Nên lưu ý rằng lệnh "nạp" trong bản mô tả đã được chỉnh sửa từ lệnh "nạp" có thể được tìm thấy trong thư viện chuẩn Stdlib. Lệnh "nạp" đã bị thay đổi sao cho bên cạnh việc thực hiện chức năng chuẩn, khi hệ điều hành gặp hoạt động "nạp", bản đồ nhị phân 427 (được tạo bởi

môđun thời gian chạy FPAA 422 và được lưu trữ trong bộ nhớ được bảo vệ trong bộ nhớ xử lý 330) sẽ được cập nhật với địa chỉ của mã nguồn được biên dịch hoặc thư viện chia sẻ được nạp vào mã nguồn được biên dịch 305. Trong ví dụ nêu trên, khi hệ điều hành gặp lệnh "*nạp 405*" trong mã nguồn được biên dịch 305, điều này nghĩa là địa chỉ của mã nguồn được biên dịch "*405*" sẽ được nạp vào bản đồ nhị phân 427.

Sau đó, môđun thời gian chạy FPAA 422 tiếp tục cập nhật bản đồ ánh xạ 427 với các địa chỉ của các mã nguồn được biên dịch và/hoặc các thư viện chia sẻ do các tham số này được nạp bởi hệ điều hành khi mã nguồn được biên dịch 305 được thực thi.

Ngoài ra, trong khi chạy mã nguồn được biên dịch 305, khi hệ điều hành gặp lời gọi hàm ngoài gián tiếp chẳng hạn hàm dlsym( ), môđun thời gian chạy FPAA 422 sẽ được kích hoạt để cập nhật bảng hàm thời gian chạy 425 với hàm được đề cập đến trong hàm dlsym( ). Trong ví dụ nêu trên, điều này nghĩa là khi hệ điều hành thực thi lệnh "*f1 = dlsym(405, "funcB")*", môđun thời gian chạy FPAA 422 sẽ thêm "funcB" từ mã nguồn được biên dịch "*405*" vào bảng hàm thời gian chạy 425. Nên lưu ý rằng ở giai đoạn này, địa chỉ của mã nguồn "405" đã được nạp và thực tế, đã được thêm vào bản đồ ánh xạ 427. Quá trình cập nhật bảng hàm thời gian chạy 425 lặp lại mỗi lần hệ điều hành gặp hàm dlsym( ) hoặc chức năng tương tự trong mã nguồn được biên dịch mà được thực thi.

Thông thường, lệnh "nạp" sẽ phải được khai báo trước khi hàm "dlsym( )" được khai báo do hàm "dlsym( )" có thể đề cập đến bản đồ ánh xạ 427 để có địa chỉ của mã nguồn ngoài được đề cập đến trong hàm "dlsym( )". Chuyên gia trong lĩnh vực sẽ nhận ra rằng đây không phải là yêu cầu và rằng sáng chế cũng đề cập đến các biến thể khác trong việc bố trí các hàm và/hoặc các lệnh.

Do hệ điều hành thực thi mã nguồn được biên dịch 305, khi hệ điều hành gặp hàm kiểm tra CFI đã được chèn trước đó vào mã nguồn được biên dịch 305, hệ điều hành sẽ kiểm tra trước nếu hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI được chứa trong bảng hàm rút gọn 315. Để ngắn gọn, từ đây, các tham chiếu bất kỳ được thực hiện đến các lời gọi hàm gián tiếp đề cập đến cả các lời gọi hàm gián tiếp lẫn các lời gọi hàm ngoài gián tiếp trừ khi thực hiện các phân biệt tường minh. Nếu hệ điều hành xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI được chứa trong bảng hàm rút gọn được liên kết với mã nguồn được biên dịch 305, thì thiết bị 100 sẽ thực thi lời gọi hàm gián tiếp liên kết như bình thường.

Ngược lại, nếu hệ điều hành xác định rằng lời gọi hàm được đề cập đến trong lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI không được chứa trong bảng hàm rút gọn được liên kết với mã nguồn 305, hệ điều hành sẽ cố gắng xác định liệu hàm kiểm tra CFI được liên kết với lời gọi hàm ngoài gián tiếp. Hệ điều hành thực hiện điều này bằng cách cố gắng truy xuất địa chỉ được đề cập đến trong lời gọi hàm gián tiếp từ bản đồ ánh xạ 427. Nếu hệ điều hành không thể thu thập địa chỉ liên quan, thì hệ điều hành sẽ hủy bỏ thực thi lời gọi hàm gián tiếp liên kết do điều này nghĩa là đã xuất hiện vi phạm.

Ở ví dụ nêu trên, giả sử rằng hàm "*f1( )*" được gọi bởi hệ điều hành, tức là, lời gọi hàm ngoài gián tiếp được gọi, điều này nghĩa là hệ điều hành sẽ cố gắng thu thập địa chỉ của "*405*" từ bản đồ ánh xạ 427. Do địa chỉ của "405" đã được thêm trước đó vào bản đồ ánh xạ 427, nên địa chỉ này sẽ được nạp bởi hệ điều hành. Sau đó, hệ điều hành khẳng định rằng lời gọi hàm gián tiếp là loại lời gọi hàm ngoài gián tiếp do địa chỉ đã được nạp thành công.

Dựa vào mã nguồn được biên dịch 405 được nạp, sau đó hệ điều hành sẽ xác định nếu mã nguồn được biên dịch 405 liên quan đến mã tính toán kế thừa. Nếu hệ điều hành xác định rằng mã nguồn được biên dịch 405 liên quan đến mã tính toán kế thừa, thì hệ điều hành sẽ thực thi lời gọi hàm ngoài gián tiếp như bình thường. Trong bản mô tả này, mã tính toán kế thừa đề cập đến mã nguồn đã không được biên dịch theo các phương án thực hiện sáng chế.

Ngược lại, nếu hệ điều hành xác định rằng mã nguồn được biên dịch 405 không liên quan đến mã tính toán kế thừa và thay vào đó được biên dịch theo các phương án thực hiện sáng chế, thì hệ điều hành sẽ kiểm tra bảng hàm rút gọn được liên kết với mã nguồn được biên dịch 405 để xác định nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong đó. Nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm rút gọn được liên kết, thì lời gọi hàm ngoài gián tiếp được thực thi như bình thường bởi hệ điều hành.

Nếu không được chứa trong bảng hàm rút gọn liên kết, thì hệ điều hành tiến hành kiểm tra liệu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm thời gian chạy 425. Trong ví dụ này, do môđun thời gian chạy FPAA 422 đã thêm vào trước đó "*funcB*" (tức là, hàm được đề cập đến trong f1( )" vào bảng hàm thời gian chạy 425, điều này nghĩa là hệ điều hành sẽ xác định rằng "*funcB*" có mặt trong bảng hàm thời gian chạy 425. Sau đó, hệ điều hành sẽ thực thi lời gọi hàm ngoài gián tiếp, "*f1( )*", như bình thường. Nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp không được chứa trong bảng hàm thời gian chạy 425, thì hệ điều hành hủy bỏ chạy chương trình do điều này nghĩa là vi phạm đã xuất hiện. Ở giai đoạn này, thiết bị 100 có thể được tạo cấu hình để bỏ qua cả lời gọi hàm gián tiếp bị ảnh hưởng lẫn cảnh báo cho

quản trị viên hoặc thiết bị 100 có thể được tạo cấu hình để loại bỏ hoàn toàn việc chạy nhị phân 320.

