Side-Channel Security 2024
Exercise 1

In this exercise, you will get hands-on experience with a small selection of microarchitectural attacks.

1. This exercise is worth **up to 15 points**.
2. For a positive grade, you need **at least 4 points** on both exercise 1 (this sheet) and exercise 2.
3. Only task 1 is mandatory: if you do not submit it, you will not get a grade.
4. Put all source code, tools, intermediate files, and result files for this submission into folder `ex1` in your repository.
5. Tag your submission with the tag `ex1-1` before the first deadline so we know you will participate. Tagging the second submission (`ex1-2`) is only necessary if you don't want the last commit on main to be used.
6. For all tasks, you can find more information at `https://www.iaik.tugraz.at/course/side-channel-security-705048-sommersemester-2023/`

1 Introduction to Cache Attacks – 0.5 Point

Mandatory
Deadline: Wednesday, March 20 2024, 8:00am

Produce a Flush+Reload cache hit/miss histogram on at least one machine per team member, and **submit a plot image** into your repository. Use your histogram to decide on a *good* threshold for each machine. **Both team members** must be able to reproduce their histogram in the exercise interview and explain their choice of threshold.
You may use the histogram demo in the upstream repository, **no programming is necessary** for this task!

2 Flush+Reload Attack on PIN entry – 3 Points

Optional
Deadline: Wednesday, May 8 2024, 8:00am

We recently got our hands on a PIN-entry library. Rumours suggest that it was written by a student that had just learned about timing attacks on PIN entries, and tried to make the function `check_pin` constant-time by comparing each letter individually and calling either `markDummyAsWrong` or `markInputAsWrong`. To us, this sounds like a case for a Flush+Reload attack! Use Flush+Reload to leak the pin from the library. You may change the library/PIN for testing, but we will provide a pre-compiled library for the interviews with a 10 digit *numeric* PIN.

3 Cache Covert Channel – 5.5 Points

Optional
Deadline: Wednesday, May 8 2024, 8:00am

Implement a *fast* (see Table 1) and reliable cross-core covert channel based on a cache attack (Flush+Reload, Prime+Probe or Flush+Flush). Your channel must be able to take arbitrary *binary* files as input, and output to a file. Transmit a file large enough that the transmission takes 60 seconds, with random (`/dev/rand`) or meaningful non-uniform content (`png`, `jpeg`, `mp3`, etc.). Compute the true channel capacity T as

$$T = C \cdot (1 + ((1-p) \cdot \log_2(1-p) + p \cdot \log_2(p))),$$

where C is the *raw capacity* and p is the *bit error ratio*.
You do not need to implement any error correction.

Capacity T (kiB/s)	7.5	13	20	27	35	45	55	67	**80**	130	200	500
Points	1.5	2	2.5	3	3.5	4	4.5	5	**5.5**	6	6.5	7

Table 1: True channel capacity to points.

3.1 Bonus Points

- +1 – Your group has the fastest channel[1].
- +1 – your channel uses Prime+Probe or Flush+Flush.
- +3 – Prime+Probe on exotic hardware (ask us what qualifies).
- +1 – your Prime+Probe channel needs no access to physical addresses.
- +1 – your Prime+Probe channel works across virtual machines.

A scaling factor of 2/3 will be applied to the capacity requirement for Prime+Probe channels.

4 `Spectre` Attacks – 3 Points

Optional
Deadline: Wednesday, May 8 2024, 8:00am

We have found a library that we think is susceptible to a Spectre-PHT attack[2]. We have provided it to you in your repository. You can access anything defined in the library's header to extract the secret string. You may not change the library or read the secret directly from memory. Most modern processors are affected by Spectre, but we recommend you use Intel or AMD. During the exercise interview, you will be given a new shared object file with a different secret, so make sure to test if your solution works for any string (though you can assume ASCII 63-95). Points will be awarded according to Table 2/Table 3 (Intel/AMD), though for this task your hardware and your explanation during the exercise interviews will also have an influence.

time to secret (s)	<160	<100	<70	<50	<**25**	<8	<0.5
Points	1	1.5	2	2.5	**3**	3.5	4

Table 2: Time to recovery of the (correct) secret to points on Intel.

time to secret (s)	<160	<100	<70	<50	<**25**	<12	<3
Points	1	1.5	2	2.5	**3**	3.5	4

Table 3: Time to recovery of the (correct) secret to points on AMD.

4.1 Bonus Points

- +1.5 – you can mistrain the target branch from a different process[3], without calling the library function in that process.

5 KASLR is bad, please break it – 3 Points

Optional
Deadline: Wednesday, May 8 2024, 8:00am
Break kernel address space layout randomization using one of the methods presented in the lectures. Your goal is to find the starting virtual address of the Kernel within the possible address range. The simplest method is timing of prefetch instructions, but you can earn extra points for using Data Bounce[4]. If you use prefetch instructions, investigate which ones (or combinations thereof) work best for finding the kernel. You can also earn extra points for especially fast or robust implementations[5]. Please add a sample visualization (plot) of your program's output to your repo. For simplicity, we recommend using an Intel CPU for this task.

Notes

[1]Send a discord message to Lukas (redrabbyte) with *raw capacity*, *bit error ratio*, and any extra details you want included to update the speed record page `https://www.iaik.tugraz.at/teaching/materials/scs/exercises/ex1/`. Final rankings are based on the exercise interviews.

[2]`http://spectreattack.com/`

[3]On newer hardware, you may need to deactivate STIBP, ask in Discord if you're unsure. `https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/single-thread-indirect-br html`

[4]As described here `https://arxiv.org/pdf/1905.05725.pdf`

[5]Send a discord message to Lukas (redrabbyte) with *method*, *run time*, and *reliability*, to update the KASLR record Wiki page `https://www.iaik.tugraz.at/teaching/materials/scs/exercises/ex1/`.