Projekt: Rowhammer

Gilian Henke
Dominik Mairhöfer
University of Lübeck
Email: gilian.henke@student.uni-luebeck.de
dominik.mairhoefer@student.uni-luebeck.de

***Abstract*—Das Ziel des Projekts ist es einen One-Location Rowhammer Angriff nach dem Paper [1] zu implementieren. Dieser Angriff soll dann verwendete werden, um lokal die Rechte zu erweitern. Dafür soll unter anderem auch die neue Methode des Memory chasing verwendet werden.**
Beim hierbei zu Grunde liegenden Modell, kann die Software Schutzmechanismen besitzen, aber nicht die Hardware. Der Angreifer kann ein arbiträres, unprivilegiertes Programm starten. Unter diesen Voraussetzungen kann der Angriff durchgeführt werden.

I. Motivation

[Know what you want to do and why that is interesting (maybe with bullet points). But do not write this section until you know what you actually have done so that the motivation fits your work. Aus diesen genannten Gründen noch nicht geschrieben.]

■ 1

We will stick to the following timeline: (Zur Orientierung beibehalten)

- 12/18: Submission of motivation, background, project scope and related work: please give a description of the technical aspect of your work, i.e. detailed background description of attacks. Also describe related work and outline, in more detail, your technical steps.
- 1/15: First version of report. This should be ver close to the final version, only technical parts that have not yet been finished should be missing.
- 2/1 : Final submission of complete project description.
- 2/8: Project Presentations in class: 25 minutes presentation plus questions.

II. Hintergrund

Die hier betrachteten Rowhammer Angriffe basieren auf folgendem Modell. Das System, auf welchem der Angriff ausgeführt werden soll, besitzt keine Maßnahmen der Hardware, um Rowhammer Angriff zu entdecken und zu unterbinden. Die verwendete Software darf jedoch versuchen, zu erkennen, ob ein Angriff stattfindet, und jenen zu verhindern. Damit solch ein Angriff überhaupt ausgeführt werden kann, muss der Angreifer in der Lage sein, ein beliebiges, nicht privilegiertes Programm auf dem System zu starten.
Im Folgendem werden dann die Methoden beschrieben, mit denen ein Rowhammer Angriff ausgeführt werden kann. Im Rahmen dieses Projekts wird dann versucht, einen Teil dieser Methoden anzuwenden und zu implementieren.

A. Memory Waylaying

In dem Paper werden zwei neue Methoden vorgestellt, um eine Seite an einer bestimmten physikalischen Adresse im Arbeitsspeicher zu platzieren. Dieser Prozess nennt sich Memory Waylaying. Hierbei wird im Gegensatz zu anderen Methoden nicht der gesamte Speicher mit Daten gefüllt, was die Entdeckung schwieriger macht. Das dies möglich ist, beruht darauf, dass, wenn Daten, wenn sie aus dem DRAM entfernt werden, beim Laden an eine zufällige Adresse platziert werden. Durch wiederholtes Anwenden dieser Methode kann die Seite an die richtige Position im Speicher positioniert werden.
Mit Hilfe eines Prefatch-based Prediction Oracle, näher beschrieben in [2], kann dann überprüft werden, ob zwei virtuelle Adressen, auf die gleiche physikalische verweisen. Damit kann dann überprüft werden, ob eine Seite an einer bestimmten Stelle im Speicher liegt. Dieses Orakel besteht aus zwei Schritten. Zunächst der prefatch und dann eine Flush and Reload Attacke.
Da der komplette Prozess des Memory Waylaying Laufzeiten von 100 Stunden benötigt. werden wir die effizientere Variante des Memory Chasing betrachten und implementieren. Beim Memory Chasing wird der copy-on-write Efekt des *fork*-Befehls ausgenutzt.

