A Lightweight Delay-based Authentication Scheme for DMA Attack Mitigation

Yutian Gui, Ali Shuja Siddiqui, Geraldine Shirley Nicholas, Marcus Hughes, Fareena Saqib
University of North Carolina at Charlotte, NC, USA
ygui@uncc.edu, asiddiq6@uncc.edu, gnichola@uncc.edu, mhughe35@uncc.edu, fsaqib@uncc.edu

2021 22nd International Symposium on Quality Electronic Design (ISQED) | 978-1-7281-7641-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISQED51717.2021.9424262

***Abstract*—With the extensive application of the Direct Memory Access (DMA) technique, the efficiency of data transfer between the peripheral and the host machine has been improved dramatically. However, these optimizations also introduce security vulnerabilities and expose the process of data transmission to DMA attacks that utilize the feature of direct access to steal the data stored in the live memory on the victim system. In this paper, we propose a lightweight scheme to provide resilience to DMA attacks without physical and protocol-level modification. The proposed scheme constructs a unique identifier for each DMA-supported PCIe device based on profiling time and builds a trusted database for authentication. The experimental result shows that the proposed methodology eliminates most of the noise produced in the measuring process for identifier construction and the success rate of authentication is 100% for all the devices.**

Keywords—direct memory access, DMA attack, side-channel attack, hardware security

I. Introduction

Traditionally, the data transmission between the external device and the main memory was realized by using programmed input/output or interrupt-driven I/O which require the full involvement of the CPU. The CPU has to transfer the data word to word which is inefficient especially when the size of data to be transferred is large. DMA allows peripheral devices to read/write data from/to the main memory without passing it through the processor. In DMA protocol, the CPU only takes charge of initialization and termination which frees the processor from involvement in the transfer process. Nowadays, DMA is widely supported by different data buses (ISA, PCIe, etc.) in modern computing systems.

With the extensive application of DMA, security issues have become inevitable. In recent works, the vulnerability of DMA has been explored and demonstrated [1] [2]. The characteristic of direct access allows the attacker to steal data and mount the file system of the victim system through DMA-supported devices maliciously. To reduce the risk of DMA attacks, countermeasure techniques were introduced using memory virtualization and key-based authentication [3] [4], but are either expensive or unfeasible. Furthermore, recent attacks show that the DMA-based data transfer is still vulnerable even with security features enabled [5] [6].

In this work, we propose a lightweight authentication scheme for PCIe devices. To the best of our knowledge, this work is the first publicly reported study that utilizes the profiling time to construct identifiers used in authentication for PCIe devices.

Contribution: This paper makes the following contributions:

1. We propose a comprehensive scheme for DMA attack mitigation, including the registration process and the authentication process.

2. We design a framework to remove noise in collected measurements for identifier construction.

3. The effectiveness and feasibility of the proposed design are verified by experiments.

Paper organization: The paper is organized as follows. The related work is discussed in section II. Section III shows the detail of the attack model. Section IV describes the proposed methodology and section V presents the experimental setup and results. The security analysis is discussed in section VI.

II. Related Works

A. Direct Memory Access (DMA)

In the traditional computing system, the CPU reads every block of data using a peripheral bus from the I/O devices and writes it into the main memory. In this process, the CPU is fully occupied and is thus not available to perform other works. This structure slows down the performance of the system significantly. In contrast, DMA allows peripheral hardware devices (disk drive controllers, graphics cards, network cards, and sound cards, etc.) to send/read I/O data directly to/from main memory. With DMA, the interaction between external devices and memory is carried out independently of the CPU, therefore the overload of CPU is reduced remarkably.

The feature of DMA is provided by several bus architectures, such as Industry Standard Architecture (ISA), Advanced Microcontroller Bus Architecture (AMBA), and Peripheral Component Interconnect (PCI). To manage the data transfer between the host system and DMA devices, a DMA controller is needed. The DMA controller is a control unit, part of the interface circuit, which enables the movement of data blocks between I/O devices and the main memory. After the initialization of the DMA controller by CPU, the memory controller provides memory addresses and initiates read or write cycles for data transfer, and sends an interrupt to the CPU when the whole process of data transmission is done.

