Fakultät für Ingenieurwissenschaften

Exposé zur Master-Thesis

Automating ROP and JOP Chain Generation aka ROP Compiler

eingereicht von:	David Schunke
	Studiengang IT-Sicherheit und Forensik
Matrikelnummer:	417785

Inhaltsverzeichnis

1 Problemstellung und Motivation

Return-oriented-programming (ROP) und *Jump-oriented-programming* Angriffstechniken, welche genutzt werde, um Schadcode trotz vorhandener Sicherheitsmechanismen ausführen zu können. Dies erfordert lediglich ein Zielprogramm, welches einen User Input zulässt. Hierzu werden die im Zielprogramm vorhandenen Programmanweisungen, welche sich bereits im Program Memory befinden, genutzt, um sogenannte *ROP Gadgets* zu bilden.
Ein Gadget ist ein kleiner Code Teil, welcher eine bestimmte Aufgabe durchführt, z.B. das Kopieren eines Speicherregisters in ein Anderes oder die Addition zweier Register. ROP oder JOP Gadgets enden typischerweise mit einer *return*, *jump* oder *call* Anweisung, welche die Ausführung zu einer vorherig gespeicherten Stelle zurückbringen soll. Durch Manipulation dieser Return-Address erlaubt es dem Angreifer Gadgets zu verketten, um eigene Operationen durchzuführen und ein gewünschtes Ergebnis zu erzielen. Da lediglich Maschineninstruktionen genutzt werden, welche sich bereits im Speicher des Programms bzw. einer Library befindet, können viele Sicherheitsmechanismen, wie z.B. NX oder W^X, umgangen werden.

Die Entwicklung von solchen Exploits erfordert es derzeit noch manuell solche ROP Gadget Chains sowie den passenden Program Input zu finden und zu verketten. Diese Prozesse durchzuführen ist nicht nur aus Angreifersicht von Interesse, sondern auch aus Sicht von Security Researchers (White Heads / Blue Teams) wichtig, um mögliche Exploits finden und anschließend beheben bzw. melden zu können.

Das manuelle Finden dieser ROP Chains ist hoch zeitintensiv und erfordert tiefgreifende Kenntnisse der zugrundeliegenden Hardware, Architektur, des Zielprogramms sowie der Funktionsweise von Maschinenprozessen und Speicherverwaltung. An den Security Researcher werden somit hohe Anforderungen gestellt, welche das eigentliche Auffinden von Schwachstellen erschwert.
Durch die Erstellung eines *ROP Compilers* könnten diese Prozess schrittweise automatisiert werden. Dies würde den zeitlichen Aufwand verringern sowie die Möglichkeit der Suche von Schwachstellen an eine breitere Masse von Personen geben, um so gefundene Schwachstellen schneller finden und beheben zu können.

2 Ziele und Grenzen

2.1 Zielsetzung

Im Rahmen dieser Arbeit soll ein ROP Compiler entwickelt werden, welcher als Eingaben auf zur Verfügung stehender Gadgets sowie vordefiniertem Code basierend eine Verkettung der Gadgets automatisiert durchführen soll. Die zur Verkettung benötigte Semantik sowie die zur Eingabe benötigten Formatregeln sollen ebenfalls innerhalb dieser Arbeit definiert und implementiert werden.

Ziel ist die Erstellung eines Proof-of-concept (PoC), welcher auf Basis einer zuvor ausgewählten spezifischen Prozessorarchitektur (X64, ARM32 oder ARM64) die grundsätzliche Umsetzbarkeit demonstrieren und Ausgangspunkt für weitere Arbeiten bilden soll.

Außerdem soll der Grad der Berechenbarkeit erörtert und gezeigt werden, dass - abhängig vom zugrundeliegenden Zielprogramms, welches exploited werden soll - eine Touring-Vollständigkeit nicht grundsätzlich vorliegt.

Anhand verschiedener konstruierter sowie realer Beispiele soll der PoC evaluiert und die Umsetzbarkeit eines ROP Compilers nachgewiesen werden.

Zum Schluss werden die Ergebnisse mit anderen ROP Compilern verglichen und die Unterschiede in Performance sowie Qualität evaluiert.

2.2 Abgrenzung

In dieser Arbeit wird die Findung von ROP Gadgets nicht betrachtet, sondern durch bereits bestehende Projekte, welche innerhalb der Arbeit vorgestellt und referenziert werden sollen, zur weiteren Verarbeitung vorgegeben.

Des Weiteren wird sich in dieser Arbeit ausschließlich auf eine Prozessorarchitektur spezifiziert, da diese sich tiefgehend in den zugrundeliegenden Techniken unterschei-

den. Ein universeller ROP Compiler kann daher in dieser Arbeit nicht entwickelt werden, bildet aber Möglichkeit für nachfolgende Projekte.

3 Stand der Forschung

ROP und JOP sind Techniken, welche bereits seit Ende der 90er Jahre genutzt werden. Einer der wohl bekanntesten Exploits auf ROP Basis ist der return-into-libc Overflow Exploit [Des97].

Für das Auffinden von Gadgets gibt es ebenfalls bekannte Projekte, welche in der Lage sind teilweise für verschiedene Prozessorarchitekturen Gadgets aus einer gegebenen Binary zu finden und anzeigen zu lassen. Ein Beispiel für solch ein Projekt ist das ROPgadget Tool [Sal].

