**Reviewer: 1**

Recommendation: RQ - Review Again After Major Changes

Comments:

The paper presents a novel mechanism, called Aurora, to enable trusted I/O operations on an untrusted machine by leveraging Intel SGX and System Management Mode. This is an important topic as untrusted I/O operations can compromise the integrity and confidentiality of the data.

== Threat model ==

In their threat model, the authors consider that the hardware and device drivers are trusted but the software stack (including the operating system) is untrusted. While reading the paper I had to remind myself on every page that the drivers are trusted.

***Authors' reply:*** *Thank you. In the updated manuscript, we explicitly state that the drivers are not trusted. We do not trust but verify the behavior of the drivers.*

Besides the authors write that their "solution focuses on the password security on the client side". I am not entirely convinced by this threat model: Intel SGX is primarily targeted at cloud-based applications, not personal devices on which users plug-in keyboards and usb drives. I would have liked to read a discussion on how this system will be used in practice and what are the use-cases.

***Authors' reply:*** *Thank you. Intel SGX is now available on personal laptops and NUCs. Personal passwords are very important and should be well protected. We discuss this in section VIII-B "Specific Trusted Paths".*

== Contribution ==

The main assumption of your work is that the hardware and drivers are trusted. The system would rapidly stop being secure if this assumption is broken. I believe this assumption does not hold in practice.

***Authors' reply:*** *Thank you. Only hardware and Aurora software inside SMM/SGX is trusted. Drivers are not.*

The dedicated counting thread and the thread checking for the content of the I/O APIC table are weak hacks, not a valid solution: you cannot assume that these 2 threads, which need 2 cores and will decrease the performance of your system, will do busy-waiting forever.

The contribution would be more substantial and interesting if these assumptions were removed.

***Authors' reply:*** *Thank you. We require the counting thread to be located with the main thread in the same core using Intel Hyper-Threading technology. Such a design contributes to eliminating most of the side-channels like Spectre/Meltdown/RIDL, etc.*

One obvious solution for secure I/O, never considered by the authors, is to use TLS connections from inside enclaves to contact remote peripherals (e.g., printer, storage). The TaLoS library (cited by the authors) can be used to provide such secure connections.

***Authors' reply:*** *Thank you. TaLoS would require another TLS endpoint in the remote server. In Aurora, the SMM acts as the proxy that connects with local devices. Physical attacks on these devices are out of scope of the paper.*

== Case studies ==

In Section VI.B ("Secure OpenSSL Session"), if the trusted time is needed only to obtain a random value then you could instead use the secure random number generator available to Intel SGX (and used by the TaLoS library: <https://github.com/lsds/TaLoS/blob/master/src/talos/patch/arc4random.c.patch#L178).>

***Authors' reply:*** *Thank you. Actually, the TLS server needs to provide a timestamp for the expiration time, not for the random number, which is why we need this value from hardware RTC.*

In the same section, if the server creates a certificate each time it receives a new connection then how does the client attest that this certificate is correct? This check will probably increase the time needed to open a new connection.

***Authors' reply:*** *Thank you. Such a certificate requires a trusted timestamp. A generated certificate can be sealed to the untrusted storage for reuse. We regenerate it repeatedly to measure the efficiency of Aurora’s time services.*

In Section VI.C ("Secure SQLite Database"), why do you eliminate the file-related ocalls? How do you provide similar functionalities? E.g., how do you implement `pread`, `pwrite`, `lseek`?

***Authors' reply:*** *Thank you. We replace the original file-related ocalls with Tlibaurora’s libc. Pread and pwrite are not used by SQLite.*

Why is the use-case based on USB storage and not a more traditional disk storage?

Regarding the experiment at the end of the section, where does the 18% overhead come from? Cryptographic operations or USB access?

***Authors' reply:*** *Thank you. Disk can also be used. However, the SMM will create a race condition with the underlying operating system, which needs more careful design. Therefore we use USB storage as an example and leave disk storage case as future work.*

*At the end of “VI-D Trusted USB Storage Benchmark”, we stated that “The major overheads are contributed by SMM/Protected mode transitions and cryptographic computations on each block”.*

== Evaluation ==

It is not clear to me what the authors want to show in their evaluation.

For example I do not understand the experiment of Table II (page 9). What are you expecting? What do you prove? Do you take into account accesses to trusted time when you report a 11% performance overhead to check whether an enclave requests more than 10k smcalls?

***Authors' reply:*** *Thank you. Yes, our prototype uses a threshold to block an enclave from abusing more than 10K smcalls, as described in the last paragraph of VI-A Framework Overhead.*

Furthermore, according to the first paragraph of Section VIII ("Performance Evaluation") you test your system using virtual machines. Why? How does this affect performance?

