

METHOD FOR OBFUSCATION OF CODE USING RETURN ORIENTED PROGRAMMING

Field of Invention

5 Embodiments of the invention relate to code protection and, more particularly, applying Return Oriented Programming (ROP) to obfuscate code and for non-malicious purpose.

Background

10 Obfuscation is an important technique to protect algorithms and code from disclosure. Application developers often use this technique to protect critical algorithms in a program so that the source code and machine code of the program are difficult to be understood.

15 One existing obfuscation technique involves increasing difficulty of disassembling program code so that only a small portion of the program code will be disassembled. Another existing obfuscation technique involves applying encryption to selected instructions such that the encrypted instructions can be disclosed only with a key.

20

 These existing obfuscation techniques, however, pose limitations. With the first technique, code analyzers would realise that only a small portion of program

code is disassembled. With the second technique, one leaves suspicion to program analysis and attracts attention during encryption and decryption.

Summary

5 Embodiments of the invention relates to utilization of Return Oriented Programming (ROP) for a non-malicious purpose i.e. code obfuscation to improve security. Code that is associated with a designated function of sensitive nature may be hidden by creating a ROP code that performs the designated function of the original (unobfuscated) code and constructing a ROP payload that can be loaded
10 to execute the ROP code. The original code may be replaced by a spurious code unrelated to the designated function, and a control flow instruction is provided to load the ROP payload to execute the ROP code.

According to one aspect of the invention, a method of executing obfuscated code is provided. The method comprises:

15 in a mobile device having a ROP-embedded host program which includes a ROP code configured to perform a designated function associated with a guest code which is omitted from the ROP-embedded host program, in response to invoking the designated function, retrieving a ROP payload; and

20 resetting a memory register of the mobile device with the ROP payload to execute the ROP code which performs the invoked function associated with the guest code.

In one embodiment of the above method, retrieving a ROP payload includes sending a request from the mobile device to a remote server to obtain data associated with the invoked function, and receiving both the requested data and the ROP payload from the server.

5 In another embodiment of the above method, retrieving a ROP payload includes retrieving the ROP payload from a memory device in the mobile device.

According to another aspect of the invention, a code obfuscation method is provided. The method comprises:

10 from a host program code having a guest code, identifying a plurality of gadgets related to the guest code and ascertaining a plurality of return addresses corresponding to the gadgets;

 creating a ROP code that connects the gadgets, by modifying the return addresses which are to be stored in a memory register, wherein the ROP code is to perform a designated function of the guest code;

15 constructing a ROP payload which includes parameter data required by the guest code, the return addresses of the gadgets, and a data segment configured to reset the memory register to execute the ROP code;

 replacing the guest code in the host program code with a spurious code;

20 providing a command code configured to reset the memory register with the ROP payload; and

storing the ROP payload at a memory device, wherein the ROP payload is configured to be invoked at runtime to execute the ROP code to perform the designated function associated with the guest code.

5 In one embodiment of the above method, the memory device is located at a server computer which is remote from a mobile device to be installed with a ROP-embedded host program which includes the ROP code.

In another embodiment of the above method, the memory device is located at a mobile device to be installed with a ROP-embedded host program which includes the ROP code.

10 According to another aspect of the invention, a mobile device is provided. The mobile device comprises:

a processor, and a memory device having a ROP-embedded host program which includes a ROP code configured to perform a designated function associated with a guest code which is omitted from the ROP-embedded host program, the processor is configured to:

15

retrieve a ROP payload in response to invoking the designated function; and

reset a memory register of the mobile device with the ROP payload to execute the ROP code which is to perform the invoked function associated with the guest code.

20

In one embodiment of the above device, the processor is configured to retrieve the ROP payload by sending a request to a remote server to obtain data

associated with the invoked function, and receiving both the requested data and the ROP payload from the server.

In another embodiment of the above device, the processor is configured to retrieve the ROP payload by retrieving the ROP payload from the memory device
5 in the mobile device.

In another embodiment of the above device, the processor is configured to perform authorization for retrieval of the ROP payload by verifying an external input prior to retrieving the ROP payload.

10 **Brief Description of the Drawings**

The invention will be described in detail with reference to the accompanying drawings, in which:

Figure 1A shows a host program (e.g. music player);

Figure 1B shows guest code (e.g. licence verification code) to be obfuscated;

15 Figure 1C shows the guest code of Figure 1B after code obfuscation;

Figure 2 illustrates a code obfuscation method according to one embodiment of the invention; and

Figure 3 illustrates a method for executing an obfuscated guest code according to one embodiment of the invention.