B. DMA attack

DMA increases the efficiency of data transfer between the main memory and external devices, but the native feature of direct access also brings some potential risks of security breaches. As a type of side-channel attack, the DMA compromise has been proved as a powerful and efficient attack that allows the attacker to read and write the memory on the victim system directly.

In 2016, [1] demonstrated a DMA attack that allows the attacker to read/write the memory once the peripheral PCIe device is connected to the victim system without the need for hardware drivers. Moreover, this attack is able to access the live RAM and the file system by inserting kernel implants. The same year, the Intel Advanced Threat Research team performed a DMA attack over the air by modifying a WiGig dock to compromise a laptop that is connected to the dock wirelessly [2]. The architecture of wireless connection allows the attacker to use the DMA capabilities to dump secrets out of the memory on the victim machine remotely. The work reported in [6] shows the Thunderbolt protocol with access-control enabled is not resilient to DMA attacks. The attacker-controlled device obtains full access to the main memory successfully via a Thunderbolt port by identity clone and spoofing.

C. DMA attack mitigation

To mitigate the risk of DMA attacks, some techniques have been proposed. After BitLocker was first introduced in Windows Vista, Microsoft provides pre-boot authentication to enhance data privacy. The safety of data is ensured by full-disk encryption with BitLocker and the key is generated by the Trusted Platform Module (TPM) [7]. According to the policy of pre-boot authentication, BitLocker accesses and stores the encryption key in memory only after the user provides the correct PIN or USB startup key. However, BitLocker can only be used for encrypting hard drives or removable data drives, the memory is still vulnerable to DMA attacks after the booting process is completed. Moreover, the time overhead of encryption/decryption is very high.

For Windows 10, a new feature is added from version 1709 named "Disable new DMA devices when this computer is locked" [3]. Once this feature is enabled, all the hot-pluggable PCI/PCIe ports will be blocked until a user signs in to Windows. This policy prevents attacks that use PCI/PCIe-based devices to access BitLocker keys, yet at the same time brings inconvenience in some cases when the user wants to keep PCI/PCIe devices working when the system is locked.

With the introduction of the Input-Output Memory Management Unit (IOMMU) (This technique is branded VT-d [4] by Intel and AMD-Vi [8] by AMD), the risk of DMA attack is reduced. IOMMU connects the DMA-capable I/O bus to the main memory and supports DMA-remapping which translates the address of the incoming DMA request to the correct physical memory address. After activating this function, each DMA device can only access a part of the memory which is allocated by IOMMU therefore the rest of the memory is immune to DMA attacks on that device. This technique mitigates the risk of DMA attack, but it cannot protect the whole memory especially when the memory size of the system is small.

Another strategy of DMA attack mitigation is access control which can be realized by encryption-based authentication or Trust-On-First-Use (TOFU) scheme [9]. [10] presents an authentication architecture between each component and the host system based on the key-based certification. However, the precondition of this design is that each component must be authenticatable at any time which is not practical. For the TOFU scheme, the host machine records the unique fingerprint/identifier of a trusted device into its trusted database the first time this device is attached. In this way, a device can access the host system only if its identifier can be found in the trusted database otherwise it will be blocked. However, there is no available unique identifier for devices in the case of PCIe authentication. Currently, computing systems use vendor ID, device ID, and PCIe slot ID to distinguish different PCIe devices. However, if the attacker unplugs the connected PCIe device and inserts a new device of the same model from the same manufacturer into the same slot, the host machine is not able to recognize whether the new connected device is the same as the one used previously.

In this work, we use the profiling time of each device as the identifier to build the trusted database on the host machine. Compared with existing work, our design is lightweight and does not require any physical modification or protocol-level modification on the host machine.