***Authors' reply:*** *Thank you. We re-ran the experiments on physical machines and updated the manuscript.*

Finally, the security evaluation (Section VII) does not convince the reader that your system is secure. What about replay attacks, e.g., if the recorded time values are exchanged (see "Attack-awareness" in Section IV.C)?

***Authors' reply:*** *Thank you. Aurora detects replay attacks as expected. Replay attacks are deemed as rollbacks of time values.*

How do the enclave attest each others? How do you ensure that an application enclave is certain that it is communicating with a correct enclave and not a malicious one trying to impersonate Aurora?

***Authors' reply:*** *Thank you. SGX's local attestation is used to achieve it, as stated in II-A "Software Guard eXtension".*

How does Aurora ensure that it is communicating with a valid enclave and not a malicious one trying to impersonate an application?

***Authors' reply:*** *Thank you. Aurora’s Mutual Attestation in section "Secure Session Life-cycle" solves the issue.*

Is it possible to conduct a buffer overflow attack on the buffer used by the SMRAM to cache keyboard input and thus maliciously access sensitive information?

***Authors' reply:*** *Thank you. No. To detect such an overflow, we have ported a hardened allocator named FreeGuard (FreeGuard: A Faster Secure Heap Allocator, CCS ‘17) into SMRAM in Section III-C* "*SMVisor*"*. The buffer will be sent back to enclave and recycled when it’s about to be full.*

== Related work ==

Here are additional works that might be of interest to the authors:

-ROTE (Matetic et al, "ROTE: Rollback Protection for Trusted Execution", Usenix Security '17) use a distributed protocol to provide monotonic counter to enclaves. Even if it probably has a higher latency than the time service described by the authors, there exists at least one successful implementation ("LibSEAL: revealing service integrity violations using trusted execution", Aublin et al., Eurosys '18)

-SGX-FS (Burihabwa et al., "SGX-FS: Hardening a File System in User-Space with Intel SGX", CloudCom '18) proposes a secure file system that leverages Intel SGX. This is more generic than the secure USB storage mechanism proposed by this paper;

-similarly, Pesos (Krahn et al., "Pesos: Policy Enhanced Secure Object Store", Eurosys '18) leverages Intel SGX to propose a secure object store.

***Authors' reply:*** *Thanks for the information. All above references are now added. Besides, we add more recently published related work, such as* "*Fidelius: Protecting User Secrets from Compromised Browsers (IEEE S&P '19)*"*.*

== Other comments ==

The paper could benefit from some reorganization, which would then allow the authors to address the above comments.

-while the title is correct, I do not understand the emphasis on "X86 systems". I would suggest to remove this term and rename the paper, e.g., to "Establishing Trusted I/O Paths on Intel SGX-based Systems";

***Authors' reply:*** *Thank you. We are glad to accept your suggestions, and remove the term "X86 systems".*

-section II.C ("Trusted I/O Paths") does not bring anything to the paper;

***Authors' reply:*** *Thank you. We describe the definition of trusted I/O paths and summaries the methods of building two kinds of trusted paths, with the hope to help readers better comprehend our work.*

-same for the second paragraph of Section V.A ("SMM Supervisor and Drivers");

***Authors' reply:*** *Thank you. We remove the paragraph for simplicity.*

-the ashmd module is never clearly defined;

***Authors' reply:*** *Thank you. It is described in “Kernel Module (ashmd)” in III-E).*

-in Fig. 1 the UEFI drivers box should not be gray as they are trusted.

***Authors' reply:*** *Thank you. They are untrusted but verified by Aurora.*

The paper contains several English mistakes:

-page 3, "forth" -> "fourth";

-page 4, "these drivers are lack" -> remove "are";

-page 6, "as to the hardware clock" -> replace "to" by "for";

-page 7, "encalve" -> "enclave";

-page 9, "with as less noise" -> replace "less" by "little";

-etc.

***Authors' reply:*** *Thank you. We fixed them.*

Additional Questions:

1. Is the topic appropriate for publication in these transactions?: Adequate Match

1. Is the paper technically sound?: Yes

2. How would you rate the technical novelty of the paper?: Somewhat Novel

3. Is the contribution significant?: Incremental

4. Is the coverage of the topic sufficiently comprehensive and balanced?: Treatment somewhat unbalanced, but not seriously so

5. Rate the Bibliography: Satisfactory

1. How would you rate the overall organization of the paper?: Could be improved

2. Are the title and abstract satisfactory?: No

3. Is the length of the paper appropriate? If not, recommend what should be added or eliminated.: Yes

4. Are symbols, terms, and concepts adequately defined?: Yes

5. How do you rate the English usage?: Needs improvement

-----------------------------------------------------------------------------------------------------------------

**Reviewer: 2**

Recommendation: AQ - Publish With Minor, Required Changes

Comments:

\* Paper summary

The paper presents Aurora architecture for trusted I/O paths for enclave applications on Intel platforms with SGX and SMM support. Specifically, the paper outlines design for securing communication with a keyboard, printer clocks, and storage in the presence of an untrusted OS. By default, Intel SGX does not guarantee secure interface to devices, and Aurora bridges this gap for applications where it is essential to have a trusted path to one or more I/O devices. The authors make use of SMM which is protected from the OS to communicate with these devices over a trusted channel.