Theo các phương án thực hiện sáng chế, phương pháp cho thiết bị để củng cố CFI của ứng dụng phần mềm bao gồm thiết bị thực hiện các bước sau:

Bước 1, tiếp nhận tệp tin mã nguồn thứ nhất cho ứng dụng phần mềm nhờ đó tệp tin mã nguồn thứ nhất bao gồm các hàm và các lời gọi hàm gián tiếp;

Bước 2, biên dịch mã nguồn thứ nhất để tạo tệp tin nhị phân thứ nhất nhờ đó việc biên dịch mã nguồn thứ nhất bao gồm cho mỗi lời gọi hàm gián tiếp được biên dịch,

Bước phụ (a), liên kết hàm kiểm tra CFI với mỗi lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân thứ nhất sao cho hàm kiểm tra CFI liên kết sẽ được thực thi trước khi thực thi mỗi lời gọi hàm gián tiếp được biên dịch; và

Bước phụ (b), thêm hàm, được đề cập đến trong mỗi lời gọi hàm gián tiếp được biên dịch, vào bảng hàm rút gọn.

Để đề xuất phương pháp này, cần quá trình để tạo cấu hình thiết bị để củng cố CFI của ứng dụng phần mềm. Phần mô tả sau và các hình vẽ từ Fig.5 đến Fig.7 mô tả các phương án thực hiện các quá trình cung cấp các bước cần thiết theo sáng chế.

Fig.5 minh họa quá trình 500 được thực hiện bởi thiết bị trong suốt thời gian chạy của mã nguồn đã được biên dịch theo các phương án thực hiện sáng chế. Quá trình 500 bắt đầu ở bước 505 bằng cách nạp tệp tin mã nguồn đã được biên dịch theo các phương án thực hiện sáng chế. Nên lưu ý rằng trong khi biên dịch mã nguồn này, các hàm kiểm tra CFI đã được liên kết với các lời gọi hàm gián tiếp và bảng hàm rút gọn đã được

tạo dựa trên các lời gọi hàm gián tiếp được tìm thấy trong mã nguồn được biên dịch.

Sau đó, quá trình 500 chạy mã nguồn được nạp như bình thường. Khi quá trình 500 gặp hàm kiểm tra CFI mà trước đó đã được nạp vào mã nguồn ở bước 510, quá trình 500 sẽ nhận diện lời gọi hàm gián tiếp được liên kết với hàm kiểm tra CFI. Sau đó, quá trình 500 xác định ở bước 515 nếu hàm được đề cập đến trong lời gọi hàm gián tiếp được nhận diện được chứa trong bảng hàm rút gọn (RFT) được tạo trước đó. Nếu quá trình 500 xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp không được chứa trong RFT, thì quá trình 500 sẽ hủy bỏ việc chạy mã nguồn được nạp do điều này nghĩa là vi phạm đã xảy ra. Sau đó, quá trình 500 kết thúc.

Ngược lại, nếu quá trình 500 xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp được nhận diện được chứa trong RFT được tạo trước đó, quá trình 500 sẽ chuyển sang bước 520. Ở bước 520, sau đó quá trình 500 sẽ thực thi CFI như bình thường. Khi điều này được thực thi, sau đó quá trình 500 chuyển sang bước 525 nhờ đó quá trình 500 tiếp tục thực thi mã nguồn như bình thường cho đến khi quá trình 500 gặp hàm kiểm tra CFI khác hoặc cho đến khi tất cả mã trong mã nguồn đã được thực thi. Nếu gặp hàm kiểm tra CFI khác, quá trình 500 sẽ trở về bước 510. Sau đó, quá trình 500 lặp lại các bước 510 đến 525. Khi tất cả mã trong mã nguồn đã được thực thi, thì quá trình 500 kết thúc.

Fig.6 minh họa quá trình 600 mà được thực hiện bởi thiết bị trong suốt thời gian chạy của mã nguồn A đã được biên dịch theo các phương án thực hiện sáng chế nhờ đó mã nguồn A này bao gồm các lời gọi hàm gián tiếp và các lời gọi hàm ngoài gián tiếp đến các chức năng trong mã nguồn B. Quá trình 600 bắt đầu ở bước 605 với thiết bị nạp nguồn được biên dịch A. Nên lưu ý rằng trong khi biên dịch mã nguồn A, các hàm kiểm tra

CFI đã được liên kết với các lời gọi hàm ngoài gián tiếp và bảng hàm rút gọn đã được tạo dựa trên các lời gọi hàm gián tiếp được tìm thấy trong mã nguồn được biên dịch A như đã được đề cập trước đó.

Khi quá trình 600 gặp việc gán lệnh hàm ngoài gián tiếp hoặc lệnh nạp trong mã nguồn A ở bước 610, quá trình 600 sẽ trước hết nhận diện lệnh đang diễn ra. Nếu gặp lệnh "*nạp*", quá trình 600 sẽ chuyển sang bước 625 nhờ đó quá trình 600 sau đó nhận diện tệp tin nguồn còn lại hoặc tệp tin sẽ được nạp trong khi chạy mã nguồn A. Sau đó, quá trình 600 tiếp tục ở bước 630 để cập nhật bản đồ ánh xạ với địa chỉ của tệp tin nguồn hoặc thư viện được nhận diện trong suốt bước 610 trước. Theo phương án thực hiện, nếu quá trình 600 gặp lệnh "nạp B", điều này nghĩa là địa chỉ của mã nguồn B sẽ được thêm vào bản đồ nhị phân.

Theo cách khác, nếu ở bước 610 quá trình 600 gặp việc gán lời gọi hàm ngoài gián tiếp, thì quá trình 600 nhận diện hàm và mã nguồn ngoài hoặc thư viện được đề cập đến trong lời gọi hàm ngoài gián tiếp. Dựa trên mã nguồn ngoài được đề cập đến trong lời gọi hàm ngoài gián tiếp và địa chỉ của mã nguồn ngoài trong bản đồ nhị phân, hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp sau đó được thêm vào bảng hàm thời gian chạy bởi quá trình 600 ở bước 620.

Sau đó, quá trình 600 tiến hành như bình thường để chọn lệnh tiếp theo trong mã nguồn A ở bước 635. Đối với mỗi lệnh được chọn, quá trình 600 sẽ truy vấn lệnh đó ở bước 640 để xác định liệu lệnh này bao gồm phép gán khác của lệnh hàm ngoài gián tiếp hoặc lệnh nạp trong mã nguồn A. Nếu lệnh này thỏa mãn các điều kiện ở bước 640, thì quá trình 600 trở về bước 610 nhờ đó quá trình 600 xác định ở bước này lệnh đã xảy ra. Sau đó, quá trình 600 tiến hành sang bước 615 hoặc 625 theo yêu cầu.

Theo cách khác, nếu quá trình 600 xác định ở bước 640 rằng lệnh này được chọn ở bước 635 không bao gồm phép gán khác của lệnh hàm ngoài gián tiếp hoặc lệnh nạp trong mã nguồn A, quá trình 600 tiến hành bước 645 thay vì thực thi lệnh này được lựa chọn. Sau đó, quá trình 600 trở về bước 635 để chọn lệnh tiếp theo. Các bước trên Fig.6 tự lặp lại cho đến khi tất cả các lệnh này hoặc các lệnh trong mã nguồn A đã được thực thi. Sau đó, quá trình 600 kết thúc.