III. Attack Model

As the successor of PCI, PCIe is a high-speed serial computer expansion bus standard for attaching hardware devices to a computer. In a PCIe system, endpoints are connected with the memory subsystem via the interface named root complex. Once the PCIe device is connected, the communication is achieved by exchanging Transaction Layer Packets (TLPs) which relate to PCIe's uppermost layer between the PCIe device and the host. This standard does not implement any security to protect the privacy of data which makes the DMA attack feasible on the PCIe system.

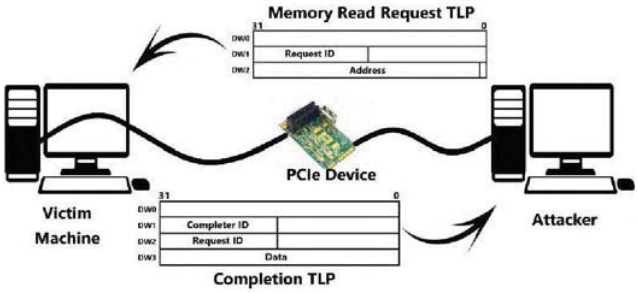

Fig. 1. Basic DMA attack flow.

In this work, we perform an attack using a PCIe device to show the vulnerability of DMA. The attack presented in this work references the attack model presented in [1] and [11]. Fig. 1 shows the basic attack flow. The PCIe device is connected to the victim machine via the PCIe port and connected to the attacker machine by any of the interfaces (USB, Ethernet, etc.). The attacker controls the PCIe device to send a Memory Read Request (MRd) TLP to the victim machine. Once the MRd TLP reaches the PCIe root complex, the victim machine will respond

with the completion TLP which contains actual data back to the PCIe device, then back to the attacker. The attacker can either dump all the data stored in the main memory of the victim machine, or a part of the data using a specific range of memory addresses.

To configure the attack, we use a PCIe-compatible development board named "NeTV2" with an onboard Xilinx XC7A35T FPGA chip. Xilinx provides PCIe DMA and PCIe Bridge hard and soft IP blocks for compatible FPGA devices [12], as well as full access to 64-bit memory space without relying on a kernel module running on the victim system.

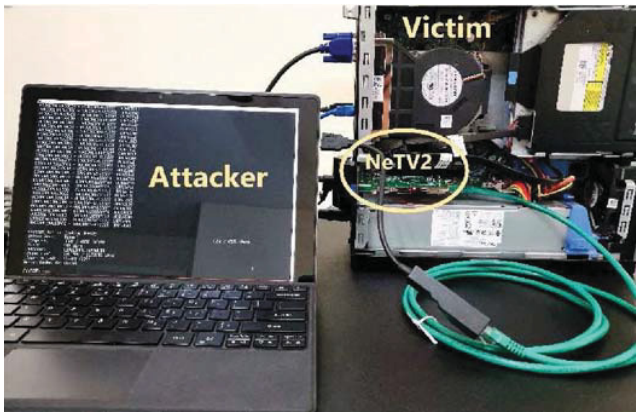

Fig. 2. Experimental setup of DMA attack.

Fig. 2 shows the experimental setup of the proposed DMA attack. The NeTV2 board is connected with the victim machine via PCIe port, and connected with the attacker by an Ethernet cable. There is no need to install any hardware driver on the victim machine. The direct access supported by DMA enables the attacker to control the NeTV2 board to send memory read/write request TLPs to the victim machine, thereby read/write data from/to the main memory on the victim maliciously.

```
E:\PCILeech>pcileech probe -device rawudp://ip=192.168.0.222 -v

DEVICE: FPGA: NeTV2 RawUDP PCIe gen2 x1 [0,0,0] [v4.2,0200]
 Memory Map:
 START              END                #PAGES
 0000000000000000 - 000000000009ffff  000000a0
 00000000000e0000 - 000000006dffffff  0006bd40
 0000000100000000 - 0000000165fb5fff  00065fb6
 0000000165fc0000 - 000000095e9bbfff  007f89fc
 000000095ec00000 - 000000108dffffff  0072f400
 Current Action: Probing Memory
 Access Mode:    Normal
 Progress:       67808 / 67808 (100%)
 Speed:          858 MB/s
 Address:        0x000000108E000000
 Pages read:     16751506 / 17358848 (96%)
 Pages failed:   607342 (3%)
Memory Probe: Completed.
```

Fig. 3. Memory probing on the victim machine.