\* Strengths

- Good observation about the use of SMM for securing I/O communication.

- Well-detailed design and discussion of corner cases for attacks.

- The evaluation is well executed with a good effort to account for many of the design choices made by Aurora.

\* Detailed comments to the authors for minor revision

- Lacks secure display support by design, which makes the appeal of Aurora's standard architecture for secure IO weaker. Network paths are never mentioned in the context of I/O operations.

***Authors' reply:*** *Thank you. Display is not addressed in Aurora because the image frame (video) needs real-time transfer. Network is not considered as one can simply use TLS inside enclaves as both ends to provide secure network communication. We provide protection for storage is that SGX is vulnerable to replay/drop attacks and storage devices cannot detect this, so we use SMM as a trusted proxy.*

- Orthogonal techniques can also bridge the same gap, albeit in bits and pieces, so the novelty of Aurora is incremental.

***Authors' reply:*** *Thank you. Yes, Aurora propose a novel architecture without modifying the system software.*

- The design for keyboard input protection is weak. Blinking caps lock key to notify the user that the login screen is not being forged is not user-friendly. But more glaringly, a random integer from the interval of [1, 5] simply does not have enough entropy. The attacker can also blink the caps lock key for a randomly picked value between [1, 5] and can be correct with very high probability. The authors need to address this with a substantially secure method.

***Authors' reply:*** *Thank you. We then use an improved approach from TrustLogin (AsiaCCS ‘11). At the beginning when the computer is not connected to the network (when the compromised kernel is not possible), we allow the user to use enclave to control SMM Supervisor to program the PC speaker, the user can choose any music he/she wishes. After that, the user connects to the network in the danger of cyber attacks. He/She uses the prior knowledge pre-shared with the SMM Supervisor to know if the trusted path is built.*

- The threat model is unclear and keeps changing while discussing the design in Section 4. For example, the security boundary of the printer I/O ends at the SMM is not really an end-to-end secure channel from the enclave to the end device. Same is the case for the keyboard, where the adversary can physically tamper with the keyboard device to extract keystrokes. It is fine if the design does not protect against these scenarios because it involves an on-premise adversary, but they cannot be excluded from the discussion in the paper.

***Authors' reply:*** *Thank you. Yes, the paper mainly addresses the attacks from compromised software system, where physical attacks on hardware devices are not considered, as given in the threat model section (II-D).*

- In Section III-E (page 5, column 2, last para), strong claims about cache side channel freedom merely because of AES-NI, especially in Intel SGX setting are misleading and self-contradictory to the threat model which rules out all side-channels.

***Authors' reply:*** *Thank you. We complement this in the threat model part: “Even thought we do not consider physical attacks on devices, side-channel attacks, or denial-of-service attacks, we adopt side-channel free cryptographic algorithms and apply an oblivious packet transferring method between hardware enclaves.”*

- Any encryption scheme for integrity such as AES-GCM will have MAC data apart from the encrypted content. Nowhere in the paper did I see the discussion about where this metadata is stored.

***Authors' reply:*** *Thank you. As described in III-A-3), "… the last one is the message authentication code used for integrity protection. All the arguments are marshaled and encrypted." MAC is encoded as part of the packet, and is transparent to the developers.*

- The claim for defense against Iago attacks is not well explained. It is unclear how Aurora can detect mapping attacks without keeping any information about the memory allocation system.

***Authors' reply:*** *Thank you. We adopt the similar method as that in Graphene-sgx Library OS [16] for memory mapping. Aurora uses a shim layer (i.e. Tlibaurora) to verify the result of untrusted calls. We add this in III-E-1).*

- The "Verification Module" discussed in Section III-C does not verify the correctness of the return values, it merely sanitizes the values by checking if they are within the valid range of error codes.

***Authors' reply:*** *Thank you. It is a mistake in our presentation, the module does verify the correctness of the return values and we add this in the updated manuscript.*

- It is unclear if the design choice of disconnecting the secure session is useful in terms of performance, because it highly depends on how frequently the application uses the I/O paths. It is unclear how / when Tlibaurora decides to notify SSV for termination.