Fig.7 minh họa quá trình 700 mà được thực hiện bởi thiết bị trong suốt thời gian chạy của mã nguồn A trong suốt bước 645 ở quá trình 600. Quá trình 700 bắt đầu khi quá trình 600 cố gắng thực thi lệnh được chọn ở bước 645.

Sau đó, quá trình 700 xác định ở bước 705 nếu lệnh được chọn bao gồm hàm kiểm tra CFI. Nếu lệnh này không phải là hàm kiểm tra CFI, quá trình 700 trở về bước 645 để thực thi lệnh được chọn.

Theo cách khác, nếu quá trình 700 xác định ở bước 705 rằng lệnh được chọn là hàm kiểm tra CFI, thì quá trình 700 tiến hành bước 710. Ở bước 710, quá trình 700 trước hết nhận diện CFI được liên kết với hàm kiểm tra CFI. Sau đó, quá trình 700 kiểm tra liệu hàm được đề cập đến trong lời gọi hàm gián tiếp được chứa trong bảng hàm rút gọn được liên kết với mã nguồn A. Nếu quá trình 700 xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp được chứa trong bảng hàm rút gọn được liên kết với mã nguồn A, thì quá trình 700 sẽ thực thi lời gọi hàm gián tiếp liên kết như bình thường bằng cách trở về bước 645.

Ngược lại, nếu quá trình 700 xác định rằng hàm được đề cập đến trong lời gọi hàm gián tiếp không được chứa trong bảng hàm rút gọn được liên kết với mã nguồn A, thì quá trình 700 sẽ tiến hành bước 715. Ở bước 715, quá trình 700 sẽ cố gắng xác định liệu hàm kiểm tra CFI được liên kết với lời gọi hàm ngoài gián tiếp. Quá trình 700 thực hiện điều này bằng cách

cố gắng truy xuất địa chỉ được đề cập đến trong lời gọi hàm gián tiếp từ bản đồ ánh xạ. Nếu quá trình 700 không thể thu thập địa chỉ liên quan, quá trình 700 sẽ kết thúc do điều này nghĩa là vi phạm đã xuất hiện.

Theo phương án thực hiện, giả sử rằng lời gọi hàm gián tiếp chứa địa chỉ cho mã nguồn B. Nếu quá trình 700 có thể thu thập địa chỉ được đề cập đến trong lời gọi hàm gián tiếp, tức là, địa chỉ của mã nguồn B, điều này nghĩa lời gọi hàm gián tiếp là lời gọi hàm ngoài gián tiếp. Sau đó, quá trình 700 tiến hành bước 720. Ở bước này, sau đó quá trình 700 tận dụng địa chỉ của mã nguồn B để xác định liệu mã nguồn được liên kết với địa chỉ có liên quan đến mã tính toán kế thừa hay không. Nếu quá trình 700 xác định rằng mã nguồn B liên quan đến mã tính toán kế thừa, thì quá trình 700 sẽ thực thi lời gọi hàm ngoài gián tiếp như bình thường bằng cách trở về bước 645.

Tuy nhiên, nếu quá trình 700 xác định rằng mã nguồn B không liên quan đến mã tính toán kế thừa và thay vào đó được biên dịch theo các phương án thực hiện sáng chế, quá trình 700 sẽ kiểm tra RFT được liên kết với mã nguồn B để xác định nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong đó. Điều này diễn ra ở bước 725. Nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm rút gọn liên kết của mã nguồn B, thì lời gọi hàm ngoài gián tiếp được thực thi như bình thường bởi quá trình 700 bằng cách trở về bước 645.

Nếu không được chứa trong RFT liên kết của mã nguồn B, thì quá trình 700 tiến hành kiểm tra liệu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm thời gian chạy. Điều này diễn ra ở bước 730. Nếu hàm có thể được tìm thấy ở bảng hàm thời gian chạy, thì quá trình 700 sẽ tiến hành bước 645 để thực thi lời gọi hàm ngoài gián tiếp như bình thường. Tuy nhiên, nếu hàm được đề cập đến trong lời gọi

hàm ngoài gián tiếp không được chứa trong bảng hàm thời gian chạy, thì quá trình 700 hủy bỏ việc chạy mã nguồn A do điều này nghĩa là vi phạm đã xuất hiện và quá trình 700 kết thúc.

Theo các phương án thực hiện sáng chế, khi quá trình 700 đã xác định rằng lời gọi hàm gián tiếp là lời gọi hàm ngoài gián tiếp ở bước 715, quá trình 700 có thể tiến hành ngay bước 730 trong đó quá trình 700 tiến hành kiểm tra liệu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm thời gian chạy. Nếu hàm có thể được tìm thấy trong bảng hàm thời gian chạy, thì quá trình 700 sẽ tiến hành bước 645 để thực thi lời gọi hàm ngoài gián tiếp như bình thường. Tuy nhiên, nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp không được chứa trong bảng hàm thời gian chạy, thì quá trình 700 hủy bỏ chạy mã nguồn A do điều này nghĩa là vi phạm đã xuất hiện và quá trình 700 kết thúc. Theo phương án thực hiện sáng chế, các bước 720 và 725 có thể bị bỏ qua.

Theo các phương án thực hiện sáng chế, khi quá trình 700 đã xác định rằng lời gọi hàm gián tiếp là lời gọi hàm ngoài gián tiếp ở bước 715, quá trình 700 có thể tiến hành ngay bước 725 trong đó quá trình 700 sẽ kiểm tra bảng hàm rút gọn (RFT) được liên kết với mã nguồn B để xác định liệu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp có được chứa trong đó không. Điều này diễn ra ở bước 725. Nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong RFT liên kết của mã nguồn B, thì lời gọi hàm ngoài gián tiếp được thực thi như bình thường bởi quá trình 700 bằng cách trở về bước 645.

Nếu không, quá trình 700 chuyển sang bước 730 trong đó quá trình 700 tiến hành kiểm tra liệu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp được chứa trong bảng hàm thời gian chạy. Nếu hàm có thể được tìm thấy trong bảng hàm thời gian chạy, thì quá trình 700 sẽ tiến hành bước 645 để thực thi lời gọi hàm ngoài gián tiếp như bình thường. Tuy

nhiên, nếu hàm được đề cập đến trong lời gọi hàm ngoài gián tiếp không được chứa trong bảng hàm thời gian chạy, thì quá trình 700 hủy bỏ chạy mã nguồn A do điều này nghĩa là vi phạm đã xảy ra và quá trình 700 kết thúc. Theo phương án thực hiện sáng chế, bước 720 có thể bị bỏ qua.

Phần trên là mô tả các phương án thực hiện thiết bị và quá trình theo sáng chế như được nêu trong các điểm yêu cầu bảo hộ sau. Các phương án thực hiện khác có thể và sẽ thiết kế các tùy chọn sẽ nằm trong phạm vi của các điểm yêu cầu bảo hộ sau.