20

Detailed Description

In the following description, numerous specific details are set forth in order to provide a thorough understanding of various illustrative embodiments of the invention. It will be understood, however, to one skilled in the art, that 5 embodiments of the invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure pertinent aspects of embodiments being described. In the drawings, like reference numerals refer to same or similar functionalities or features throughout the several views.

10 Embodiments of the invention disclose a new obfuscation technique using Return Oriented Programming (ROP).

ROP has become one of the most powerful runtime attack techniques recently. An attacker using ROP typically, as a first step, identifies gadgets from valid code sequences. Gadgets may be referred to as small snippets of valid code 15 sequences, and more particularly to instruction sequences starting potentially at the middle of an existing machine instruction and ending with a return instruction. The identified gadgets provide the basic units to form a ROP program. In a second step, control flow of the program is changed to the address of the first gadget in a ROP program. Since every gadget ends with a return instruction, when the first 20 gadget returns, the program can “return” to the second gadget with such address being carefully prepared on the register. If a gadget contains instructions like pop, which will get parameters from the register, the parameter should be also arranged on the register just after the return address of that gadget. In sum, ROP essentially

gathers snippets of valid code from existing instructions in a program to form another program that performs a specific function and changes control flow of the original program.

5 ROP is traditionally used for malicious attacks on vulnerable programs. In contradistinction, embodiments of the invention employ ROP for a non-malicious purpose—to protect and hide program code, and by embedding ROP during application development.

10 Figures 1A to 1C illustrate a non-limiting example for which the invention may be applied. Figure 1A shows a host program (e.g. music player) having guest code (e.g. licence verification code) as shown in Figure 1B. The licence verification code is to be obfuscated so that an adversary is unable to, for example, forge a new licence file by reverse engineering. Figure 1C shows the guest code of
15 Figure 1B after code obfuscation.

Figure 2 illustrates a code obfuscation method 200. The method 200 will be described with reference to the example in Figures 1A to 1C, where the host program is Android-based and written using native code (C/C++).

20

In **block 202**, a guest code within a host program is identified for code obfuscation. Here, the guest code is unobfuscated and the host program is in non-ROP form.

In **block 204**, the host program code and native libraries are analysed to identify useful gadgets (ROP gadgets) related to the guest code. Return addresses corresponding to the identified gadgets are also ascertained.

To locate these gadgets, a semi-automated tool that supports both .so files
5 and apk files on ARM architecture may be developed. The tool is configured to analyze the host program (e.g. Android application), native libraries in the Android system, as well as the libraries in Android packages, and ascertain return addresses corresponding to the identified gadgets.

In **block 206**, the identified gadgets are chained or connected together to
10 create a ROP code. This is achieved by modifying the return addresses (as ascertained in block 204) that are to be stored in a memory register. The ROP code, upon execution in due course, is to perform a designated function associated with the guest code.

In **block 208**, a ROP payload is constructed which is configured to change
15 control flow of the host program and thereby execute the obfuscated guest code (ROP code).

To change control flow, the *setjmp()* and *longjmp()* subroutines, as defined in the C standard library to provide non-local jumps, may be used. *Setjmp()* is to save the calling environment or contents of the memory registers, as defined by
20 *jmp_buf*, so that *longjmp()* can restore them. In this way, *longjmp()* "returns" to the state of the program when *setjmp()* was called. Accordingly, *jmp_buf* holds the information required to restore a calling environment.

As the *jmp_buf* saved by *setjmp()* would be changed by a heap overflow vulnerability, the ROP payload should contain (i) parameter data required by the guest code, (ii) return addresses of the identified gadgets (as ascertained earlier in block 204) and (iii) a data segment configured to reset the memory register to
5 execute the ROP code.

In **block 210**, the guest code is removed from the host program code. A spurious code is provided or embedded to replace the guest code. Following the spurious code, a command code configured to reset the memory register with the ROP payload is provided. Particularly, a *longjmp()* subroutine is provided to reset
10 the calling environment and provide a mechanism to change the return address of the host program.

