IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 9, 2022, accepted February 24, 2022, date of publication March 3, 2022, date of current version March 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3156596

ThermalBleed: A Practical Thermal Side-Channel Attack

TAEHUN KIM AND YOUNGJOO SHIN
School of Cybersecurity, Korea University, Seoul 02841, South Korea
Corresponding author: Youngjoo Shin (syoungjoo@korea.ac.kr)

This work was supported as part of Military Crypto Research Center (UD210027XD) funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

ABSTRACT Modern OSs expose an interface for monitoring CPU temperature to unprivileged users for effective user decision-based thermal management. Due to the low sampling rate and resolution, thermal sensors have generally been restricted to the construction of covert channels. However, exposing the thermal interface to unprivileged users may be problematic, because the heat emission inside a CPU core is affected by program execution on the core; an attacker may be able to infer the secret information of the program by exploiting the thermal interface as a side-channel. In this paper, we extensively analyze digital thermal sensors in Intel CPUs and show that it is possible to implement a software-based thermal side-channel attack. Specifically, by analyzing some properties of the thermal sensors, we inferred that the thermal sensor makes it possible to distinguish between a cache hit and a physical memory access in memory load operations. Based on the analysis results, we implement ThermalBleed, a thermal side-channel attack that breaks kernel address space layout randomization (KASLR) in Linux systems. Moreover, by conducting an in-depth analysis, we identify useful hidden properties of the Intel thermal sensors. Our analysis establishes a stepping stone to build a more precise and effective thermal side-channel attack in the future. To the best of our knowledge, this is the first work that extends a thermal covert channel to a practical side-channel attack by exploring the properties of Intel digital thermal sensors.

INDEX TERMS Breaking KASLR, digital thermal sensor, thermal side-channel attack.

I. INTRODUCTION

CPU vendors make use of various performance-optimizing techniques (e.g., cache, out-of-order, and speculative execution) to meet the requirements of various applications. For the purpose of CPU monitoring, they provide interfaces through which various types of measurement data such as CPU frequency, energy consumption, and temperature are obtained. Exposing the monitoring interface to any user, including unprivileged ones, will enable a more accessible, efficient, and scalable system management. However, it may introduce another attack surface if security is not sufficiently considered.

Unfortunately, little attention has been paid to security issues on unprivileged CPU monitoring interfaces in previous literature. Some previous works have studied covert channels using a clock skew [1] and dynamic frequency scaling (DFS) [2] interfaces. Recently, some researchers found that the Intel running average power limit (RAPL), an unprivileged interface that monitors power consumption, can be a reliable leakage source [3], [4]. After the vulnerability has been disclosed, OS maintainers came up with a security patch that restricted the use of the interface only to privileged users [5].

Hence, the current versions of the Linux kernel no longer expose the RAPL interface to unprivileged users. However, we deduce that there is still a vulnerable interface on the processors. In this paper, we focus on a thermal sensor in Intel CPUs, another unprivileged interface that supports temperature monitoring. Specifically, we extensively analyze various properties of the Intel digital thermal sensor. The analysis results give us primitives that distinguish between a cache hit and a physical memory access in memory load operations through elaborate monitoring of the thermal sensors.

To demonstrate the vulnerability of Intel digital thermal sensors, we present ThermalBleed, a practical thermal side-channel attack. ThermalBleed breaks kernel address space

The associate editor coordinating the review of this manuscript and approving it for publication was Zesong Fei.

layout randomization (KASLR) in state-of-the-art Linux systems that are equipped with software- [6] and hardware-based [7] countermeasures against KASLR breaking side-channel attacks. Owing to the properties of the CPU thermal sensors that we evaluated, the memory access to the physically backed address shows a sharp increase in the temperature as causes dTLB hits. However, memory access to invalid addresses shows a slow increase in temperature because it causes dTLB misses and a page table walk. By measuring the CPU temperature, an attacker can successfully de-randomize the kernel address. Such thermal differences between the dTLB hits and dTLB misses provide a sufficiently reliable channel to the attacker despite various noise factors (e.g., dynamic voltage and frequency scaling (DFVS), cooling devices, and remnant heat [8]).

We evaluate ThermalBleed under various environments, including servers, desktops, and laptop CPUs. Because Intel digital thermal sensors have been introduced since Pentium processors, ThermalBleed attacks are applicable to most Intel CPUs. Our experimental results show that for certain CPU models, ThermalBleed took only approximately 9 min to de-randomize KASLR with 100% accuracy.

We would like to emphasize that ThermalBleed is a *timer-free* side-channel attack. That is, ThermalBleed does not rely on any timing information to deliver the attack. Compared to previous side-channel attacks that require precise timers, our attack is more resistant to defense mechanisms that limit access to the timing source [9], [10].

To the best of our knowledge, ThermalBleed is the first software-based practical thermal side-channel attack in real-world scenarios. However, a limited number of studies [8], [11], [12] on thermal properties make it difficult to construct more precise and effective thermal side-channel attacks. For instance, the previous work simply utilizes a basic thermal property that a compute-intensive workload generates more heat than a lightweight workload. This restricts the research direction to coarse-grained covert channel attacks only. Moreover, there is no previous work that attempts to overcome the low resolution (i.e., $\pm\ 1°C$) and sampling rate (i.e., 2 ms), which hinders enlarging an attack surface. Thus, we conducted an in-depth analysis of the Intel digital thermal sensors to help future research.

To disclose unexplored properties on the sensors, we thoroughly investigated Intel documents [13]–[17] related to thermal management and conducted a reverse engineering on the sensors. We also considered some plausible elements that correlate with core temperature and verified this hypothesis with the experiment result. Based on the analysis, we identified two important thermal properties. First, we figured out that the digital thermal sensors were individually placed in each core. Although our finding seems somewhat trivial, we show that the location of the sensor is a crucial factor in building a precise thermal side-channel attack. Second, we found a correlation between the core temperature and the current and voltage applied to the CPU. This finding is also important; This means that we can observe a difference in the temperature even for the same instructions with different operand values. This raises the possibility of implementing more precise thermal side-channel attacks that can distinguish instructions. These findings regarding unexplored thermal properties may be utilized as a stepping stone to build advanced thermal side-channel attacks that leak confidential data (e.g., a secret key).

Finally, as a countermeasure, we proposed restricting the thermal interface to the privileged level to mitigate software-based observable thermal side-channel attacks.

A. CONTRIBUTIONS

The contributions of this paper are as follows.

1) We evaluate digital thermal sensors in Intel CPUs and uncover some properties that can be used as a practical leakage source.
2) We present ThermalBleed, a first practical thermal side-channel attack that breaks KASLR in Linux from user mode applications.
3) We analyze some properties of CPU thermal sensors. Based on this observation, we uncover useful hidden properties of thermal sensors that make it possible to enforce a thermal side-channel attack.

B. OUTLINE

The remainder of this paper is organized as follows. In Section II, we discuss background knowledge regarding thermal side-channel analysis. In Section III, we present attack primitives for the ThermalBleed attack. In Section IV, we present how the ThermalBleed breaks KASLR and evaluate the attack on various systems. In Section V, we conduct an in-depth analysis on the properties of an Intel digital thermal sensor for future research. In Section VII, we discuss the related work and finally, we conclude this paper in Section VIII.

II. BACKGROUND

In this section, we present background knowledge on thermal analysis and the ThermalBleed attack.

A. THERMAL SIDE-CHANNEL ANALYSIS

The law of energy conservation (i.e., the first law of thermodynamics) [18] explains that energy is not created or dissipated but conserved, where it is only transformed to other forms (e.g., heat). More importantly, Joule's first law (i.e., Joule heating) [19] states that when an electric current flows through a conductor, the electrical energy is transformed into thermal energy. Equation (1) represents the heating effect of an electric current, where H is the amount of heat, I is the electric current, R is the electrical resistance of the conductor, and t is the amount of time that the current flows.

$$H = I^2 \cdot R \cdot t \ = V \cdot I \cdot t \ = P \cdot t \ [J] \tag{1}$$

With the Ohm's law, H can be expressed by the other form, where P is the power consumption and V is voltage. Thus, the

heat generated by the electric components is highly related to the power consumption.

The microprocessor is a set of electric components (i.e., transistors), and each instruction has a distinct path in which the current flows depending on its mnemonic operands. Thus, the heat generated by running a process differs according to the workload. Thermal side-channel analysis is a method that infers security-sensitive data such as secret keys, by exploiting the observable variance of thermal data during the execution of a victim application. Using this method, an attacker can leak sensitive information from the victim's thermal behavior.

B. HWMON INTERFACE

Since the P6 microarchitecture, Intel includes a thermal sensor in its processors to control and manage the system by monitoring the thermal data [20]. It enables safety and user-decision based thermal management. For example, when the core temperature is higher than the threshold, the sensor sends a shutdown signal to the CPU. Consequently, it protects the silicon from being permanently damaged by overheating.