Fig. 3 shows the process of memory probing. The attacker enumerated the memory of the target system for readable memory pages starting from the first address to the last one. As shown in Fig. 3, around 96% of the whole memory space is readable in this case.

```
011EAFCC40  00 00 5E EE 52 A0 5F 29 A6 43 7B 22 64 69 73 70  ..^îR _)¦C{"disp
011EAFCC50  6C 61 79 54 65 78 74 22 3A 22 47 6F 6F 67 6C 65  layText":"Google
011EAFCC60  20 43 68 72 6F 6D 65 22 2C 22 61 63 74 69 76 61   Chrome","activa
011EAFCC70  74 69 6F 6E 55 72 69 22 3A 22 6D 73 2D 73 68 65  tionUri":"ms-she
011EAFCC80  6C 6C 61 63 74 69 76 69 74 79 3A 22 2C 22 61 70  llactivity:","ap
011EAFCC90  70 44 69 73 70 6C 61 79 4E 61 6D 65 22 3A 22 47  pDisplayName":"G
011EAFCCA0  6F 6F 67 6C 65 20 43 68 72 6F 6D 65 22 2C 22 62  oogle Chrome","b
011EAFCCB0  61 63 6B 67 72 6F 75 6E 64 43 6F 6C 6F 72 22 3A  ackgroundColor":
011EAFCCC0  22 62 6C 61 63 6B 22 7D 03 69 50 41 6B 6A 62 4C  "black"}.iPAkjbL
```

Fig. 4. A memory fragment dumped from the victim machine.

Fig. 4 shows a part of live memory dumped from the victim machine. This memory fragment was being used by the Google Chrome browser when the attacker was dumping the main memory illegally.

IV. Proposed Methodology

To construct a unique identifier for each PCIe device, the profiling time is used in this work. This unique identifier serves as a fingerprinting of each device. Due to the variation in the manufacturing process, each device has a unique physical characteristic such as current flow, IR drop, threshold voltage, and unique delays. These delays affect the response time of the device and have a high stability (drift due to aging is not considered in this work). Manufacturing variations are also used in creating cryptographic functions such as Physical Unclonable Functions (PUFs).

Before building the trusted database with the profiling time of each device, one issue is the high variance among multiple samples which makes the raw data unusable. This variance is caused by multiple reasons, such as the change of environmental parameters and real-time resource utilization. In this work, we design a data processing framework to reduce the noise in raw measurements for extracting accurate and stable profiling time.

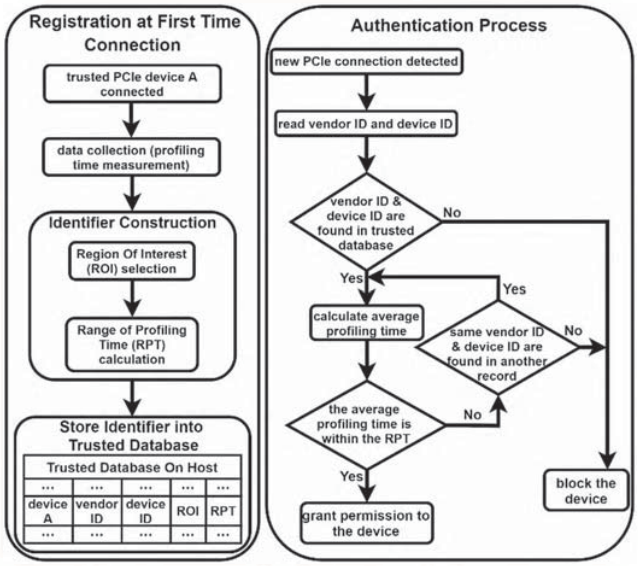

Fig. 5. Proposed registration and authentication scheme.

Fig. 5 shows the proposed scheme which consists of two main parts: registration and authentication.