In **block 212**, the APK file containing the host program code including the spurious code and ROP code is repackaged and signed. The APK file is accordingly distributed and installed as ROP-embedded host program in various
15 mobile devices. Such mobile devices are known in the art, and at least comprise a processor, and a memory device for containing instructions executable by the processor to result in the mobile device performing various operations.

In **block 214**, the ROP payload is stored in a memory device to be invoked at runtime to execute the ROP code which would perform the designated function
20 associated with the guest code. In one embodiment, the ROP payload is stored in a memory device at a remote server computer. In another embodiment, the ROP payload is stored in a memory device of the mobile device.

Following from the above, Figure 3 illustrates a method 300 for executing an obfuscated guest code.

In **block 302**, a mobile device installed with a ROP-embedded host program is provided. Particularly, in the ROP-embedded host program, pre-selected guest
5 code has been obfuscated in the form of ROP code. When a designated function (e.g. licence verification function as in Figure 1A) associated with the obfuscated guest code is invoked, the mobile device, in response, sends a request to a remote server computer for data associated with the invoked function. The mobile device also executes the spurious code which is generally unrelated to the invoked
10 function.

In **block 304**, the remote server responds with data normally associated with the invoked function. The ROP payload is also retrieved from either the remote server or the mobile device installed with the ROP-embedded host program.

In one embodiment where the ROP payload is stored in a memory device at
15 the remote server, the remote server additionally responds with the ROP payload which was pre-stored in the remove server.

In another embodiment where the ROP payload is stored in a memory device at the mobile device, the ROP payload pre-stored in the mobile device is retrieved upon invoking the function. In this embodiment, invoking the obfuscated
20 guest code may be either automatic (unconditional) or with an external input preconfigured to trigger execution of the obfuscated guest code (conditional). For example, prior to retrieving the ROP payload from the memory device,

authorization of ROP payload retrieval is performed by verifying an external input which may be received from a user of the mobile device.

In **block 306**, the retrieved ROP payload is applied to reset the memory register of the mobile device. Accordingly, the original calling environment is restored and control flow returns to the host program to execute the ROP code to perform the designated function associated with the obfuscated guest code.

The foregoing disclosure is described with reference to Android-based application. It is to be appreciated that embodiments of the invention are suitable to be applied to other platforms including x86 and SPARC.

As will be appreciated from the above, embodiments of the invention overcome the limitations of existing obfuscation techniques, and provide several advantages and distinctions, including but not limited to the following:

- By using ROP, the code to be obfuscated is in ROP form when embedded in the host program. The ROP-embedded host program can be fully and successfully disassembled, and therefore no suspicion of code obfuscation will be raised.
- Control flow of the host program is changed by overwriting return addresses in the host program. With the use of ROP technique, static analysis is not able to detect the return addresses of a program (a runtime concept). At runtime, some of the return addresses would be changed to the addresses of ROP gadgets.

- In certain embodiments, a remote server is used to control behaviour after invoking a pre-selected program function. Particularly, the server sends ROP payloads that contain return addresses of the ROP gadgets to the mobile device. The payload is not accessible using static analysis of the host program. Thus, the ROP payload controls the invoking of the obfuscated guest code at runtime.
- Android applications are changed using native code, particularly to embed a mechanism to modify *jmp_buf* or the return addresses.
- ROP is now applied to a non-malicious purpose and in fact is to improve security. As the ROP takes place during program development, the program is under full control of the developer who is also the ROP constructor. Therefore, the developer can modify any aspect of the program to make ROP embedding possible. Whereas existing ROP techniques are applied to malicious attacks where the ROP constructor cannot change instructions in the program.
- A semi-auto tool for analyzing gadgets in Android applications may be provided that supports apk files and libraries that would be used by the applications, including native libraries in Android system and the apk files.

Other embodiments will be apparent to those skilled in the art from consideration of the specification and practice of the invention. Furthermore, certain terminology has been used for the purposes of descriptive clarity, and not to limit the disclosed embodiments of the invention. The embodiments and features described above should be considered exemplary.

Claims

1. A method of executing obfuscated code, the method comprising:

in a mobile device having a ROP (Return Oriented Programming)-
embedded host program which includes a ROP code configured to perform a
5 designated function associated with a guest code which is omitted from the ROP-
embedded host program, in response to invoking the designated function,
retrieving a ROP payload; and

resetting a memory register of the mobile device with the ROP payload to
execute the ROP code which performs the invoked function associated with the
10 guest code.