YÊU CẦU BẢO HỘ

1. Phương pháp củng cố tính toàn vẹn luồng điều khiển (control flow integrity, CFI) của ứng dụng phần mềm nhờ sử dụng thiết bị tính toán, trong đó phương pháp bao gồm bước:

thu được tệp tin nhị phân thứ nhất bằng cách biên dịch tệp tin mã nguồn thứ nhất được nhận bởi thiết bị, trong đó tệp tin mã nguồn thứ nhất bao gồm các hàm và lời gọi hàm gián tiếp, và trong đó tệp tin nhị phân thứ nhất bao gồm:

mã được tạo bằng cách biên dịch tệp tin mã nguồn thứ nhất đối với ứng dụng phần mềm, và

lời gọi hàm gián tiếp được biên dịch được kết xuất bằng cách biên dịch lời gọi hàm gián tiếp;

liên kết, trong quá trình biên dịch tệp tin mã nguồn thứ nhất, hàm kiểm tra CFI trong tệp tin nhị phân thứ nhất với lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân thứ nhất, sao cho trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi trước khi thực thi lời gọi hàm gián tiếp được biên dịch; và

bổ sung hàm, được viện dẫn trong lời gọi hàm gián tiếp được biên dịch, vào bảng hàm rút gọn (reduced function table, RTF), và

trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất; và

thực thi tệp tin nhị phân thứ nhất được biên dịch,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm các bước:

trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trongRTF; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân, và

tệp tin nhị phân thứ hai được xác định đã được biên dịch theo cách thức kế thừa.

2. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm:

thực thi tệp tin nhị phân thứ nhất, trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi nhiều lần, và trong đó trong suốt mỗi một lần trong số nhiều lần hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT; và

thực thi, theo xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT, lời gọi hàm gián tiếp được biên dịch.

3. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất, trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi nhiều lần, và trong đó trong suốt mỗi một lần trong số nhiều lần hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT; và

thực thi, theo xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT, lời gọi hàm gián tiếp được biên dịch.

4. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm bước trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT;

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân,

tệp tin nhị phân thứ hai được xác định là đã không được biên dịch theo cách thức kế thừa, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong RFT được liên kết với tệp tin nhị phân thứ hai.

5. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất; và

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm bước trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân,

tệp tin nhị phân thứ hai được xác định là đã không được biên dịch theo cách thức kế thừa,

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết không được chứa trong RFT được liên kết với tệp tin nhị phân thứ hai, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

6. Phương pháp theo điểm 1, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp in tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

7. Phương pháp theo điểm 1, trong đó bảng hàm thời gian chạy được lưu trữ trong vật ghi bất biến được bảo vệ có thể đọc được bằng bộ xử lý.

8. Phương pháp theo điểm 3, trong đó bản đồ nhị phân được lưu trữ trong vật ghi bất biến được bảo vệ có thể đọc được bằng bộ xử lý.

9. Phương pháp theo điểm 1, trong đó lời gọi hàm gián tiếp bao gồm lời gọi hàm gián tiếp được gán dưới dạng một phần của cấu trúc dữ liệu.

10. Phương pháp theo điểm 9, trong đó lời gọi hàm gián tiếp còn bao gồm lời gọi hàm gián tiếp được gán trực tiếp cho hàm.

11. Phương pháp theo điểm 10, trong đó lời gọi hàm gián tiếp còn bao gồm lời gọi hàm gián tiếp được gán dưới dạng lời gọi hàm hợp ngữ.

12. Thiết bị củng cố CFI của ứng dụng phần mềm bao gồm:

bộ phận biên dịch bao gồm:

bộ xử lý; và

vật ghi bất biến có thể đọc được bằng bộ xử lý, vật ghi bất biến lưu trữ các lệnh mà, khi được thực thi bằng bộ xử lý, tạo thuận lợi cho thiết bị thực hiện phương pháp bao gồm bước:

thu được tệp tin nhị phân thứ nhất bằng cách biên dịch tệp tin mã nguồn thứ nhất được nhận bởi thiết bị, trong đó tệp tin mã nguồn thứ nhất bao gồm các hàm và lời gọi hàm gián tiếp, và trong đó tệp tin nhị phân thứ nhất bao gồm:

mã được tạo bằng cách biên dịch tệp tin mã nguồn thứ nhất ứng dụng phần mềm, và

lời gọi hàm gián tiếp được biên dịch được kết xuất bằng cách biên dịch lời gọi hàm gián tiếp;

liên kết, trong quá trình biên dịch tệp tin mã nguồn thứ nhất, hàm kiểm tra CFI trong tệp tin nhị phân thứ nhất với lời gọi hàm gián tiếp được biên dịch trong tệp tin nhị phân thứ nhất, sao cho trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi trước khi thực thi lời gọi hàm gián tiếp được biên dịch; và

bổ sung hàm, được viện dẫn trong lời gọi hàm gián tiếp được biên dịch, vào RFT; và

bộ phận thực thi bao gồm:

bộ xử lý khác; và

vật ghi bất biến khác có thể đọc được bằng bộ xử lý khác, vật ghi bất biến khác lưu trữ các lệnh mà, khi được thực thi bằng bộ xử lý khác, tạo thuận tiện cho thiết bị khác thực hiện phương pháp bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất; và

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm bước:

trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân, và

tệp tin nhị phân thứ hai được xác định đã được biên dịch theo cách thức kế thừa.

13. Thiết bị theo điểm 12, trong đó phương pháp còn bao gồm các bước:

thực thi tệp tin nhị phân thứ nhất, trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi nhiều lần, và trong đó trong suốt mỗi một lần trong số nhiều lần hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT; và

thực thi, theo xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT, lời gọi hàm gián tiếp được biên dịch.

14. Thiết bị theo điểm 12, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất, trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi nhiều lần, và trong đó trong suốt mỗi một lần trong số nhiều lần hàm kiểm tra CFI được thực thi, hàm kiểm tra CFI xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT; và

thực thi, theo xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết được chứa trong RFT, lời gọi hàm gián tiếp được biên dịch.

15. Thiết bị theo điểm 12 trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp in tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân,

tệp tin nhị phân thứ hai được xác định là đã không được biên dịch theo cách thức kế thừa, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong RFT được liên kết với tệp tin nhị phân thứ hai.

16. Thiết bị theo điểm 12 trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp in tệp tin nhị phân thứ nhất; và

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân,

tệp tin nhị phân thứ hai được xác định là đã không được biên dịch theo cách thức kế thừa,

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết không được chứa trong RFT được liên kết với tệp tin nhị phân thứ hai, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

17. Thiết bị theo điểm 12, trong đó phương pháp còn bao gồm các bước:

nạp tệp tin nhị phân thứ nhất để thực thi;

cập nhật bản đồ nhị phân có địa chỉ được viện dẫn trong lệnh nạp theo bước thực thi lệnh nạp trong tệp tin nhị phân thứ nhất;

cập nhật bảng hàm thời gian chạy với hàm được viện dẫn trong lời gọi hàm bên ngoài gián tiếp theo bước thực thi lệnh gán lời gọi hàm bên ngoài gián tiếp trong tệp tin nhị phân thứ nhất;

thực thi tệp tin nhị phân thứ nhất,

trong đó trong khi thực thi tệp tin nhị phân thứ nhất, hàm kiểm tra CFI được thực thi,

trong đó phương pháp còn bao gồm trước hết xác định, theo bước thực thi hàm kiểm tra CFI, hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT; và

thực thi, theo bước trước hết xác định hàm được viện dẫn trong lời gọi hàm gián tiếp được liên kết không được chứa trong RFT, lời gọi hàm gián tiếp liên kết, sau khi xác định thêm:

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong tệp tin nhị phân thứ hai có địa chỉ được chứa trong bản đồ nhị phân, và

hàm được viện dẫn trong lời gọi hàm gián tiếp liên kết được chứa trong bảng hàm thời gian chạy.