A. Trusted Device Registration

In the registration process, when a trusted PCIe device is first time connected to the host machine, the authorized user

measures the profiling time of this device, constructs an identifier with ROI selection algorithms for this device, and stores the identifier (Vendor ID, Device ID, Region of Interest (ROI) and Range of Profiling Time (RPT)) into the trusted database.

1) Data collection

For each device, multiple sub-datasets are collected, and each sub-dataset contains an equal number of profiling time measurements. Let R denote the number of sub-datasets for each device, and S denotes the number of recorded measurements of profiling time in each sub-dataset. Totally, there are $(R \times S)$ measurements for each device. After data collection, the raw measurements of profiling time are sorted from lowest to highest in each sub-dataset.

2) Region Of Interest (ROI) selection

In data processing, ROI is a group of samples selected from a dataset for a particular purpose. In this work, the goal of ROI selection is to find the region in the sub-dataset that has the lowest noise. Since the variance in raw measurements is too high, we propose two different algorithms to seek the most valuable ROI in the sorted sub-dataset.

a) Difference based algorithm

In the difference based algorithm, we calculate the difference of the same selected regions between each pair of sub-datasets from the first measurement to the last one. Let L denote the length of the Difference-based ROI (DROI). For R sub-datasets (each sub-dataset has S measurements of profiling time), there are $C(R, 2)$ different combinations of comparison pairs, and $(S - L)$ possible regions. Algorithm 1 gives the detail of DROI selection.

Algorithm 1: DROI selection

Inputs:
- Number of sub-datasets (R), Number of measurements in each sub-dataset (S), Length of DROI (L)

Output:
- Difference-based ROI (DROI)

Steps:

```
Res[] ← empty list
for i = 1 to (S − L)                    //for all the possible regions
    for j = 1 to (R − 1)
        for k = (j + 1) to R            //for all the combinations
            calculate AD (absolute difference value between sub-dataset[j][i : i + L] and sub-dataset[k][i : i + L])
            TD (accumulative difference for region i) ← TD + AD
        end for
    end for
    Res[i] ← TD
    TD ← 0
end for
DROI ← [d : d + L] where d is the index of the minimum value in Res[]
```

b) Correlation coefficient based algorithm

The correlation coefficient is a statistical index used to measure the dependency of two variables. If the same regions from different sub-datasets have the highest correlation with each other, it means that the noise level in this region is the lowest. In this work, we compute the Pearson's correlation coefficient of all the regions from each pair of sub-datasets to find the Correlation-based ROI (CROI). The detail of CROI selection is shown in Algorithm 2.

Algorithm 2: CROI selection

Inputs:
- Number of sub-datasets (R), Number of measurements in each sub-dataset (S), Length of CROI (L)

Output:
- Correlation-based ROI (CROI)

Steps:

```
Res[] ← empty list
for i = 1 to (S − L)                    //for all the possible regions
    for j = 1 to (R − 1)
        for k = (j + 1) to R            //for all the combinations
            calculate CC (correlation coefficient between sub-dataset[j][i : i + L] and sub-dataset[k][i : i + L])
            TC (accumulative coefficient for region i) ← TC + CC
        end for
    end for
    Res[i] ← TC
    TC ← 0
end for
CROI ← [d : d + L] where d is the index of the maximum value in Res[]
```

In DROI selection, the lowest value in result list Res[] is used because the lowest difference means the same selected regions from each pair of sub-datasets have the highest similarity (in other words, the lowest noise). However, in CROI selection, the highest value in result list Res[] is selected because the highest coefficient represents the highest similarity. At the end of the selection process, the overlapped region between the DROI and CROI is selected as the ROI for identifier construction.

3) Construct and store identifier

The process of ROI selection figures out the region with the lowest level of noise. By calculating the average profiling time of all the measurements in ROI for each sub-dataset and combining the results of all the sub-datasets, we get the Range of Profiling Time (RPT) which can be used as a part of the identifier. The ROI and RPT will be stored in the trusted database on the host machine, along with the vendor ID and device ID of this trusted device.