2. The method of claim 1, wherein retrieving a ROP payload includes sending a
request from the mobile device to a remote server to obtain data associated with
the invoked function, and receiving both the requested data and the ROP payload
15 from the server.

3. The method of claim 1, wherein retrieving a ROP payload includes retrieving the
ROP payload from a memory device in the mobile device.

4. The method of claim 3, further comprising:

prior to retrieving the ROP payload from the memory device in the mobile device, performing authorization for retrieval of the ROP payload by verifying an external input.

5

5. The method of claim 1, wherein the ROP-embedded host program is an Android application.

6. The method of claim 1, wherein the ROP payload includes parameter data
10 required by the guest code, return addresses of a plurality of gadgets which form the ROP code, and a data segment configured to reset the memory register to execute the ROP code.

7. The method of any of claims 1 to 5, wherein the ROP-embedded program is
15 prepared by:

from a non-ROP host program code having the guest code,
identifying a plurality of gadgets related to the guest code and ascertaining a
plurality of return addresses corresponding to the gadgets;

creating the ROP code that connects the gadgets, by modifying the
20 return addresses which are to be stored in a memory register;

constructing the ROP payload which includes parameter data required by the guest code, the return addresses of the gadgets, and a data segment configured to reset the memory register to execute the ROP code;

replacing the guest code in the host program code with a spurious code;

providing a command code configured to reset the memory register with the ROP payload; and

storing the ROP payload.

8. A code obfuscation method comprising:

from a host program code having a guest code, identifying a plurality of gadgets related to the guest code and ascertaining a plurality of return addresses corresponding to the gadgets;

creating a ROP code that connects the gadgets, by modifying the return addresses which are to be stored in a memory register, wherein the ROP code is to perform a designated function of the guest code;

constructing a ROP payload which includes parameter data required by the guest code, the return addresses of the gadgets, and a data segment configured to reset the memory register to execute the ROP code;

replacing the guest code in the host program code with a spurious code;

providing a command code configured to reset the memory register with the ROP payload; and

storing the ROP payload at a memory device, wherein the ROP payload is configured to be invoked at runtime to execute the ROP code to perform the
5 designated function associated with the guest code.

9. The method of claim 8, wherein the memory device is located at a server computer which is remote from a mobile device to be installed with a ROP-embedded host program which includes the ROP code.

10

10. The method of claim 8, wherein the memory device is located at a mobile device to be installed with a ROP-embedded host program which includes the ROP code.

15 11. The method of claim 8, wherein the host program is an Android application.

12. A mobile device comprising:

a processor, and a memory device having a ROP-embedded host program which includes a ROP code configured to perform a designated function
20 associated with a guest code which is omitted from the ROP-embedded host program, the processor is configured to:

retrieve a ROP payload in response to invoking the designated function; and
reset a memory register of the mobile device with the ROP payload to
execute the ROP code which is to perform the invoked function associated with the
guest code.

5

13. The device of claim 12, wherein the processor is configured to retrieve the
ROP payload by sending a request to a remote server to obtain data associated
with the invoked function, and receiving both the requested data and the ROP
payload from the server.

10

14. The device of claim 12, wherein the processor is configured to retrieve the
ROP payload by retrieving the ROP payload from the memory device in the mobile
device.

15 15. The device of claim 14, wherein the processor is configured to perform
authorization for retrieval of the ROP payload by verifying an external input prior to
retrieving the ROP payload.

16. The device of claim 12, wherein the ROP-embedded host program is an
20 Android application.

17. The device of claim 12, wherein the ROP payload includes parameter data required by the guest code, return addresses of a plurality of gadgets which form the ROP code, and a data segment configured to reset the memory register to execute the ROP code.

5

18. The device of any of claims 12 to 16, wherein the ROP-embedded host program is prepared by:

from a non-ROP host program code having the guest code,
identifying a plurality of gadgets related to the guest code and ascertaining a
10 plurality of return addresses corresponding to the gadgets;

creating the ROP code that connects the gadgets, by modifying the
return addresses which are to be stored in a memory register;

constructing the ROP payload which includes parameter data
required by the guest code, the return addresses of the gadgets, and a data
15 segment configured to reset the memory register to execute the ROP code;

replacing the guest code in the host program code with a spurious
code;

providing a command code configured to reset the memory register
with the ROP payload; and

20 storing the ROP payload.