18. Thiết bị theo điểm 12 trong đó RFT được lưu trữ trong vật ghi bất biến được bảo vệ có thể đọc được bằng bộ xử lý.

19. Thiết bị theo điểm 14 trong đó bản đồ nhị phân được lưu trữ trong vật ghi bất biến được bảo vệ có thể đọc được bằng bộ xử lý.

20. Thiết bị theo điểm 12 trong đó lời gọi hàm gián tiếp bao gồm lời gọi hàm gián tiếp được gán dưới dạng một phần của cấu trúc dữ liệu.

21. Thiết bị theo điểm 20 trong đó lời gọi hàm gián tiếp còn bao gồm lời gọi hàm gián tiếp được gán trực tiếp cho hàm.

22. Thiết bị theo điểm 21 trong đó lời gọi hàm gián tiếp còn bao gồm lời gọi hàm gián tiếp được gán dưới dạng lời gọi hàm hợp ngữ.

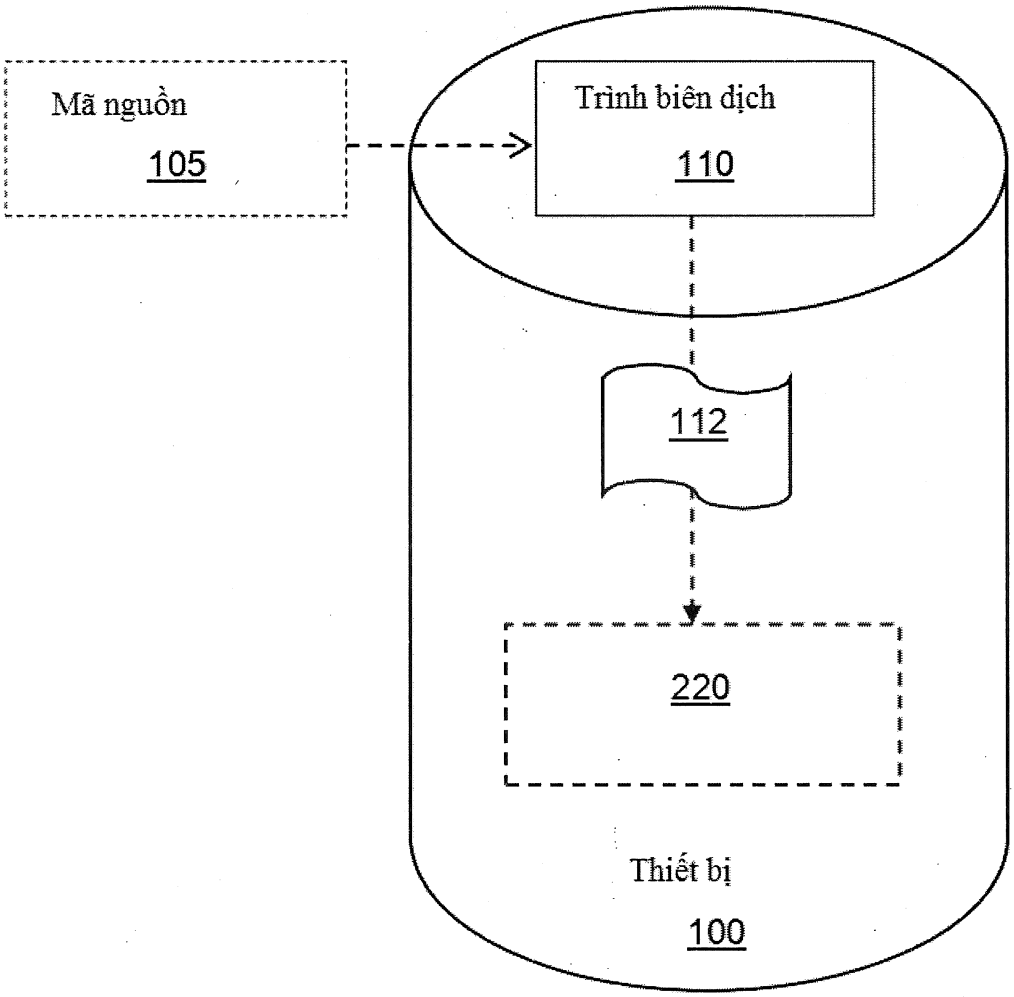


Fig.1

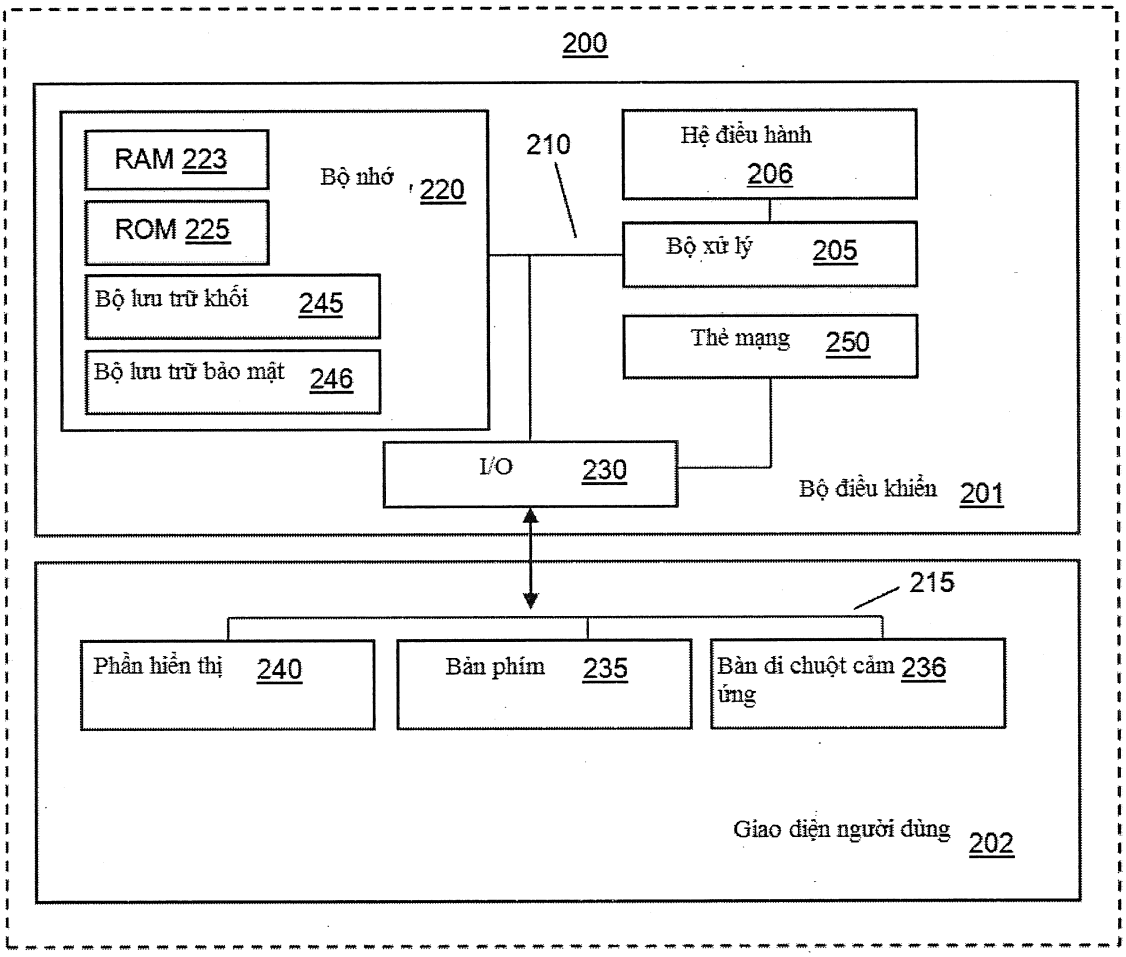

200
RAM 223
Bộ nhớ 220
ROM 225
Bộ lưu trữ khối 245
Bộ lưu trữ bảo mật 246
210
Hệ điều hành 206
Bộ xử lý 205
Thẻ mạng 250
I/O 230
Bộ điều khiển 201
215
Phần hiển thị 240
Bàn phím 235
Bàn di chuột cảm ứng 236
Giao diện người dùng 202

Fig.2

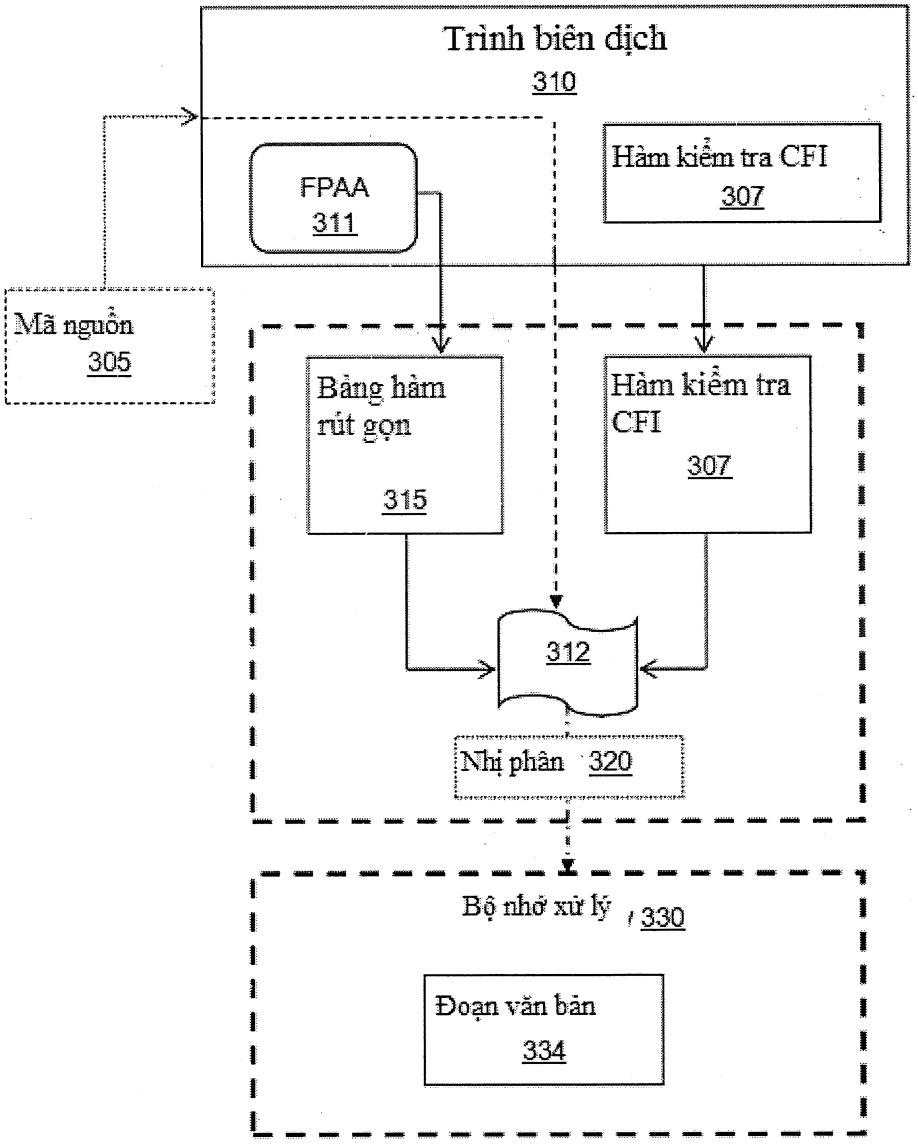

Trình biên dịch
310
FPAA
311
Hàm kiểm tra CFI
307
Mã nguồn
305
Bảng hàm rút gọn
315
Hàm kiểm tra CFI
307
312
Nhị phân 320
Bộ nhớ xử lý 330
Đoạn văn bản
334