B. Authentication

For authentication, every time a new PCIe connection is detected, the system reads the vendor ID and device ID of this PCIe device. Once the vendor ID and device ID are found in the trusted database, the host machine will collect a number of profiling time measurements and calculate the average profiling time based on the ROI stored in the same record. If the average profiling time of this device is within the RPT stored in the same record, the permission will be granted to the device, otherwise the system will recheck for other records that contain the same vendor ID and device ID (in case of more than one device of the same model are registered) until all the records have been traversed.

V. Experimental Setup and Result

In this work, three TL-WN881ND wireless PCIe adapters from TP-LINK [13] that have the same properties (called device A, device B, and device C) are used for verifying the proposed design. In data collection, the elapsed time of reading configuration space on the PCIe device is measured as profiling time.

In experiments, we measured the profiling time of each device 10000 times and repeated this process 30 times under the

same conditions. After data collection, we have 30 sub-datasets and each sub-dataset has 10000 measurements of profiling time for each device.

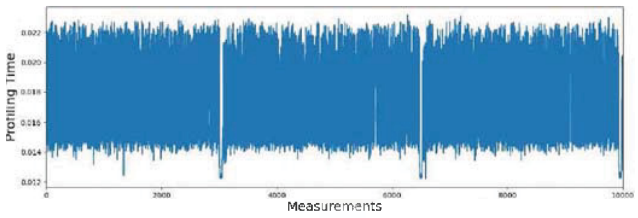

Fig. 6. All 10000 measurements of profiling time in one sub-dataset collected from device A.

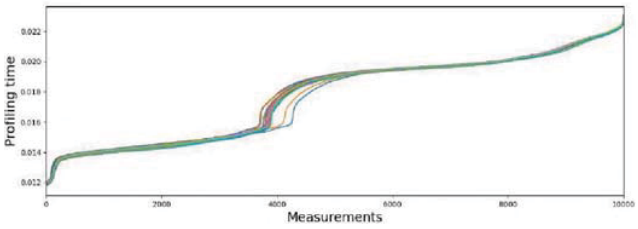

Fig. 7. Comparison of all 30 sorted sub-datasets collected from device A.

Fig. 6 shows all 10000 measurements of profiling time in one sub-dataset collected from device A, and Fig. 7 shows all 30 sub-datasets with sorted measurements collected from device A. Without data processing, the variance of profiling time is very high (from 12 ms to 23 ms), and the difference among different sub-datasets is also very high in some regions which makes identifier construction difficult. Then, we applied the two proposed algorithms of ROI selection on the raw measurements collected from device A. In this case, we set the length of DROI and CROI to 3000 (to balance the uniqueness and the applicability, the length of CROI/DROI should be between 20% and 40% of the sub-dataset's size). Fig. 8 and Fig. 9 show the result of DROI selection and CROI selection.

```
Min Difference:  0.02302205618222555
Difference based ROI (DROI):  5464 - 8464
>>>
Max correlation:  434.7485805839838
Correlation based ROI (CROI):  6991 - 9991
>>>
```

Fig. 8. The output of DROI selection and CROI selection.

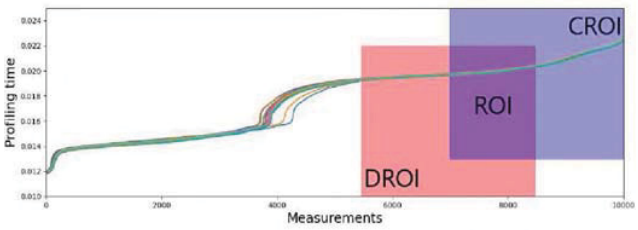

Fig. 9. Result of DROI selection and CROI selection.

The result shows that the DROI of device A is from the 5464th point to the 8464th point in each sorted sub-dataset, and the CROI is from the 6991st point to the 9991st point. We selected the overlapped region of DROI and CROI as ROI (6991st - 8464th). This region has the lowest noise and the highest stability because samples in this region from data collections over different periods (sub-datasets in this case) have both the lowest difference and the highest correlation with each other. Moreover, based on the result of comparison among three devices, we noticed that all the devices of the same type have very similar ROIs with each other, so we used the ROI of device A for all three devices to construct the identifier.