1) hwmon

Hardware devices support thermal, voltage, current, and fan speed sensors to monitor the system [21]. Linux provides `hwmon`, a software subsystem that provides a hardware monitoring interface. The `hwmon` interface allows users to retrieve these data from a CPU, motherboard and graphic processing unit (GPU). Because most Intel CPUs support digital thermal sensors on each core, `hwmon` always has a *coretemp* domain, which reports the current CPU temperature from the package and each core.

Because the `hwmon` interface is implemented for Linux, the other OSs (e.g., Windows or Mac OS) do not support it by default. However, with additional application installation, they can also monitor the temperature of hardware resources such as the CPU, GPU, and storage.

In this work, we mainly focus on the *coretemp* domain of the `hwmon` interface, which reports the temperature on each core. The core temperature can be retrieved using `/sys/class/hwmon/hwmon[0-*]` in the Linux file system. It allows any user mode applications to obtain the current CPU temperature by reading files. Importantly, the interface has a $\pm 1°C$ resolution and a 2 ms sampling rate.

C. ADDRESS TRANSLATION

In modern OSs, each process has a separate virtual address space, so that it does not overlap with the space of other processes. The process isolation through memory virtualization is a crucial technique for implementing system security. Because data are accessible through physical addressing, virtual-to-physical address translation needs to be performed through a memory management unit (MMU).

Fig. 1 illustrates the overall procedure of address translation in x86-64 processors. The virtual address consists of a virtual page number (i.e., the upper 36 bit) and a page offset (i.e., the lower 12 bit), where each part is used to look up in a translation lookaside buffer (TLB) and L1 cache, respectively. Because the page offset is the same for both virtual and physical addresses, only the virtual page number needs to be translated to a physical frame number through a multilevel page table walk. Because the page table walk is time-consuming, the TLB stores the recent results of address translation and serves subsequent requests to the same address directly from the results. When a TLB miss occurs, a page miss handler retrieves a page table entry through the page table walk, and then serves a physical address depending on its permission bits (e.g., present bit).

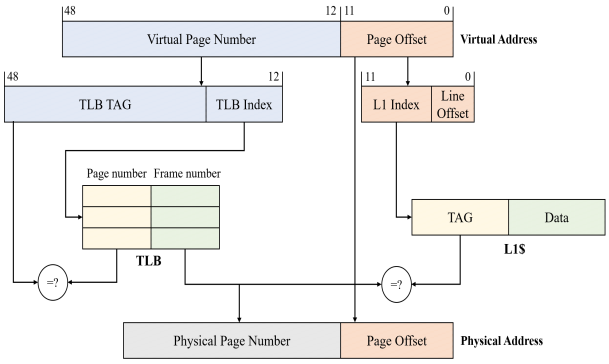

FIGURE 1. Workflow of address translation.

D. ADDRESS SPACE LAYOUT RANDOMIZATION

Memory corruption attacks such as ROP attacks [22], [23] use the knowledge of the memory layout to compromise a system. Thus, recent OSs provide address space layout randomization (ASLR), a defense mechanism based on non-deterministic behavior. The kernel ASLR (i.e., KASLR) is an efficient strategy that mitigates memory corruption attacks against OS kernels. However, KASLR has been shown to be vulnerable to microarchitectural side-channel attacks [24]–[27]. These attacks mainly exploit the timing difference between a load from allocated and non-allocated pages. More specifically, a kernel text section in Linux is mapped in a 1 GB range (i.e., `0xffffffff80000000 – 0xffffffffc0000000`), where its base address is aligned to a 2 MB boundary. Hence, Linux has only nine bits of entropy in KASLR, and attackers can successfully determine the kernel base address through a maximum of 512 ($= 2^9$) times of guessing.

III. ATTACK PRIMITIVES

In this section, we describe ThermalBleed attack primitives. These attack primitives are identified by conducting experiments on thermal sensors. We present our setup for the experiment and then describe the attack primitives in detail. Specifically, we first show how to distinguish memory access using Intel digital thermal sensors: cache access and physical memory (i.e., DRAM) access. We also show that with the sensors, we can infer whether the executed instruction caused

TABLE 1. Experimental setup and average core temperature of the baseline, physical memory access, and cache hit. σ indicates standard deviation from the collected temperature data and Δ indicates the difference in temperature between the cache hit and physical memory access.

Class	CPU	Frequency	μ-arch	CPU cooler	Baseline	Physical memory access	Cache hit	Difference (Δ)
Laptop	Core i7-7200U	2.50GHz	Kaby Lake	Samngsung	32.281 $^\circ C$ ($\sigma = 0.62$)	44.644 $^\circ C$ ($\sigma = 1.92$)	52.412 $^\circ C$ ($\sigma = 2.49$)	7.768 $^\circ C$
Laptop	Core i7-10510U	1.80GHz	Comet Lake	Lenovo	40.150 $^\circ C$ ($\sigma = 0.38$)	43.692 $^\circ C$ ($\sigma = 0.84$)	46.657 $^\circ C$ ($\sigma = 1.29$)	2.965 $^\circ C$
Desktop	Core i7-6700K	4.00GHz	Sky Lake	Cooler Master	20.850 $^\circ C$ ($\sigma = 0.40$)	29.418 $^\circ C$ ($\sigma = 0.58$)	36.841 $^\circ C$ ($\sigma = 0.57$)	7.423 $^\circ C$
Desktop	Core i5-7400	3.00GHz	Kaby Lake	Intel	35.450 $^\circ C$ ($\sigma = 0.13$)	41.596 $^\circ C$ ($\sigma = 1.00$)	47.774 $^\circ C$ ($\sigma = 1.26$)	6.178 $^\circ C$
Desktop	Core i7-8700	3.20GHz	Coffee Lake	Cooler Master	26.136 $^\circ C$ ($\sigma = 0.51$)	34.308 $^\circ C$ ($\sigma = 0.81$)	46.591 $^\circ C$ ($\sigma = 0.85$)	12.283 $^\circ C$
Desktop	Core i5-9600K	3.60GHz	Coffee Lake	Cooler Master	28.137 $^\circ C$ ($\sigma = 0.43$)	35.532 $^\circ C$ ($\sigma = 0.51$)	44.453 $^\circ C$ ($\sigma = 0.80$)	8.921 $^\circ C$
Desktop	Core i7-10700	2.90GHz	Comet Lake	Deepcool	28.750 $^\circ C$ ($\sigma = 0.51$)	36.864 $^\circ C$ ($\sigma = 0.39$)	48.642 $^\circ C$ ($\sigma = 0.65$)	11.778 $^\circ C$
Desktop	Core i9-10900	2.80GHz	Comet Lake	Cooler Maser	28.282 $^\circ C$ ($\sigma = 0.46$)	33.396 $^\circ C$ ($\sigma = 0.53$)	39.140 $^\circ C$ ($\sigma = 0.97$)	5.744 $^\circ C$
Server	Xeon E3-1270 v6	3.80GHz	Kaby Lake	Cooler Maser	33.665 $^\circ C$ ($\sigma = 0.50$)	43.677 $^\circ C$ ($\sigma = 0.45$)	53.607 $^\circ C$ ($\sigma = 0.59$)	9.930 $^\circ C$
Server	Xeon Silver 4210	2.20GHz	Cascade Lake	Noctua	29.104 $^\circ C$ ($\sigma = 0.32$)	32.276 $^\circ C$ ($\sigma = 0.57$)	35.503 $^\circ C$ ($\sigma = 0.57$)	3.227 $^\circ C$

a TLB hit or not, giving unprivileged users the ability to distinguish address translations.

A. EXPERIMENTAL SETUP

We conducted experiments on various Intel CPUs ranging from laptop to server-class processors. In the experimental setup, we used a default system configuration in which no modifications were applied to the system (i.e., disabling Intel C-states, Turbo Boost, and DVFS). Our systems run Ubuntu 18.04 or later, with versions 4 and 5 of Linux kernels. Table 1 lists the tested system configurations, for which various kinds of air CPU coolers are used. In each system, we run a *target application*, a program that simply executes a number of load instructions, which may result in cache hits or physical memory accesses. During the execution, another program, referred to as a *collecting application*, measures the heat generated with the thermal sensor every 2 ms, until 100 K measurements are obtained. To minimize heat propagation between these two programs, we isolated each program in different physical cores.