GENROP [Bra22]
MIT ROP Compiler [SD15]
angrop [ang]
SpecROP [BSK+20]
PCOP [SNR18]
Q: [SAB11]
MAJORCA [NVLK21]
ROPecker [CZM+14]

4 Vorarbeiten

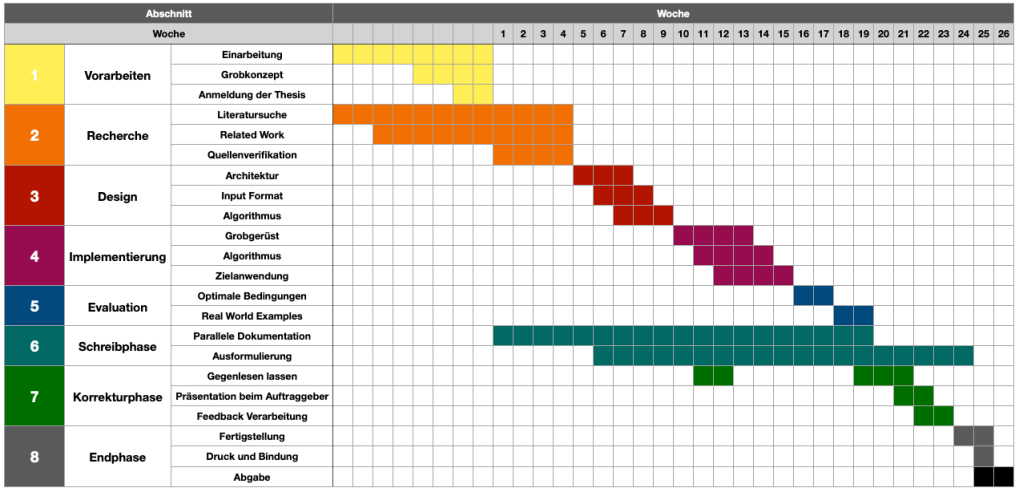
Diese Abschlussarbeit wird in Kooperation mit der Zentralen Stelle für Informationstechnik im Sicherheitsbereich (ZITiS) erstellt. Hierzu wurde bereits Kontakt mit der ZITiS aufgenommen sowie alle benötigten vertraglichen Grundsätze geklärt. Die benötigte Sicherheitsüberprüfung 1 befindet sich zum aktuellen Zeitpunkt (10. Januar 2023) noch beim Bundesamt für Verfassungsschutz (BfV) im Prozess. Unter Voraussetzung der erfolgreichen Überprüfung ist mit der ZITiS ein offizieller Vertragsbeginn am 01.03.2023 festgelegt.

Zur Bearbeitung wurde bereits durch die ZITiS ein Betreuer festgelegt, mit welchem bereits erster Austausch stattgefunden hat. Des Weiteren soll durch die ZITiS ab offiziellem Vertragsbeginn Hardware sowie Zugang zu gewissen Informationen und Infrastruktur bereitgestellt werden.

Eine Einarbeitung in das Thema, die technischen Grundlagen sowie etwaige Suche von Quellen und verwandten Projekten fand und findet bereits statt.

5 Vorgehensweisen und Methoden

6 Zeit und Arbeitsplan



Literaturverzeichnis

[ang] *angr/angrop*. – Abgerufen: 09.01.2023

[Bra22] BRANTING, Jonatan. *ROP-chain generation using Genetic Programming: GENROP*. 2022

[BSK+20] BHATTACHARYYA, Atri ; SÁNCHEZ, Andrés ; KORUYEH, Esmaeil M. ; ABU-GHAZALEH, Nael ; SONG, Chengyu ; PAYER, Mathias: {SpecROP}: Speculative Exploitation of {ROP} Chains. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, S. 1–16

[CZM+14] CHENG, Yueqiang ; ZHOU, Zongwei ; MIAO, Yu ; DING, Xuhua ; DENG, Robert H.: ROPecker: A generic and practical approach for defending against ROP attack. (2014)

[Des97] DESIGNER, Solar: *Getting around non-executable stack (and fix)*. 1997. – Abgerufen: 09.01.2023

[NVLK21] NURMUKHAMETOV, Alexey ; VISHNYAKOV, Alexey ; LOGUNOVA, Vlada ; KURMANGALEEV, Shamil: MAJORCA: Multi-Architecture JOP and ROP Chain Assembler. In: *2021 Ivannikov Ispras Open Conference (ISPRAS)* IEEE, 2021, S. 37–46

[SAB11] SCHWARTZ, Edward J. ; AVGERINOS, Thanassis ; BRUMLEY, David: Q: Exploit hardening made easy. In: *20th USENIX Security Symposium (USENIX Security 11)*, 2011

[Sal] SALWAN, Jonathan: *JonathanSalwan/ROPgadget*. – Abgerufen: 09.01.2023

[SD15] STEWART, Jeff ; DEDHIA, Veer: ROP compiler. In: *URL: https://css. csail. mit. edu/6.858/2015/projects/je25365-ve25411. pdf* (2015)

[SNR18] SADEGHI, AliAkbar ; NIKSEFAT, Salman ; ROSTAMIPOUR, Maryam: Pure-Call Oriented Programming (PCOP): chaining the gadgets using call

instructions. In: *Journal of Computer Virology and Hacking Techniques* 14 (2018), Nr. 2, S. 139–156