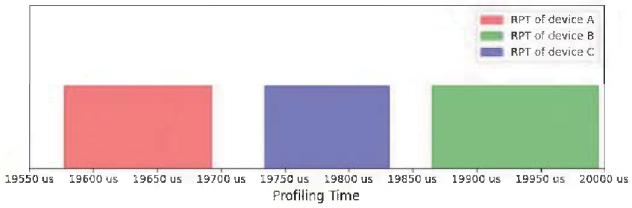

Fig. 10. Range of Profiling Time (RPT) of each device.

For each device, we calculated the average profiling time of all the measurements in ROI in each sub-dataset. The average profiling times of all the sub-datasets were combined to form the RPT for each device. As shown in Fig. 10, RPTs range from 19577.90 us to 19692.26 us, 19865.05 us to 19994.50 us, 19734.17 us to 19831.31 us for device A, B, and C, respectively. There is no overlapped area among the three devices, which means that the RPT of each device is unique. RPTs were used as identifiers of each device and stored in the trusted database with vendor ID, device ID, and ROI on the host machine for future authentication.

In authentication, each time we collected 10000 measurements of profiling time and calculated the average profiling time of all the measurements in the ROI. This process was repeated 20 times for each device. The result shows that all the average profiling times of each device are within their respective RPTs, the success rate of authentication is 100% for all three devices.

VI. Security Analysis

The proposed framework uses profiling time to construct identifiers for PCIe devices. In comparison with other existing mitigation countermeasures, the delay-based authentication model has two major advantages:

- The proposed design does not require any hardware-level or protocol-level modification. Existing countermeasures to DMA attacks, such as [10] and [14], need either an adjustment to the current protocol or physical modification of devices that make designs less practical in the real world. In this work, the authentication is achieved by extracting the time-delay characteristic in device profiling and no additional change on the hardware is needed, therefore it is more feasible as compared to other existing works.
- The cost of the proposed design is low. IOMMU has been proved as an efficient countermeasure to DMA attacks, but activating IOMMU will reduce the real-time performance of computing systems significantly [15]. As a contrast, the delay-based authentication model presented in this work is lightweight. The registration is

a one-time process and there is no impact on performance after the authentication is done.

There are two types of overhead in this design. The first one is the resource overhead which is caused by the storage of the trusted database. In the real world, the PCIe devices are fewer therefore the size of the trusted database is fairly small. For example, if the trusted database has 30 records, the overall overhead of storage is only 1.81 KB.

TABLE I. TIME OVERHEAD (EACH DEVICE)

Time Overhead	Registration Overhead (in minutes)		Authentication Overhead (in minutes)	
	Measuring	*Construction*	*Measuring*	*Calculation*
	100 mins	15.7 mins	3.4 mins	4e-7 mins

Another overhead is the time overhead which is shown in Table I. In this design, there are two kinds of time overhead: registration overhead and authentication overhead. For each device, we collected 300000 measurements in the registration process which took 100 minutes and the process of identifier construction took 15.7 minutes. For authentication, 10000 measurements were collected which took 3.4 minutes and the calculation of average profiling time took 4e-7 minutes. The registration process takes longer, but it is just a one-time process. In comparison, the time overhead of authentication is 3.4 minutes. It is important to note that, the time overhead of identifier construction depends on the performance of the host machine. For both the registration process and authentication process, the CPU used is Core i5-3470.

In this work, we use a Python-based interface for PCIe and the profiling time is measured using the time library of Python. As shown in Fig. 6, the variance of profiling time is very high. Even the proposed design is able to seek out the ROI with the lowest noise and construct unique identifiers for different devices, the lack of high accuracy brings two major issues:

- Overlapping. As shown in Fig. 10, there is no overlapped area. However, if the number of devices increases, there might be some areas of overlap which makes each identifier not unique anymore. Assuming the devices from the same vendor and the same family will remain fewer per PCIe interface, this proposed scheme works.
- High overhead of registration time. The higher the accuracy of measurement, the less quantity of samples for identifier construction is needed. If the accuracy of measurement can be increased, the registration process will need fewer measurements for constructing identifiers therefore the time overhead will be also reduced.