B. DISTINGUISHING MEMORY ACCESS

We analyze the feasibility of the thermal sensor in distinguishing cache access (i.e., cache hit) from physical memory access. In previous studies [8], [11], [12], [28], remnant heat or heat propagation occurring during the execution of a compute-intensive application were exploited to construct a covert channel. However, because of the limited capacity (i.e., resolution and sampling rate), these approaches are impractical for implementing *side-channel attacks*. To implement a practical thermal-based side-channel attack, we focus on the heat difference observed in different types of memory load operations: the load from the cache and that from the physical memory. This difference is attributed to the fact that the CPU thermal sensors measure and report the current CPU temperature from the die [14], [20]. This means that the heat generated from the physical memory (i.e., DRAM) is excluded from the measurement as it is

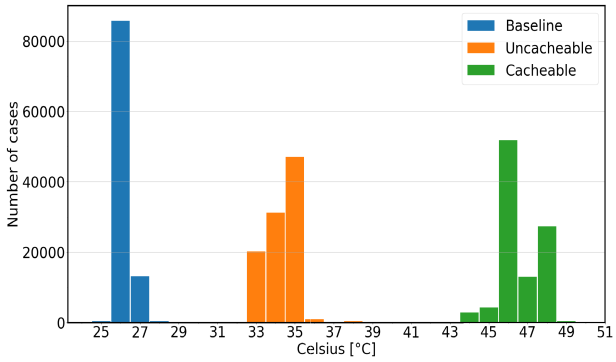

FIGURE 2. Distribution of temperature measurements on i7-8700 under a baseline, load from cacheable and uncacheable pages.

physically located outside the package. Thus, by measuring the temperature with the thermal sensor, we can determine which component of either the cache or physical memory is in charge of serving the load.

To precisely identify the difference, we conducted an experiment to measure the temperature under various settings; 'Baseline', 'Physical memory access' and 'Cache hit'. Under the 'Baseline' setting, we measured the CPU temperature without executing a target application. 'Physical memory access' is the setting in which the temperature is measured while the target application is running where a load occurs on *uncacheable* memory page. 'Cache hit' is the same as the setting of 'Physical memory access' except that the load takes place on a *cacheable* memory page. Table 1 lists the average temperature measured on various systems and its standard deviation. Fig. 2 particularly shows the core temperature traces on an i7-8700 CPU. As shown in the figure, there is a clear difference in the temperature, where 'Cache hit' (i.e., *cacheable*) shows approximately 46.591 $^\circ C$ ($n = 10^5$, $\sigma = 0.85$), and 'Physical memory access' (i.e., *uncacheable*) shows approximately 34.308$^\circ C$ ($n = 10^5$, $\sigma = 0.81$). In our experiments, we could reliably distinguish the cache access from the physical memory access with an observable thermal difference. This difference is consistent

with various CPU models, regardless of the CPU cooler (cf. Table 1). It is also noteworthy that the measured standard deviation of the collected data is smaller than expected. We attribute this to the remnant heat. That is, the temperature measurement may be affected by the remnant heat caused from preceding instructions. This phenomenon allows us to build a reliable and efficient thermal-based side-channel, through which we can distinguish different types of memory accesses.

C. DISTINGUISHING ADDRESS TRANSLATION

We further show that thermal sensors make it possible to distinguish an address translation that hits a TLB from the one that does not. Hence, we can eventually distinguish access to an *allocated kernel page* from the access to a *non-present* page. For this, we utilize software-based prefetch instructions such as `prefetchnta` and `prefetcht2`. These prefetch instructions lack a privilege check on the target address [29]. Thus, we can perform prefetching at arbitrary addresses regardless of the user/supervisor bit.

Prefetching data from a physical memory to a cache needs an address translation. When a TLB miss occurs, a page miss handler looks up a page table entry through a multilevel page table walk. If the translation fails, a TLB entry is not created for a non-present page. If it succeeds, the entry is allocated as a result of address translation. We observe that as long as the present bit of a page table entry is set, the MMU always allocates a TLB entry for a valid address regardless of the user/supervisor bit. Thus, any user mode application (i.e., unprivileged users) can cause a TLB hit and miss from the allocated kernel page and non-present page, exploiting the lack of permission checks.

We measured the temperature from a TLB hit and miss on the various environments. Likewise, we collected 100 K of temperature traces every 2 ms, executing `prefetchnta` and `prefetcht2` instructions on the allocated kernel page and non-present page, respectively. In the core thermal sensor, the average temperature measured on the TLB hit was 62.125 $°C$ ($\sigma = 3.77$) in i7-7200U, 68.883 $°C$ ($\sigma = 0.67$) in i7-10700, and 62.934 $°C$ ($\sigma = 0.92$) in Xeon E3-1270 v6. For the TLB miss, the temperature was 47.856 $°C$ ($\sigma = 2.50$) in i7-7200U, 46.246 $°C$ ($\sigma = 0.53$) in i7-10700, and 47.141 $°C$ ($\sigma = 0.98$) in Xeon E3-1270 v6. The difference in the temperature between the 'TLB hit' (i.e., *allocated kernel page*) and 'TLB miss' (i.e., *non-present page*) is approximately twice as higher than the difference in memory access. Specifically, the temperature difference on address translation was 14.269 $°C$ in i7-7200U, 22.637 $°C$ in i7-10700, and 15.493 $°C$ in Xeon E3-1270 v6. This higher difference is consistent with various CPU models, regardless of their experimental configuration. Thus, we figured out that the Intel digital thermal sensors provide a more reliable channel for inferring the TLB status than distinguishing memory access.

Fig. 3 illustrates the core temperature trace on i7-8700. This shows that there is also a clear temperature difference between the TLB misses and hits. In all CPUs classes, we can distinguish the TLB status by exploiting the core thermal sensors.

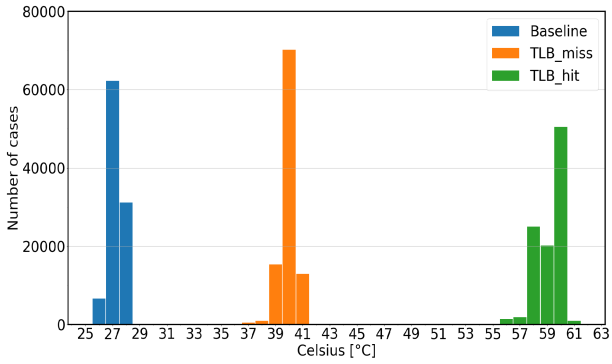

FIGURE 3. Distribution of temperature measurements on i7-8700 under a baseline, TLB miss, and TLB hit.

Another interesting observation is that we could infer the TLB status of the other physical cores using Intel digital thermal sensors. Generally, as the TLB hardware is placed in each core, a cross-core attack leveraging the TLB status as a side-channel was hard to implement. However, `hwmon` interface provides an unprivileged user to per core temperature. Thus, we can mount our thermal-based attack primitives regardless of the location of the victim application. This is a characteristic that further expands the attack surface of our thermal-based side-channel attack. For instance, we can infer the TLB status on *core 0* from *core 5* in i7-8700. This observation can be leveraged to implement cross-core thermal side-channel attacks in the future.

IV. BREAKING KASLR

In this section, we propose ThermalBleed, a practical thermal side-channel attack that de-randomizes KASLR. The basic idea of ThermalBleed is to infer kernel mappings by distinguishing an *allocated kernel page* from a *non-present page* using thermal information. For this, we apply the attack primitive presented in the previous section, which allows us to distinguish a TLB hit (for the allocated page) from a page table walk (for the non-present page).

ThermalBleed is the first cross-core attack that infers the TLB status of the other core without depending on hyperthreading. There are some challenges that need to be addressed to successfully deliver our ThermalBleed attack. One of the challenges is to deal with the heat propagation, i.e., the heat generated from a core may affect the other core's temperature. Another challenge comes from the thermal capacity and resistance, i.e., a minimal amount of heat is needed to increase the core temperature, and the remnant heat can be a noise source for precisely measuring the core temperature [8]. For instance, if the heat generated by the target application on a core is insufficient to increase the core temperature, there is no surge in temperature even if TLB hits occur by the execution of the application. It makes the

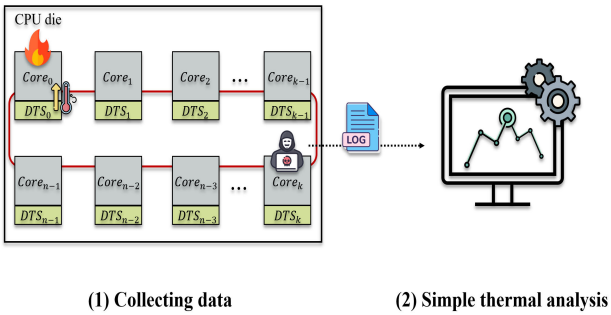

FIGURE 4. Overview of ThermalBleed attack.

reliable inference of the TLB state with ThermalBleed more challenging.

To overcome these challenges, we use a spatially partitioning strategy in which a target and collecting application are running on physically separated cores. In this setting, the collecting application obtains 300, or 500 number of sufficient temperature traces on each slot (i.e., possible kernel base address). Because the current version of the Linux kernel only has 9-bits of entropy in KASLR (cf. Section II-D), a kernel text section is mapped to one of 512 possible slots. By bruteforcing all the slots, we can reliably distinguish a physically backed kernel address from an invalid one.

A. THREAT MODEL

We assume an unprivileged attacker running a spy program on a target system. We also assume that the target system has neither bugs nor known vulnerabilities that allow the attacker to obtain the information of the kernel address. Because most Intel CPUs support digital thermal sensors, the attacker can retrieve the core temperature through the `hwmon` interface. We also do not make any modifications to our system such as disabling Intel C/P-state, Intel turbo boost, DFVS. The tested environment is the same as that in Table 1.