Fig.3

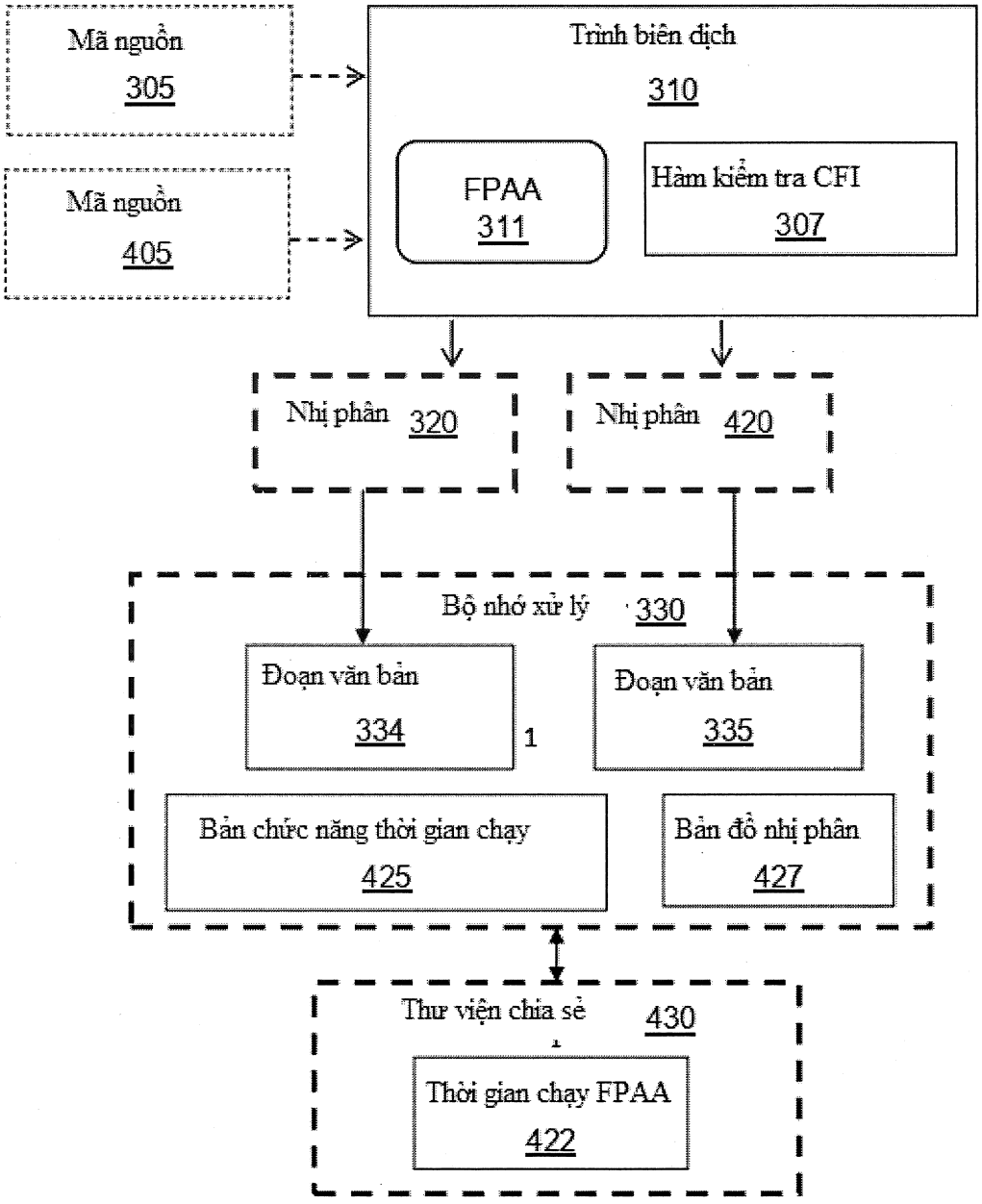

Mã nguồn
305
Mã nguồn
405
Trình biên dịch
310
FPAA
311
Hàm kiểm tra CFI
307
Nhị phân 320
Nhị phân 420
Bộ nhớ xử lý 330
Đoạn văn bản
334
1
Đoạn văn bản
335
Bản chức năng thời gian chạy
425
Bản đồ nhị phân
427
Thư viện chia sẻ 430
Thời gian chạy FPAA
422

Fig.4

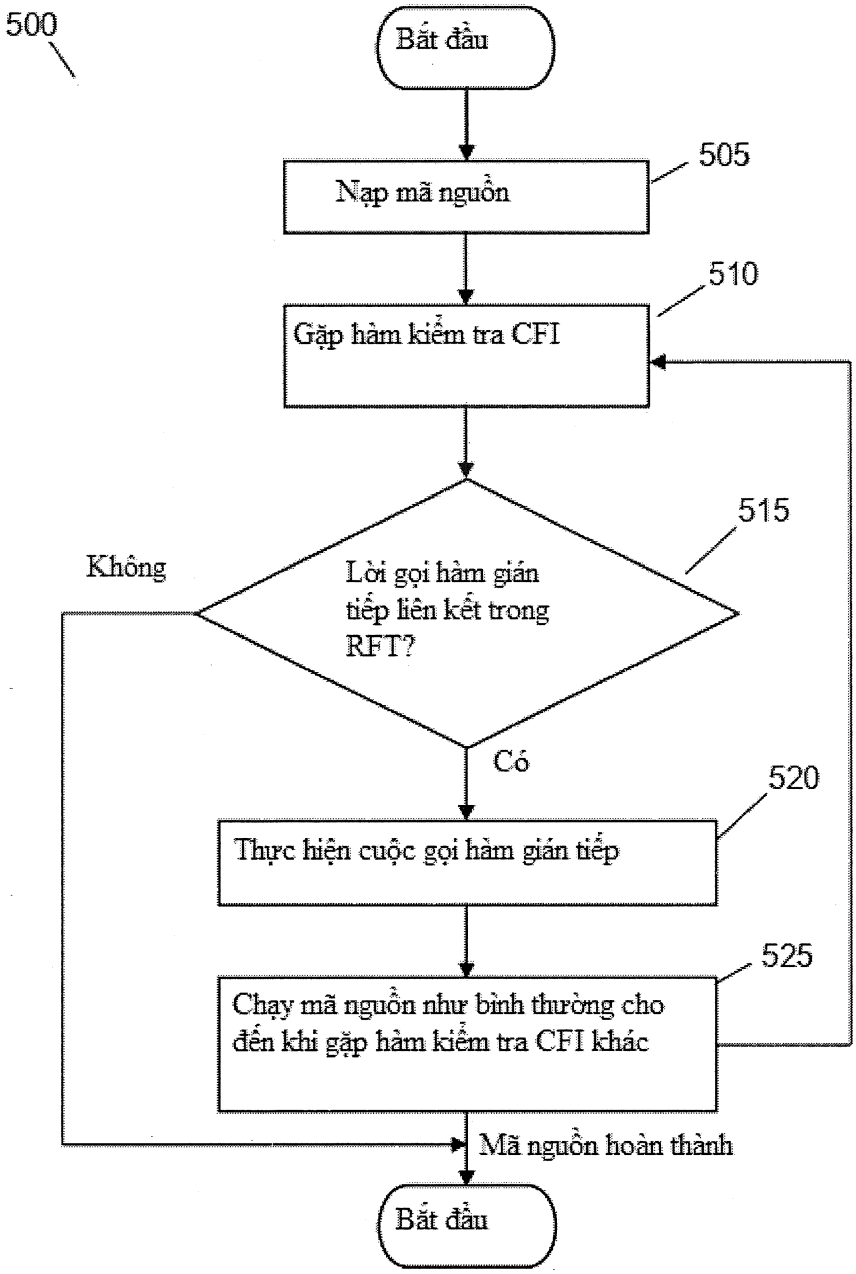

500
Bắt đầu
505
Nạp mã nguồn
510
Gặp hàm kiểm tra CFI
515
Lời gọi hàm gián tiếp liên kết trong RFT?
Không
Có
520
Thực hiện cuộc gọi hàm gián tiếp
525
Chạy mã nguồn như bình thường cho đến khi gặp hàm kiểm tra CFI khác
Mã nguồn hoàn thành
Bắt đầu