***Authors' reply:*** *Thank you. As soon as a trusted app is terminated, Tlibaurora notifies SMM supervisor for session termination. Our evaluation (VIII-A) shows that the performance overhead is acceptable, as Aurora’s trusted services are requested on demand.*

- The evaluation of SQLite only performs insert operations and excludes other DB operations which exhibit a very different file IO behavior. Thus the overhead numbers are not representative of a real workload.

***Authors' reply:*** *Thank you. In the updated manuscript, we added a comprehensive evaluation (TPC-H benchmark) to measure the performance of Aurora-enhanced SQLite in VII-C* "*Secure SQLite Database".*

- The use cases are not well motivated because it is not impossible to deploy them in the absence of Aurora. The OpenSSH and OpenSSL examples can be deployed without support for keyboard-based username password authentication by forcing PKI-based authentication. For cases where the enclave logic may need a username and password, it can receive it over a secure network or via sealed data in the form of files. Further, as the authors themselves mention, the only use of timers in OpenSSH and OpenSSL is for generating randomness which can be addressed by Intel's support for a secure source for randomness for enclaves. For timeouts, the enclave can use secure clocks built on monotonic counters. Thus, the argument that Aurora is vital for execution of these applications is not compelling.

***Authors' reply:*** *Thank you. In those cases which lack of secure network or PKI authentication, or a local computer with compromised system, Aurora provides alternative trusted paths. Yes the sealed file is a good idea, but for the first time a user inputs his password, he needs a trusted path between the keyboard and the target enclave. For highly frequent network I/O, to protect from replay attacks, we need high-precision clock, which current SGX trusted clock and monotonic counters fail to offer.*

*We need a better app.*

- The storage cheat attack example in Section VII raises concern about the threat model as mentioned earlier. After reading the para in Section VII, it seems that Aurora is attempting to detect when the kernel driver drops the I/O requests which is a form of DOS. While it may be advantageous to detect such DOS-like evasion by the kernel driver whenever possible, it is misleading to classify it under the storage attack. Mainly because there are so many DOS- like attacks at the storage interface which Aurora cannot detect.

***Authors' reply:*** *Thank you. We use the example to show that Aurora can be used to check the consistency of storage operations and provide auditable proofs, though there are many DOS- like attacks at the storage interface.*

- The intro and the middle sections paint a picture that there is not a single existing solution for secure IO, which is misleading. Consider mentioning Intel's PAVP and trusted monotonic counters which are well-known solutions in the earlier sections. They can then be dismissed as point-wise solutions as opposed to a generic solution to I/O.

***Authors' reply:*** *Thank you for your good idea. We reorganize them to make the presentation better. We highlight these point-wise solutions in the Introduction section. As you pointed out, some issues are addressed by newly released Intel SGX platform, but they are case by case and not general solutions.*

- Minor fixes

-- Page 1, Column 2, Para 3: incomprehensible opening sentence.

***Authors' reply:*** *Thank you. Done*

-- "time cheat attacks and storage cheat attacks: consider replacing cheat with "deception?

***Authors' reply:*** *Great. Thank you.*

-- Use output instead of "outputted".

***Authors' reply:*** *Thank you. Done*

-- The claim about running 56 enclaves is wrong because (a) the enclave may need more than 1MB of heap+stack memory depending on the code executing inside the enclave (b) the SSV also may need larger heap sizes apart from the code size.

***Authors' reply:*** *Thank you. We remove the wrong claim.*

-- The performance overhead for OpenSSH is not mentioned in the paper.

***Authors' reply:*** *Thank you. We added in the updated manuscript that the login overhead is around 13% for each keystroke.*

-- The figures are referred to on a different page than they appear on.

***Authors' reply:*** *Thank you. Done.*

-- The discussion in Section VII is not Security Evaluation and can be merged without Section VIII as practical effectiveness of Aurora.

-- It is unclear if the ashmd driver is custom written for Aurora or is something that already exists on Linux platforms. Please clarify for ease of reading.

***Authors' reply:*** *Thank you. Sure! It is custom written for Aurora.*

-- Consider changing the para title "Verification Module" to Sanitization / Validation module.

***Authors' reply:*** *Thank you. Done.*

Additional Questions:

1. Is the topic appropriate for publication in these transactions?: Adequate Match

1. Is the paper technically sound?: Yes

2. How would you rate the technical novelty of the paper?: Somewhat Novel

3. Is the contribution significant?: Incremental

4. Is the coverage of the topic sufficiently comprehensive and balanced?: Treatment somewhat unbalanced, but not seriously so

5. Rate the Bibliography: Satisfactory

1. How would you rate the overall organization of the paper?: Could be improved

2. Are the title and abstract satisfactory?: Yes

3. Is the length of the paper appropriate? If not, recommend what should be added or eliminated.: Yes

4. Are symbols, terms, and concepts adequately defined?: Not always

5. How do you rate the English usage?: Needs improvement