B. ThermalBleed ATTACK

The ThermalBleed attack proceeds in two phases: a data collection phase and a simple thermal analysis (STA) phase, as illustrated in Fig. 4.

1) PHASE 1: COLLECTING DATA

In this phase, the attacker collects data of the measured temperature via the `hwmon` interface. For this, the attacker runs a collecting and target application; the former collects and reports a core temperature every 2 ms, and the latter repeatedly executes a prefetch instruction with a target kernel address. However, the heat interference between these applications may degrade the attack capacity. For example, the heat emitted from the collecting application itself can be a source of noise at the core temperature. Our spatial partitioning approach can avoid this problem. That is, we place the target application on the first core (i.e., core 0) and the collecting application runs on the core farthest away from core 0.

The kernel text has 9-bit entropy in KASLR, and its base address is randomly mapped to 1 GB of the kernel address space with a granularity of 2 MB. This gives us a total of 512 possible kernel base addresses. Thus, the target application repeatedly performs prefetch instructions on all 512 possible base addresses while collecting application measures the temperature. However, owing to the thermal capacity and resistance, a single trace on each address (i.e., slot) is not sufficient to offer a reliable channel. There is no clear difference in the temperature between an allocated kernel page and a non-present page with a single trace. To overcome this problem, the collecting application obtains multiple temperature traces, i.e., T_n, from each slot. If the slot is a physically backed address, the target application will result in a dTLB hit, inducing a surge increase in the temperature. However, if the slot is an invalid address, it will cause a dTLB miss and page table walk, accessing a physical memory outside of the CPU package die. It introduces a slight increase in the temperature. Thereafter, the collected data of all the temperature traces against possible base addresses are handed over to the next phase.

Table 2 shows code snippets of a target and collecting application, which are used in the collecting data phase. As aforementioned, ThermalBleed needs to address several challenges caused by heat propagation, thermal capacity and resistance. For the elaborate temperature measurement, it is crucial to synchronize executions of a collecting and target application. We resolve the challenges by devising a synchronization algorithm (① in Table 2).

To address the heat propagation issue, a spatial partitioning strategy is used in the synchronization algorithm to minimize the noise caused by heat propagation from adjacent cores. That is, we can place a target and collecting application on spatially separate cores by using a `taskset` command (Lines 2 and 5 in ①).

Dealing with thermal capacity and resistance is not trivial. Due to those thermal properties, a certain amount of workload is necessary to make distinguishable variation in temperature. For instance, if the collecting application gets executed prior to the target application, the heat generated by the target application cannot immediately affect the core temperature. Thus, the obtained temperature traces will contain a huge amount of noise, disturbing the simple thermal analysis in Phase 2.

To address the challenges caused by thermal capacity and resistance, we come up with two solutions. The first solution is an execution ordering: run the target application (② in Table 2) first, and then run the collecting application (③). As the target application is initially heating the core, the execution ordering can address thermal capacity and resistance well. Our second solution is to increase the number of obtained temperature traces (i.e., T_n). By increasing T_n, we can reduce the noise introduced by the thermal capacity and resistance owing to the sufficient execution of the target

TABLE 2. Algorithms for the collecting data phase.

① Synchronization algorithm	② Code snippet of a target application	③ Code snippet of a collecting application
1: **Function** measure ($kaslr\_slot$) **begin**	**Input:** $kaslr\_slot$	**Input:** None
2: `taskset` -c $CORE1 ./target $kaslr\_slot$ &	**Output:** None	**Output:** a trace file
3: PID ← `getpid`(target)	// Core part of the target application	// Core part of the collecting application
4: `sleep`(20ms)	**while** $true$ **do**	**for** $i \leftarrow 0$ **to** $T_n - 1$ **do**
5: `taskset` -c $CORE2 ./collecting	`prefetch`(KERNEL_BASE + 2MB × $kaslr\_slot$)	temperature ← `get_core_temperature`()
6: `kill` PID	**end while**	`log_trace`(temperature)
7: **end Function**		`sleep`(2ms)
		end for

application. However, there is a trade-off between the attack accuracy and the overall execution time of the attack over the number of traces. We discuss this in more detail in Section IV-C.

2) PHASE 2: SIMPLE THERMAL ANALYSIS

There is a clear difference in the temperature between a dTLB hit from an *allocated kernel page* and a dTLB miss (and a subsequent page table walk) from a *non-present page* (cf. Section III-C). Thus, we use a simple thermal analysis to determine the base address of the kernel text. By analyzing the collected temperature data, we can distinguish whether the slot is an allocated page (i.e., higher temperature) or a non-present page (i.e., lower temperature).

To infer the base address of kernel text on KPTI enabled systems, ThermalBleed utilizes a kernel symbol `__entry_text_start`. This is a symbol for an entry point to enter the kernel mode or to switch the page table. Once the knowledge of the symbol address has been obtained, an attacker can compute the kernel base address from it. In other words, the attacker can identify the offset between the address of `__entry_text_start` and `startup_64`, which is a kernel symbol that refers to the kernel base. By subtracting the offset from the address of `__entry_text_start`, the attacker obtains the kernel base address (i.e., the address of the `startup_64`). On the other hand, in the case of the system where KPTI is no longer applied by default due to the silicon patch against microarchitectural timing attacks, we can directly infer the address of the `startup_64` as well as the whole size of kernel text.

We describe how ThermalBleed determines the address of the symbol `startup_64` with an experimental result on an i9-10900 CPU (see Fig. 5). While the target application performs a repetitive execution of prefetch instructions on each slot, the collecting application measures the core temperature 500 times for the slot. As shown in Fig. 5, there is a surge increase in temperature for the 478-th slot (i.e., red circles) which is approximately 11 $^{\circ}C$ higher than other slots. This indicates that the ThermalBleed reliably infers an allocated kernel page, which consequently results in the breaking of the KASLR. We also observed that

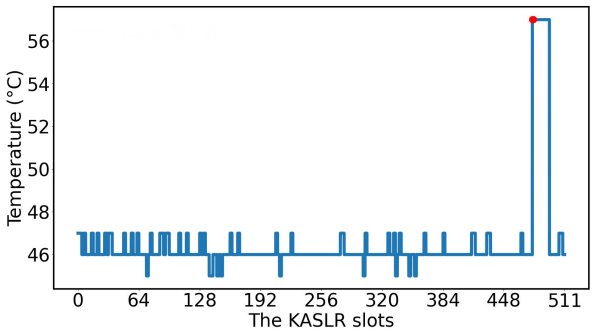

FIGURE 5. Core temperature traces when prefetch instructions are repeatedly executed on every possible base address.

additional slots between 478-th and 495-th also exhibit high temperatures. Based on the simple thermal analysis, we found that the slots with higher temperature are exactly the same as the *present pages* in the kernel text. This result shows that in an KPTI-disabled system, all the present pages in the kernel can induce an TLB hit that will increase the core temperature. It allows ThermalBleed to identify the base address as well as the size of kernel text

It is notable that at the 478-th and 495-th slot in Fig.5, the temperature drastically surges to 57 $^{\circ}C$ and falls down to 46 $^{\circ}C$. We attribute this to a `sleep()` function used in the synchronization algorithm (Line 4 in ① in Table 2). The `sleep()` function is necessary to address the side effect caused by the remnant heat during the measurement. Considering the low sampling rate (i.e., 2 ms) of the digital thermal sensor, 20 ms of a sleeping interval in the algorithm is sufficient so that the measured temperature changes drastically.

C. EVALUATION

We evaluate the performance of the ThermalBleed attack under various target systems.

Experimental Environment: For systems equipped with 8-th or lower generations of Intel CPU, we enabled KPTI, which is the default mitigation feature in Linux against microarchitectural side-channel attacks. For other CPU models that have hardware fixes against a side-channel attack, KPTI is not applied to the system. The rest of the system

configurations are the same as in Section III-A. In the experiment, the collecting application obtains a trace of the core temperature by measuring 300 and 500 times on each slot. If not mentioned otherwise, the target application runs on the core 0, and the collecting application runs on farthest away from core 0.

1) BREAKING KASLR WITHOUT NOISE

We use ThermalBleed to break KASLR from an unprivileged user. In this experiment, we consider an ideal attack scenario: there is no system noise that may affect the temperature measurement during the attack.

In the first phase of the attack, we collect a trace of core temperature while a target application is running (i.e., issuing prefetch instructions to each KASLR slot). We use our synchronization algorithm (① in Table 2) to carefully obtain the temperature traces. After collecting the traces, we conduct a simple thermal analysis with the obtained temperature traces. In this experiment, we ran our attack 100 times on each KASLR slot. As a result, we successfully broke KASLR within 6 min with $T_n = 300$ and 9 min with $T_n = 500$ on all the target systems. The details on the experimental results are presented in Table 3. The term 'Accuracy' refers to the success ratio of the attacks among 100 trials. As shown in the table, ThermalBleed can successfully infer the base address of kernel text with 95% accuracy ($T_n = 500$) in all the target systems except the ones with certain CPU models for a laptop (i.e., Core i7-7200U and i7-10510U).