In order to avoid overlapping of RPTs and reduce the time overhead of registration, further research will focus on improving the accuracy of measurement. In addition, applying classification algorithms can be valuable to reduce the number of measurements required in the authentication process.

VII. CONCLUSION

In this paper, we discussed the threats of DMA attacks and demonstrated a successful attack on the victim machine. To mitigate the DMA attacks, we propose a lightweight authentication scheme for DMA-supported PCIe devices based on the unique identifiers constructed with the profiling time. By applying the proposed ROI selection algorithms, the noise in measurements can be further reduced. The results show that there is no overlapped area among RPTs of three PCIe devices and the success rate of authentication is 100%. The proposed design does not require any modifications to hardware and protocol, and does not have any negative effect on the performance of computing systems, make this design more feasible than any other existing countermeasures.

REFERENCES

[1] Direct Memory Attack The Kernel. (2016). Ulf Frisk. Accessd Sep 15, 2020. [Online]. Available: https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Ulf-Frisk-Direct-Memory-Attack-the-Kernel.pdf

[2] Memory Lane - Direct Memory Access Attacks. (2020). Accessd Sep 15, 2020. [Online]. Available: https://eclypsium.com/2020/01/30/direct-memory-access-attacks/

[3] BitLocker Countermeasures (Windows 10) - Microsoft 365 Security. (2019). Accessd Sep 17, 2020. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-countermeasures#pre-boot-authentication

[4] Intel® Virtualization Technology for Directed I/O (VT-d). (2012). Intel. Accessd Sep 17, 2020. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices.html.

[5] B. Morgan, É. Alata, V. Nicomette and M. Kaâniche, "Bypassing IOMMU Protection against I/O Attacks," 2016 Seventh Latin-American Symposium on Dependable Computing (LADC), Cali, 2016, pp. 145-150.

[6] When Lightning Strikes Thrice: Breaking Thunderbolt 3 Security. (2020). Thunderspy. Accessd Sep 20, 2020. [Online]. Available: https://thunderspy.io/

[7] Trusted Platform Module (TPM) Summary. (2018). Trusted Computing Group. Accessed Sep 20, 2020. [Online]. Available: https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary

[8] AMD I/O Virtualization Technology (IOMMU) Specification. (2020). AMD. Accessed Sep 20, 2020. [Online]. Available: https://www.amd.com/en/support/tech-docs/amd-io-virtualization-technology-iommu-specification

[9] D. Wendlandt, D. Andersen and A. Perrig, WendlanB. "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," 2008 USENIX Annual Technical Conference, Boston, MA, 2008.

[10] PCIe® Component Authentication. (2019). PCI-SIG. Accessed Sep 25, 2020. [Online]. Available: https://pcisig.com/pcie%C2%AE-component-authentication

[11] PCILeech. (2020). GitHub. Accessed Sep 25, 2020. [Online]. Available: https://github.com/ufrisk/pcileech

[12] PCI Express and Xilinx Technology. (2020). Xilinx. Accessed Sep 26, 2020. [Online]. Available: https://www.xilinx.com/products/technology/pci-express.html

[13] TL-WN881ND. (2020). TP-LINK. Accessed Sep 29, 2020. [Online]. Available: https://www.tp-link.com/il/home-networking/adapter/tl-wn881nd/

[14] PCIe* Device Security Enhancements Specification. (2018). Intel. Accessed Oct 10, 2020. [Online]. Available: https://www.intel.com/content/www/us/en/io/pci-express/pcie-device-security-enhancements-spec.html

[15] M. Ben-Yehuda, J. Xenidis, M. Ostrowski, K. Rister, A. Bruemmer, and L. V. Doorn. "The Price of Safety: Evaluating IOMMU Performance," 2007 Ottawa Linux Symposium, Ottawa, Ontario, Canada, 2007, Vol. 1.