B. One-Location Rowhammer

Durch die fortschreitende Entwicklung vom DRAM, werden einzelne Speicherzellen mit immer weniger Spannung betrieben. Dies führt jedoch, dazu dass geringe Änderungen der Spannung ein Flip eines Bits zur Folge haben. Dieses Flippen kann man durchs gehäufte, schnelle Zugreifen auf benachbarte Speicherzellen bewusst auslösen. Beim double-sided Rowhammer wird auf beiden Seiten der zu flippenden Speicherreihen gehämmert, während beim single-sided Rowhammer nur eine Seite gehämmert wird. Im Gegensatz zum Namen wird bei diesen Angriffen jedoch mehr als nur eine einzige Speicherzelle gehämmert.
Betrachten wir als nächstes den One-Location Rowhammer. Bei dieser Methode wird im Gegensatz zu den üblichen Methoden nur eine einzige Stelle im Speicher gehämmert. Hierbei werden zwar weniger Bitflips erzeugt, aber die Entdeckung ist wesentlich schwieriger. Die Anzahl der hierbei erzeugten Bitflips ist aber immer noch ausreichend, um einen Angriff durchzuführen.

C. Privilege Escalation Attacke

Mit den beschriebenen Methoden lassen sich zwei Arten von Angriffen durchführen: Eine Privilege Escalation Attacke oder eine Denial-of-Service Attacke. In diesem Projekt verwenden wir die Methoden, um eine Privilege Escalation Attacke durchzuführen und dadurch Root-Rechte auf dem System zu erhalten.

D. Abgrenzung des Themas

Im folgenden gehen wir auf Themen ein, welche zwar im Paper angesprochen werden, wir im Rahmen dieses Projektes jedoch nicht weiter betrachten werden.

1) Intel SGX: Intel SGX (Software Guard Extension) ist eine Erweiterung, um Speicherbereich vor anderen Prozessren, insbesondere auch privilegierten, zu beschützen. Dadurch sollen Angriffe schlechter erkannt werden. Diese Erweiterung ist jedoch nicht auf jedem Rechner vorhanden. Außerdem ist sie nicht zwingend notwendig, One-Location Rowhammer Angriff durchzuführen. Daher werden wir die Verwendung dessen nicht weiter betrachten.

2) Memory Waylaying: Die erste der im Paper bechriebenen Methoden zum Memory Waylaying, hat zu lange Laufzeiten, als das sie im Rahmen dieses Projekts vernünftig untersucht werden kann. Dementsprechend werden wir jene nicht implementieren.

3) Denial-of-Service Attacke: Die hier beschrieben Denial-of-Service Attacke benötigt Intel SGX und wird dementsprechend von uns nicht weiter betrachtet.

E. Weitergehende Arbeiten

In [3] werden die Angriff in leicht verständlicher Art anschaulich dargestellt und geben einen Ansatz, um eine Implementation selbst durchzuführen.
In [4] wird eine deterministische Version des Angriffs auf Android-Systeme und eine Verallgemeinerung dieses Angriffes auf Linux-Systeme vorgestellt.

III. Work Description

Here you describe the work you have performed, problems you have solved and methods you have used. There is a fine balance between brevity and conciseness and ensuring that other people, if investing the time, would be able to reproduce
■ 2 your results given this description. `[this]`

IV. Results

`[here you will present and discuss your outcomes: implementation results or`
■ 3 `measurements or other project outcomes]`

V. Conclusion

■ 4 `[TBD last]`

References

[1] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of rowhammer defenses," *CoRR*, vol. abs/1710.00551, 2017.

[2] D. Gruss, C. Maurice, A. Fogh, M. Lipp, and S. Mangard, "Prefetch side-channel attacks: Bypassing SMAP and kernel ASLR," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 368–379, 2016.

[3] M. Seaborn and H. Flake, "Exploiting the dram rowhammer bug to gain kernel privileges," *Black Hat Briefings*, 2015.

[4] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (New York, NY, USA), pp. 1675–1689, ACM, 2016.