The low attack accuracy can be improved by simply issuing more prefetch instructions to induce more heat to the core. For instance, we could successfully de-randomize the KASLR within 18 min with $T_n = 1000$ on all the target systems including the laptops.

As CPUs have different frequencies and microarchitecture according to their model, the heat capacity and the resistance are likely to be different, which may affect the performance of ThermalBleed. Thus, for certain CPU models with low heat capacity and resistance, ThermalBleed can de-randomize KASLR with a higher attack accuracy. For instance, ThermalBleed can break KASLR with 100% accuracy ($n = 100$, $T_n = 500$) for Core i9-10900 and Xeon E3-1279 v6. However, ThermalBleed show 88% and 76% of accuracy for Core i7-7200U and i7-10510U, respectively ($n = 100$, $T_n = 500$).

2) BREAKING KASLR IN PRACTICAL SCENARIOS

In order to thoroughly evaluate the capability of ThermalBleed, we consider several practical scenarios such as attack with 1) noise from a different physical core, 2) noise from a different logical core sharing the same physical core, and 3) changes in P-state. We analyze the performance of ThermalBleed in each scenario.

a: NOISE FROM A DIFFERENT PHYSICAL CORE

We evaluate the performance of our attack under the case where an application that introduces a noise (i.e. increases the temperature) is running on a different physical core.

TABLE 3. Evaluation of ThermalBleed attack on various environments.

CPU	# of traces (T_n)	Accuracy	Time	ThermalBleed
Core i7-7200U	300	40%	6 min	✓
	500	88%	9 min	
Core i7-10510U	300	37%	6 min	✓
	500	76%	9 min	
Core i7-6700K	300	63%	6 min	✓
	500	96%	9 min	
Core i5-7400	300	63%	6 min	✓
	500	99%	9 min	
Core i7-8700	300	67%	6 min	✓
	500	90%	9 min	
Core i5-9600K	300	62%	6 min	✓
	500	93%	9 min	
Core i7-10700	300	60%	6 min	✓
	500	95%	9 min	
Core i9-10900	300	65%	6 min	✓
	500	100%	9 min	
Xeon E3-1279 v6	300	65%	6 min	✓
	500	100%	9 min	
Xeon Silver 4210	300	63%	6 min	✓
	500	91%	9 min	

As the `hwmon` interface allows an unprivileged user to obtain the temperature generated from each core, our attack is robust against the noise unless the noise-generating application is co-located with the target application. In order to validate our argument, we perform an experiment by using `stress-ng` as a noise source. Fig. 6 shows the accuracy of the ThermalBleed attack under the noise generated by running `stress-ng` with 80% of a CPU load. As expected, the accuracy is almost the same as the case without noise. Specifically, when `stress-ng` is running on the core 2 and 4, the attack accuracy reaches 100% ($T_n = 500$). For the core 6 and 8, we see a performance degradation of 1% and 2% ($T_n = 500$), respectively. However, when the `stress-ng` process is co-located with the target application, the accuracy decreases to 20% ($T_n = 500$) and 4% ($T_n = 300$).

b: NOISE FROM A DIFFERENT LOCAL CORE

The noise may also come from SMT(Simultaneous multi-threading). Thus, we consider this scenario to evaluate the robustness of ThermalBleed against the noise from SMT. Similar to the previous scenario, we use the `stress-ng` as a noise source for our experiment. In this setting, the target application and the `stress-ng` run on the same physical core (i.e., *core id 0*) while each is placed on different logical core. Then, the collecting application located at the other physical core, which is the farthest from the *core id 0*, obtains the temperature generated from the target application and `stress-ng`. Fig. 7 shows the experimental result. ThermalBleed has the highest accuracy of 91% ($T_n = 500$) when the `stress-ng` is running with 100% CPU load. However, with a 60% load, ThermalBleed has the lowest accuracy of 32% ($T_n = 500$). It is not clear why the accuracy shows a peak at the 100% CPU load. We assume

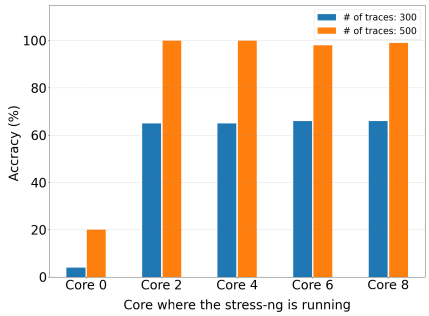

FIGURE 6. Noise from other physical core.

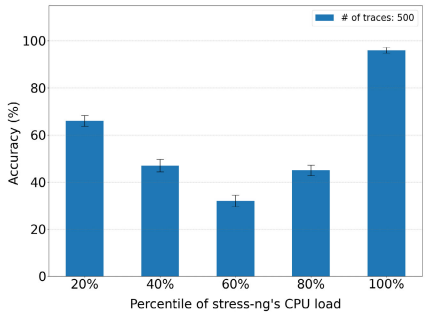

FIGURE 7. Noise from other local core.

that this result may originate from the thermal property regarding thermal capacity and resistance. We leave an in-depth analysis on this phenomenon to our future work.

c: CHANGES IN P-STATE

Intel introduced performance modes (i.e., P-state) to control voltage and frequency for efficient power management. In general, Intel provides two P-state modes, SpeedStep, and Speed Shift. Although both share the same goal regarding efficient power management, they slightly differ in their working. For SpeedStep, OS selects voltage and frequency based on current workload. For Speed Shift, however, it manages these elements based on Power Control Unit (PCU) [30].

P-state changes the power management behavior according to a system workload. Thus, it leads to variation in power consumption as well as CPU temperature (cf. Equation 1). We evaluate the ThermalBleed attack under different modes of P-state. Fig. 8 presents the accuracy of ThermalBleed under various P-state modes, which was measured on a system with Intel Xeon E3-1279 v6. Regardless of P-state modes (i.e., disabling P-state, SpeedStep and Speed Shift), the accuracy of ThermalBleed was approximately measured to be 98% ($T_n = 500$). However, we see a slight difference in the accuracy under the setting where the temperature was obtained through 300 traces. We attribute this to thermal properties such as thermal capacity and resistance, because these properties remained unchanged during the experiment while P-state modes were changed.

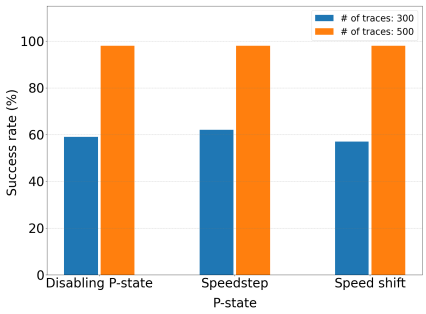

FIGURE 8. Changes in P-state.

To analyze the effect on the accuracy, we additionally explore the temperature for the 'TLB hit' under various P-state modes. In the experiment, the average temperature on the TLB hit was measured to be 61.45 $°C$ ($n = 10^5$, $\sigma = 1.40$) with Speed Step, 52.92 $°C$ ($n = 10^5$, $\sigma = 0.83$) with Speed Shift, and 53.24 $°C$ ($n = 10^5$, $\sigma = 0.73$) with disabled P-state. This result shows that the thermal capacity and resistance affect the accuracy of ThermalBleed.

V. IN-DEPTH ANALYSIS OF THERMAL SENSORS

It is intuitively acceptable that thermal sensors are not supposed to measure the temperature generated from outside the CPU die, and compute-intensive applications may cause higher temperatures than others. Hence, previous works [8], [11], [12], [28] explored the observed phenomenon without conducting a thorough analysis of the thermal sensor. This restricted the current research to a simple thermal covert channel, rather than a precise and effective thermal side-channel attack. We performed an in-depth analysis of Intel thermal sensors to promote further research on software-based thermal side-channel attacks. Specially, we study some thermal properties with the following questions.

Q1 Why does the physical memory access have a lower effect on the core temperature measured by the thermal sensor than the cache access?

Q2 Which elements in the CPU actually affect the temperature of the core?

To answer the first question, we analyzed the structure of the CPU package **(A1)**. For the second question, we uncovered the element's activity that is correlated with the core temperature by utilizing *Instruction Per Cycle (IPC)* **(A2)**.