Fig.5

6/7

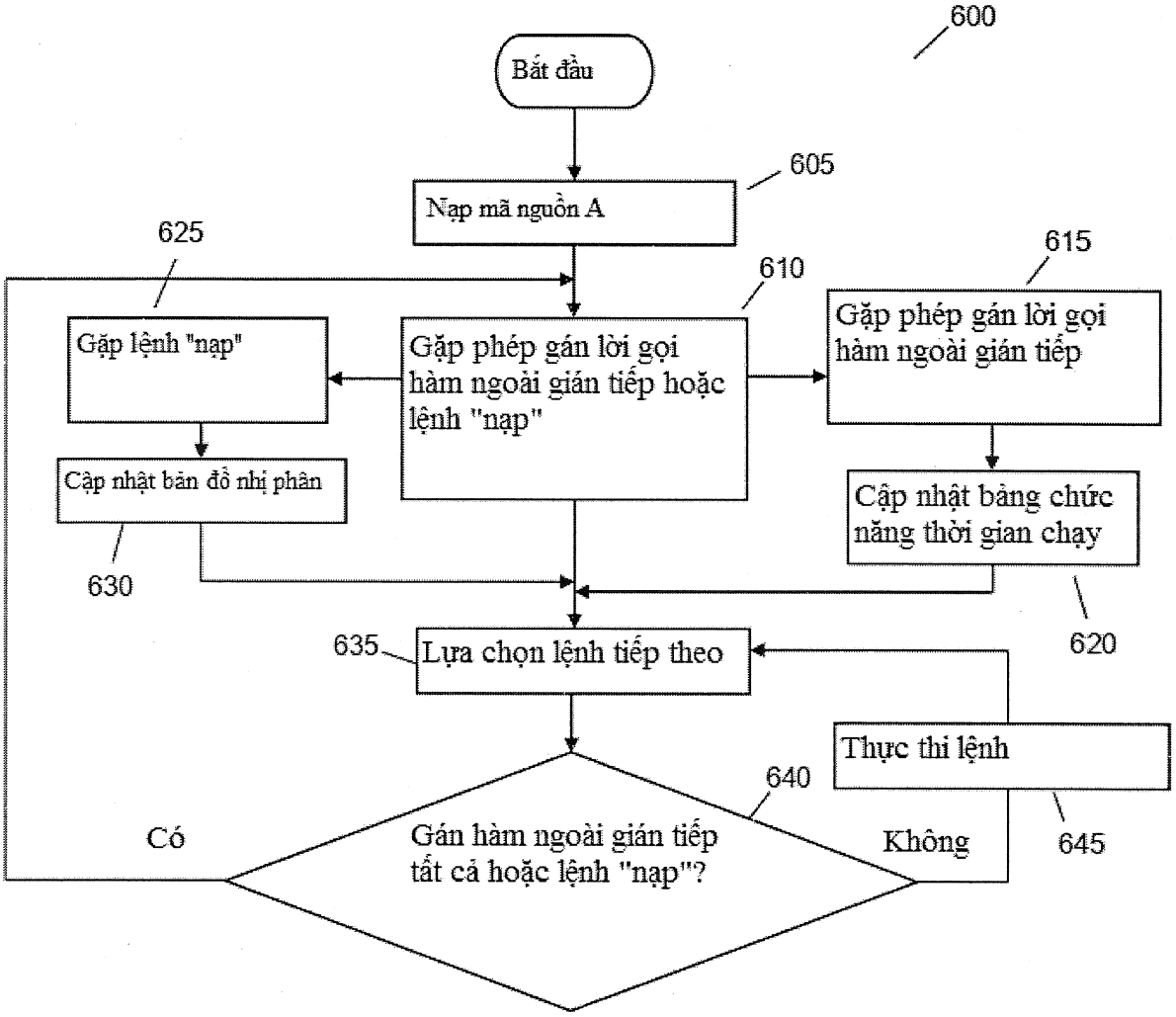

600
Bắt đầu
605
Nạp mã nguồn A
625
610
615
Gặp lệnh "nạp"
Gặp phép gán lời gọi hàm ngoài gián tiếp hoặc lệnh "nạp"
Gặp phép gán lời gọi hàm ngoài gián tiếp
Cập nhật bản đồ nhị phân
Cập nhật bảng chức năng thời gian chạy
630
620
635
Lựa chọn lệnh tiếp theo
Thực thi lệnh
640
Có
Gán hàm ngoài gián tiếp tất cả hoặc lệnh "nạp"?
Không
645

Fig.6

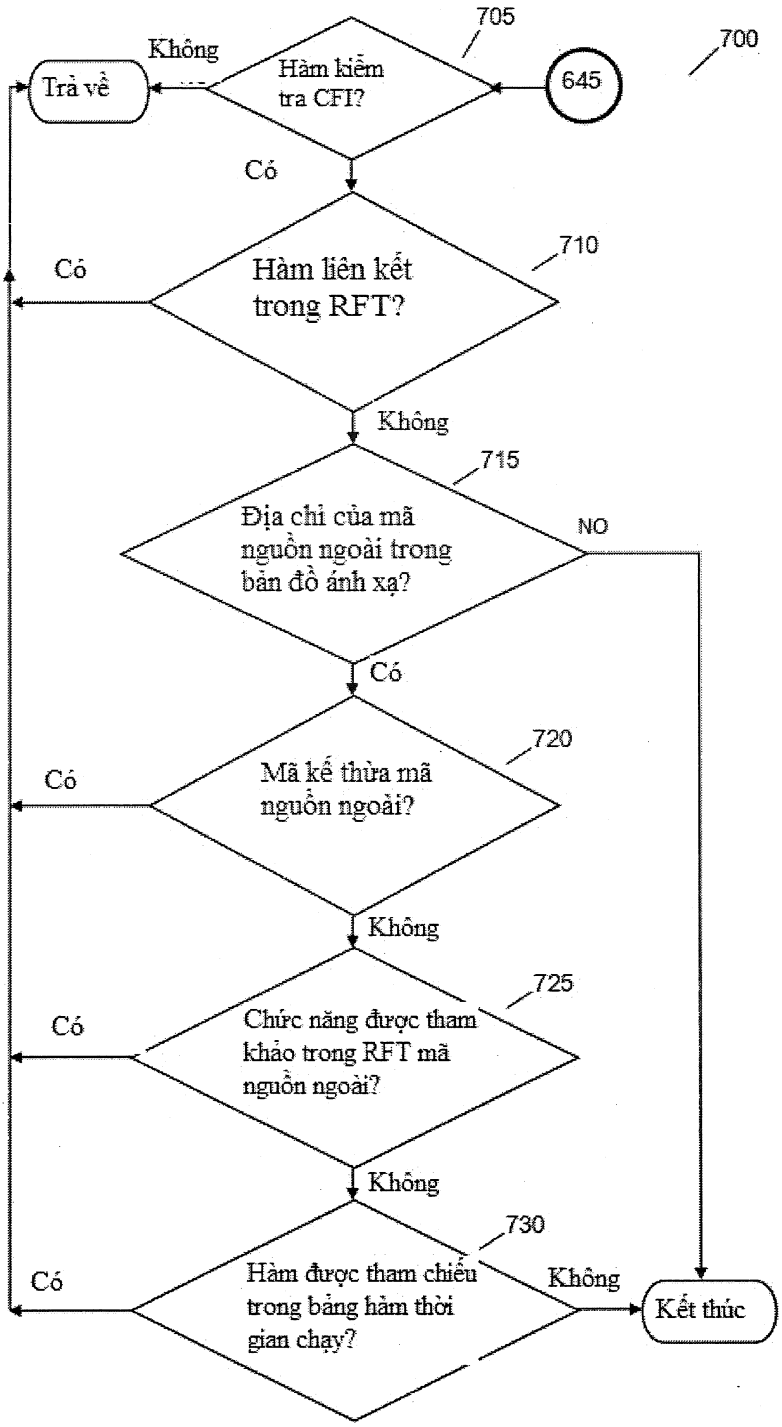

705
700
Không
Trả về
Hàm kiểm tra CFI?
645
Có
710
Có
Hàm liên kết trong RFT?
Không
715
Địa chỉ của mã nguồn ngoài trong bản đồ ánh xạ?
NO
Có
720
Có
Mã kế thừa mã nguồn ngoài?
Không
725
Có
Chức năng được tham khảo trong RFT mã nguồn ngoài?
Không
730
Có
Hàm được tham chiếu trong bảng hàm thời gian chạy?
Không
Kết thúc

Fig.7