Simplified Control Flow Integrity Method for Permutated Programs

Kai Lehniger[1], Mario Schölzel[2], Peter Tabatt[2], Marcin J. Aftowicz[1]
Peter Langendörfer[1,3]

[1]IHP - Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany
{lehniger, aftowicz, langendoerfer}@ihp-microelectronics.com

[2]Hochschule Nordhausen
Nordhausen, Germany
{peter.tabatt, mario.schoelzel}@hs-nordhausen.de

[3]BTU Cottbus-Senftenberg
Cottbus, Germany
peter.langendoerfer@b-tu.de

32nd Crypto Day, 15 January 2021

Control Flow Integrity (CFI) methods are part of research for over a decade now. Limiting a program to its intended control flow is a challenging task but also eliminates possibilities of code reuse attacks. There are many approaches, a lot of which include additional data and checking code. One of them was described by Abadi, Budiu, Erlingsson & Ligatti (2009). The main idea is to insert arbitrary but unique byte sequences, called signatures, at jump targets into the program code, as well as a checking code before each jump. The checking code compares the signature at the target address with its own expected signature. The jump only happens if both signatures matches. The need for signatures to be unique comes from the fact that duplicated signatures could be exploited and lead to unwanted control flows, since signatures of checking code and jump targets would match that are not part of the intended control flow.
When the program itself is not known to the attacker there are still methods for exploitation, i.e by leaking information (Bittau, Belay, Mashtizadeh, Mazières & Boneh (2014)) or brute forcing (Goodspeed & Francillon (2009)). However, in that case CFI is even harder to overcome.
We present a simplified approach of signature based CFI that works as protection against return-oriented programming (ROP), with the precondition that the attacker does not know the positions of the gadgets. This is achieved by permutation of the program code itself on a regular basis.
Permutation forces an attacker to start guessing for concrete addresses to jump to, making it impossible to reliably change the control flow. As a result it is no longer possible for an attacker to exploit duplicated addresses, making the uniqueness of signatures is not important anymore. There-fore we settle with

crypto day matters 32 (2021)
Beiträge vom 32. Kryptotag, 15.01.2021, online.
doi:10.18420/cdm-2021-32-21

the same signature for all potential jump targets. This has the advantage that we do not need complicated control-flow analysis and can use the same checking code for all effected jumps.
Also, for now we only focus on valid targets for function returns. This way we can make use of the fact that the return address is an address after a corresponding call instruction. Instead of inserting a signature inside the program, we can use the call instruction itself as a signature.
In result we have a simple checking method that can be inserted into any existing binary by replacing every return instruction with a jump to the checking code. With that we can guarantee that each return jumps to a position after a call instruction. Of course architectural particularities can introduce additional obstacles to overcome. With ARM and Xtensa we will show two such architectures and present possible solutions for an implementation.

Acknowledgment

This work was supported by the Federal Ministry of Education and Research (BMBF) under research grant number 01IS18065E.

References

Martín Abadi, Mihai Budiu, Úlfar Erlingsson & Jay Ligatti (2009). Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information and System Security (TISSEC)* **13**(1), 1–40.

Andrea Bittau, Adam Belay, Ali Mashtizadeh, David Mazières & Dan Boneh (2014). Hacking blind. In *2014 IEEE Symposium on Security and Privacy*, 227–242. IEEE.

Travis Goodspeed & Aurélien Francillon (2009). Half-blind attacks: mask ROM bootloaders are dangerous. In *Proceedings of the 3rd USENIX conference on Offensive technologies. USENIX Association*, 6–6.