A. A1: DISSECTING A STRUCTURE OF THE CPU PACKAGE

We investigate where the Intel digital thermal sensors are placed in the CPU package. Fig. 9 illustrates an internal structure and a longitudinal section of the CPU package [14]–[17]. A thermal interface material (TIM) is a compound material that transfers heat between the interfaces, facilitating thermal coupling. An integrated heat spreader (IHS) is a thin metal lid with high thermal conductivity, which protects the CPU die from external risk factors and provides an interface between the processor and heatsink (i.e., cooling device) for

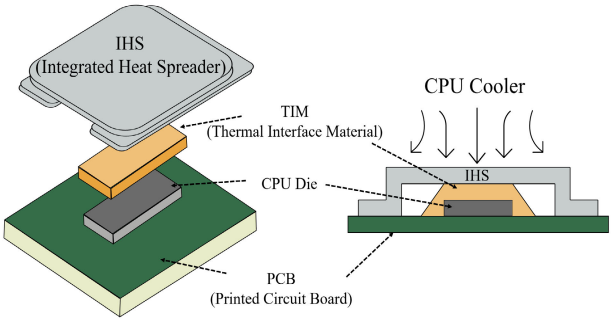

FIGURE 9. Structure of the CPU package.

efficient heat transfer. The CPU cooler cools down the CPU die from the heatsink to the IHS, TIM, and CPU die. In this structure, Intel digital thermal sensors were located in the CPU die. More specifically, the sensors were placed in *each core* to measure the heat generated from the core [13], [31]. Thus, the Intel digital thermal sensors directly retrieve the core temperature, not the temperature from the outside of the die (i.e., DRAM). For physical memory access to affect the core temperature, the heat generated from the DRAM should raise the air temperature. The hot air then disturbs the efficiency of the CPU cooler, indirectly affecting the CPU temperature. Thus, a load from a physical memory has a notably lower effect on the thermal sensors placed in the core than the load from a cache memory. Hence, Intel digital thermal sensors are more sensitive to cache access than physical memory access.

B. A2: EXPLORING PROPERTIES OF CPU THERMAL SENSORS

Because Intel digital thermal sensors are placed inside a CPU die, they are not for measuring the temperature outside the die. Thus, we can clearly distinguish a cache access (i.e., inside the die) from physical memory access (i.e., outside the die). Remarkably, we observe from Fig. 2 and 3 that a physical memory access results in higher temperature than the 'Baseline'. This means that even the physical memory access will slightly affect the core temperature. Thus, we conduct an analysis to determine what actually raises the CPU temperature.

We first hypothesized that the core temperature is correlated with the amount of current and voltage applied to the CPU. Our hypothesis is based on Joule's law of heating and Ohm's law. In particular, the Hamming weight of the operand value in an instruction is one of the factors that may affect the core temperature. It is originated from that the Hamming weight is directly related to the necessary amount of voltage for the CPU to execute instructions. Generally, the sum of the resistances of all the elements in the CPU that are involved in executing the instructions is constant. Hence, according to Ohm's law, the amount of voltage applied to the CPU was proportional to the amount of electric current. Therefore, the Hamming weight of the operand value actually affects the generation of heat from the CPU.

TABLE 4. Average core temperature and IPC on `imul` and `shl` instructions, varying its operand.

Operand	`imul r64`		Operand	`shl r64, imm8`	
(`r64`)	Temp.	IPC	(`imm8`)	Temp.	IPC
`Baseline`	27.046 °C	-	`Baseline`	27.885 °C	-
`0x00`	30.855 °C	0.67	`0x00`	31.759 °C	1.0
`0xff`	30.951 °C	0.67	`0x3f`	31.855 °C	1.0
`0xffff`	31.048 °C	0.67	`0x7f`	31.903 °C	1.0
`0xffffff`	31.336 °C	0.67	`0xbf`	31.951 °C	1.0
`0xffffffff`	31.567 °C	0.67	`0xff`	31.962 °C	1.0

Second, we hypothesized that the IPC is highly correlated with the core temperature. Actually, some previous works already discovered that there is a high correlation between power consumption and IPC [32]. As the power consumption is associated with the amount of heat based on Equation (1), we expect that the IPC may also have a strong correlation with the core temperature. Hence, we verify our hypothesis by observing the correlation between the core temperature and the IPC.

To suppress noise during the observation, we disabled the Intel Turbo Boost and fixed the CPU frequency as a base frequency. The heat propagation across CPU cores can also disturb the carrying out of a reliable experiment. Thus, we run our applications on different cores; the target application runs on the core 0, whereas the collecting application runs on the farthest away from core 0. The remnant heat may also impede the measurement of the temperature owing to the thermal capacity and resistance. Thus, we start from the instruction with the highest IPC in descending order, where we run the next instruction when the current core temperature is the same as the baseline temperature.

We first measured the core temperature and IPC for two instructions, "`imul r64`" and "`shl r64, imm8`". The values of those two operands vary from 0 × 00 to `0xffffffff`. Table 4 lists the measurements of the core temperature and IPC when issuing the instructions (i.e., `imul` and `shl`) on an i7-8700 processor. Because we disabled the Intel Turbo Boost and fixed the frequency at 3.2 GHz (i.e., base processor frequency), the value of IPC is constant over the operand values. An arithmetic instruction generally has a constant latency regardless of its operand value [33], [34], whereas the value affects the amount of voltage and current on the CPU. Thus, the temperature proportionally increases with respect to the Hamming weight of the operand value [35]. In an i7-8700 processor, "`imul` 0 × 00" and "`shl r64,` 0 × 00" instructions result in temperatures of 30.855 °C and 31.759 °C, respectively. In this case, the minimum amount of voltage and current was applied to the CPUs. The "`imul 0xffffffff`" and "`shl r64, 0xff`" instructions caused the highest temperature in the core thermal sensors.

TABLE 5. Average core temperature and IPC among various instructions.

Inst.	i7-10510U		i7-8700	
	Temp.	IPC	Temp.	IPC
`baseline`	38.717 °C	-	30.000 °C	-
`DRAM access`	43.047 °C	0.01	31.144 °C	0.01
`dTLB miss`	43.857 °C	0.05	31.288 °C	0.05
`aesenc xmm1, xmm0`	44.474 °C	0.50	31.663 °C	0.50
`imul rax, rbx`	44.759 °C	0.67	32.567 °C	0.67
`inc rax`	45.952 °C	2.0	34.673 °C	2.0
`xor rax, rbx`	46.472 °C	2.0	34.134 °C	2.0
`cache hit`	46.762 °C	2.0	34.989 °C	2.0
`dTLB hit`	47.000 °C	2.99	35.240 °C	2.99

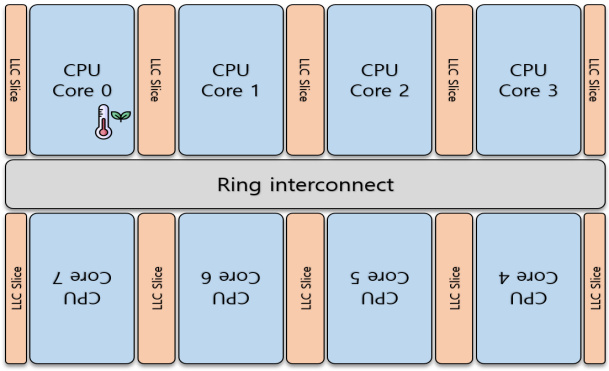

FIGURE 10. The layout of architectural components in i7-10700 CPU.

We performed an additional experiment to determine whether there is a correlation between the core temperature and IPC for various types of instructions. Table 5 shows the result of our additional experiment, where the core temperature tends to increase with respect to the value of IPC. Thus, there seems to be a degree of correlation between the temperature and IPC. We confirm this by calculating the Pearson correlation between the core temperature and IPC. The coefficients are 0.965 and 0.971 for the i7-10510U and i7-8700 processors, respectively. This implies that there is a high correlation between the temperature and IPC.

We observe from the experimental result that the heat generated from the core is correlated with the IPC. This supports that the heat generated by dTLB hits causes a higher temperature than dTLB misses. As shown in Table 5, there is a difference in the IPC between dTLB hits (IPC = 2.99) and misses (IPC = 0.05), which results in the difference of the core temperature. This property allows us to distinguish a dTLB hit from a miss, which is one of the primitives for the ThermalBleed attack.

It is noteworthy that there is an inconsistency between the core temperature and IPC in the experimental result. For instance, both "`inc rax`" and "`xor rax, rbx`" are the same kind of arithmetic instructions and they have the same IPC. However, the experimental result shows that there is a difference of about 0.593 °C between them. We attribute this inconsistency to the low resolution (i.e., ± 1°C) of the digital thermal sensor.

Despite the low resolution, the inconsistency can be alleviated by increasing the number of temperature measurements. For instance, with 100K measurements, the difference drastically drops off to 0.004°C between the "`inc rax`" and "`xor rax, rbx`". This result indicates that a side-channel attack that leverages the Intel digital thermal sensors needs to obtain more number of temperature traces to construct a reliable attack.

Based on the analysis, we observed that the core temperature is influenced by various factors of the instruction: the memory request type (i.e., to cache or DRAM), the Hamming weight of the operand value, and IPC. Hence, the actual amount of heat generated from the running application depends on these properties. Moreover, the temperature is highly correlated with the power consumption (cf. Section II-A). Thus, we infer that other side-channel analysis techniques, such as differential and correlation power analysis, are possible using the CPU thermal data. We expect that the explored thermal properties (i.e., **A1** and **A2**) can serve as stepping stones to construct more advanced thermal side-channel attacks.

VI. DISCUSSION

A. ATTACK ON CRYPTOGRAPHIC ALGORITHMS

In the previous section, we show that the ThermalBleed attack can successfully break KASLR from unprivileged user mode. In order to figure out whether ThermalBleed can be extended to another attack, we conducted an experiment that attempts to leak a secret key from a square-and-multiply RSA implementation. In the experiment, we did not observe any differences in the core temperature sufficient to distinguish executed instructions or memory access during RSA decryptions. We attribute this to the low measurement capacity (i.e., resolution and sampling rate) of the thermal interface. That is, the low resolution and sampling rate hinder us from distinguishing individual instructions and memory accesses.

However, if we consider privileged attackers in our attack model (e.g., targeting Intel SGX enclaves), we may overcome this challenging problem. For instance, a zero-stepping technique [3] allows us to execute a single instruction of SGX enclave at a time. With help of this technique, we can successfully distinguish individual operations, and ThermalBleed will be able to leak secret keys in cryptographic implementations.

B. HEAT PROPAGATION

Due to the heat propagation issue, the physical position of the target and collecting process on CPU cores may affect the attack performance. We conducted an additional experiment to analyze the effect of the physical location of

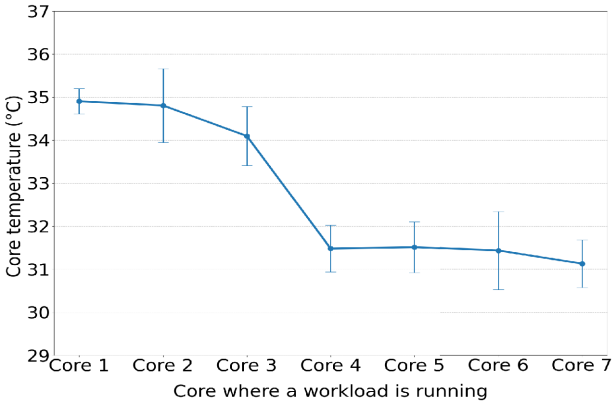

FIGURE 11. Temperatures on core 0 induced by heat propagation from other cores (Intel i7-10700).

TABLE 6. A comparison of microarchitectural side-channel attacks against KASLR.

CPU	Attack	Leakage source	Techniques	Timer
Intel	Hund *et al.* [25]	Data TLB	Page fault handling	✓
	Gruss, *et al.* [26]	Data TLB	Software-based prefetch	✓
	Jang *et al.* [24]	Data TLB	Intel TSX	✓
	Schwarz *et al.* [39]	Data TLB Store-to-load forwarding unit	Transient execution	✓
	Canella *et al.* [27]	Data TLB Meltdown fixed CPU	Transient execution	✓
	Lipp-(a) *et al.* [3]	Data TLB Power consumption on package	Transient execution	✗
	ThermalBleed	Data TLB temperature on core	Software-based prefetch	✗
AMD	Lipp-(b) *et al.* [40]	Data TLB Power consumption on core	Software-based prefetch	✗

these processes on our attack. Modern Intel processors use a ring-like core topology with a ring bus interconnection, where each core is structurally surrounded by the LLC slices and ring interconnect [36]–[38] as shown in Fig. 10. In the experiment, the collecting process is located at the core which is farthest away from core 0, and obtains 10K traces of temperatures measured at the core 0. At the same time, the target process runs repeatedly executing prefetch instructions to an allocated page at each core.

The experimental result is presented in Fig. 11. The figure shows how the temperature measurement at the core 0 is affected by the heat generated from other cores. We observe from the result that there is a noticeable difference between core 3 and 4. We attribute the difference to the physical topology of the CPU (cf. Fig. 10). In addition, the layout of the architectural components (e.g., cores, LLCs, and a ring interconnect) and the difference of the heat transfer rate among these components may also affect the result. We analyze from the result that we need to carefully choose the physical position of a target and collecting process considering the core topology and architectural components.

VII. RELATED WORK

In this section, we present some previous work related to the ThermalBleed attack.

A. MICROARCHITECTURAL SIDE-CHANNEL ATTACKS AGAINST KASLR

We compare some microarchitectural side-channel attacks to ThermalBleed in terms of the leakage source and attack techniques (cf. Table 6). To break KASLR, attackers should be able to distinguish a physically backed address from an invalid one. For this purpose, Hund *et al.* [25], Gruss *et al.*. [26], and Jang *et al.*. [24] exploited the replacement policy of data TLB (i.e., dTLB) on Intel CPUs. Specifically, if the target address of the data load is physically backed, the MMU allocates a TLB entry for PTE regardless of its user/supervisor bit. Otherwise, the MMU does not allocate a TLB entry (cf. Section III-C). All the attacks above exploit this property while using different techniques to handle an exception caused by accessing kernel addresses.

Specifically, Hund *et al.* [25] addressed the exception with a fault handling mechanism in the OS. Despite its simplicity, this technique has been shown to be time consuming compared to other approaches. Jang *et al.* [24] utilized Intel TSX technology to suppress the exception. In this technique, it only takes 5 min to de-randomize KASLR because it is resilient against any kind of measurement noise. However, as current CPUs no longer support TSX because of security issues, this technique is not available on the recent generations of Intel CPUs. Gruss *et al.* [24] leveraged a software-based prefetch instruction, which does not raise any exception even with a privileged target address. By leveraging the lack of privileged checks, their attack succeeded in circumventing KASLR in the 500 s.

In addition to using the dTLB, there are other ways to infer the present pages. Schwarz *et al.*. [39] exploited a store-to-load forwarding unit, where the stored data was only forwarded to the next load instruction if the destination address was successfully resolved in the address translation. Canella *et al.* [27] analyzed Meltdown-patched CPUs and disclosed how Intel fixed the CPUs against Meltdown attacks. More specifically, these CPUs immediately zeroed out if the illegally accessed address is the present page; otherwise, a pipeline stall occurs. This attack exploits the timing difference to break KASLR.

All these previous works commonly depend on precise timing information to deliver the attack. This implies that they are easily mitigated by a defense approach that limits the timing information [9], [10]. However, ThermalBleed is a *timer-free* side-channel attack; it does not rely on any timing information to de-randomize KASLR. Therefore, our attack is more resistant to defense mechanisms compared to previous studies.

Similar to ThermalBleed, Lipp-(a), (b) *et al.* [3], [40] presented another timer-free side-channel attack; they exploited information from the CPU power consumption using the `powercap` interface without relying on the timer information. However, Lipp-(a) *et al.* [3] was limited in that it did

TABLE 7. A comparison of thermal-based attacks.

	Measurement	Attack goal	Target platform
Hutter *et al.* [35]	Hardware	Side-channel	AVR
Aljuffri *et al.* [41]	Hardware	Side-channel	AVR
Masti *et al.* [8]	Software	Covert-channel	x86
Bartolini *et al.* [12]	Software	Covert-channel	x86
Long *et al.* [11]	Software	Covert-channel	x86
ThermalBleed	Software	Side-channel	x86

not allow cross-core side-channel attacks. Moreover, after patching the vulnerability, the powercap interface was no longer available to user mode applications (i.e., unprivileged users), which significantly reduced the effectiveness of the attack. Thus, the ThermalBleed attack is the only one timer-free thermal side-channel attack that is effective and practical in various environments.

B. THERMAL-BASED ATTACKS

Previous works on thermal-based attacks can be classified according to some criteria; an attack goal (i.e., constructing a covert- or a side-channel), a measurement method (i.e., software- or hardware-based), and a target platform (i.e., x86, AVR, etc). Table 7 presents a comparison of thermal-based attacks according to the criteria.

Hutter *et al.* [35] presented a heating fault attack on an AVR-based target device. By exploiting the fact that a fault is induced if the device temperature reaches a threshold, they successfully recovered a RSA private key via a side-channel analysis. Aljuffri *et al.* [41] improved previous hardware-based thermal side-channel attacks by applying power analysis techniques such as SPA and CPA. As a result, they successfully extracted a private key of an Montgomery ladder implementation of RSA algorithm with 100% accuracy on AVR-based devices. As shown in Table 7, all the aforementioned thermal side-channel attacks are based on hardware-based measurement. As hardware-based attacks basically require an attacker to have physical access to the target device, their attack models are restricted to limited and unpractical attack scenarios. Unlike the previous work, the ThermalBleed attack is based on a software-based temperature measurement, which eliminates the need for physical access to a target device.

On the other hand, there are other works that use software-based thermal measurements, however, they are only limited to constructing covert-channel attacks. Masti *et al.* [8] observed two thermal properties, heat propagation and remnant heat, via Intel digital thermal sensor. Based on the observation, they implemented two variants of covert-channel attacks on spatially and temporally partitioned x86 multi-core systems. Bartolini *et al.* [12] and Long *et al.* [11] improved the performance of Masti *et al.*'s attack by enhancing the covert-channel efficiency in terms of the bit error rate. However, due to the inherently low resolution of software-based temperature measurement, their techniques are also limited to constructing coarse-grained covert-channel attacks. We overcome the limitations and successfully implement ThermalBleed, the first thermal side-channel attack on a x86 system with a software-based measurement.

VIII. CONCLUSION

In this paper, we analyzed Intel digital thermal sensors and their properties. Our analysis showed that the sensors expose the current CPU temperature through an unprivileged `hwmon` interface, and the core temperature is highly correlated with different types of memory load operations, i.e., the load from the cache and that from the DRAM. Based on the analysis, we presented ThermalBleed, a practical thermal side-channel attack that exploits the Intel digital thermal sensors. We demonstrated that ThermalBleed can de-randomize the kernel addresses from unprivileged user mode applications by distinguishing cache access from physical memory access. Furthermore, the proposed attack can be delivered on various classes of Intel CPUs. Moreover, we identified several useful but unexplored thermal properties for future studies. We believe that these properties can serve as a stepping stone to construct more advanced thermal side-channel attacks.

The vulnerability that the ThermalBleed attack exploits attributes to a security issue in the `hwmon` interface, i.e., *allowing unprivileged access* to the interface. Thus, we propose that the interface should be restricted only to privileged users to secure the system.

REFERENCES

[1] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 27–36.
[2] M. Alagappan, J. Rajendran, M. Doroslovački, and G. Venkataramani, "DFS covert channels on multi-core platforms," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2017, pp. 1–6.
[3] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "PLATYPUS: Software-based power side-channel attacks on x86," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 355–371.
[4] Z. Zhang, S. Liang, F. Yao, and X. Gao, "Red alert for power leakage: Exploiting Intel RAPL-induced side channels," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2021, pp. 162–175.
[5] Intel. (2020). *Running Average Power Limit Energy Reporting/CVE-2020-8694, CVE-2020-8695/INTEL-SA-00389*. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/software-% security-guidance/advisory-guidance/running-average-power-limit-energy-reporti% ng.html
[6] D. Gruss, M. Lipp, M. Schwarz, R. Fellner, C. Maurice, and S. Mangard, "KASLR is dead: Long live KASLR," in *Proc. Int. Symp. Eng. Secure Softw. Syst. (ESSoS)*, vol. 10379, 2017, pp. 161–176.
[7] Intel. (2019). *Deep Dive: Intel Analysis of Microarchitectural Data Sampling*. [Online]. Available: https://software.intel.com/content/www/us/en/develop/topics/software-se% curity-guidance.html
[8] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 865–880.
[9] W.-M. Hu, "Reducing timing channels with fuzzy time," *J. Comput. Secur.*, vol. 1, nos. 3–4, pp. 233–254, Oct. 1992.
[10] D. Kohlbrenner and H. Shacham, "Trusted browsers for uncertain times," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 463–480.
[11] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, and T. Mak, "Improving the efficiency of thermal covert channels in multi-/many-core systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1459–1464.

[12] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores," in *Proc. 11th Eur. Conf. Comput. Syst.*, Apr. 2016, pp. 1–16.
[13] *Intel Xeon Processor E5-2400 Product Family*, Intel, Mountain View, CA, USA, 2012.
[14] *Intel Xeon Processor Scalable Family*, Thermal Mech. Specifications Des. Guide, Intel, Mountain View, CA, USA, 2019.
[15] Intel. (2021). *Intel CPU Temperature Guide*. [Online]. Available: https://forums.tomshardware.com/threads/intel-cpu-temperature-guide-202% 1.1488337/
[16] Intel. (2021). *CPU Cooler: Liquid Cooling Vs. Air Cooling*. [Online]. Available: https://www.intel.com/content/www/us/en/gaming/resources/cpu-cooler-liquid-cooling-vs-air-cooling.html
[17] Intel. *Desktop 4th Generation Intel CoreT Processor Family: Thermal Guide*. Thermal Mech. Des. Guidelines (TMDG), 2013.
[18] E. Mach, *History Root Principle Conservation Energy*. Chicago, IL, USA: Open Court Publishing Company, 1910.
[19] A. Von Meier, *Electric Power Systems: A Conceptual Introduction*. Hoboken, NJ, USA: Wiley, 2006.
[20] *Intel 64 and IA-32 Architectures Software Developer Manuals*, Volume 3 (3A, 3B, 3C & 3D): Syst. Programming Guide, Intel, Mountain View, CA, USA, 2020.
[21] G. Roeck. (2020). *Linux Hwmon Documentation*. [Online]. Available: http://blog.foool.net/wp-content/uploads/linuxdocs/hwmon.pdf
[22] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 552–561.
[23] N. Carlini and D. Wagner, "ROP is still dangerous: Breaking modern defenses," in *Proc. 23th USENIX Secur. Symp.*, 2014, pp. 385–399.
[24] Y. Jang, S. Lee, and T. Kim, "Breaking kernel address space layout randomization with Intel TSX," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 380–392.
[25] R. Hund, C. Willems, and T. Holz, "Practical timing side channel attacks against kernel space ASLR," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2013, pp. 191–205.
[26] D. Gruss, C. Maurice, A. Fogh, M. Lipp, and S. Mangard, "Prefetch side-channel attacks: Bypassing SMAP and kernel ASLR," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 368–379.
[27] C. Canella, M. Schwarz, M. Haubenwallner, M. Schwarzl, and D. Gruss, "KASLR: Break it, fix it, repeat," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 481–493.
[28] T. Claeys, F. Rousseau, B. Simunovic, and B. Tourancheau, "Thermal covert channel in Bluetooth low energy networks," in *Proc. 12th Conf. Secur. Privacy Wireless Mobile Netw.*, May 2019, pp. 267–276.
[29] D. Gruss, D. Bidner, and S. Mangard, "Practical memory deduplication attacks in sandboxed Javascript," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2015, pp. 108–122.
[30] E. Rotem, "Intel architecture, code name Skylake deep dive: A new architecture to manage power performance and energy efficiency," *Intel Developer Forum*, vol. 24, pp. 1–43, Aug. 2015.
[31] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, S. Kottapalli, and S. Vora, "A 45 nm 8-core enterprise Xeon processor," *IEEE J. Solid-State Circuits*, vol. 45, no. 1, pp. 7–14, Jan. 2010.
[32] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst. (SIGMETRICS)*, 2003, pp. 160–171.
[33] A. Abel and J. Reineke, "Uops.info: Characterizing latency, throughput, and port usage of instructions on Intel microarchitectures," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Oper. Syst. (ASPLOS)*, Apr. 2019, pp. 673–686.
[34] A. Abel and J. Reineke, "NanoBench: A low-overhead tool for running microbenchmarks on x86 systems," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Aug. 2020, pp. 34–46.
[35] M. Hutter and J.-M. Schmidt, "The temperature side channel and heating fault attacks," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, A. Francillon and P. Rohatgi, Eds., 2013, pp. 219–235.
[36] S. Junking. (2015). *The Compute Architecture of Intel Processor Graphics Gen9*. [Online]. Available: https://www.intel.com/content/dam/develop/external/us/en/documents/the-% compute-architecture-of-intel-processor-graphics-gen9-v1d0.pdf
[37] J. Turley. (2014). *White Paper: Introduction to Intel Architecture*. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/white-pape% rs/ia-introduction-basics-paper.pdf
[38] D. Schor. (2020). *Intel Launches 10th Gen Comet Lake Desktop Processors*. [Online]. Available: https://fuse.wikichip.org/news/3466/intel-launches-10th-gen-comet-lake-% desktop-processors/
[39] M. Schwarz, C. Canella, L. Giner, and D. Gruss, "Store-to-leak forwarding: Leaking data on meltdown-resistant CPUs (updated and extended version)," 2019, *arXiv:1905.05725*.
[40] M. Lipp, D. Gruss, and M. Schwarz, "AMD prefetch attacks through power and time," in *Proc. 31th USENIX Secur. Symp.*, 2022, pp. 1–18.
[41] A. Aljuffri, M. Zwalua, C. R. W. Reinbrecht, S. Hamdioui, and M. Taouil, "Applying thermal side-channel attacks on asymmetric cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 11, pp. 1930–1942, Nov. 2021.

TAEHUN KIM received the B.S. degree in computer and information engineering from Kwangwoon University, Seoul, South Korea, in 2021. He is currently pursuing the master's degree in cybersecurity with Korea University. His research interests include system security and CPU micro-architectural security.

YOUNGJOO SHIN received the B.S. degree in computer science and engineering from Korea University, Seoul, South Korea, in 2006, and the M.S. and Ph.D. degrees in computer science from KAIST, Daejeon, South Korea, in 2008 and 2014, respectively. From 2008 to 2017, he was with the National Security Research Institute (NSR), Daejeon, as a Senior Researcher. From 2017 to 2020, he was with Kwangwoon University, Seoul, as an Assistant Professor. From 2020 to 2021, he was with Korea University, as an Assistant Professor. He is currently an Associate Professor with the School of Cybersecurity, Korea University. His research interests include system and network security, CPU micro-architectural security, cloud computing security, and vulnerability analysis on embedded systems.