Do we need consumer-side enclaved execution?

Kobe Vrancken
kobe.vrancken@cs.kuleuven.be
imec-DistriNet, KU Leuven
Heverlee, Belgium

Frank Piessens
frank.piessens@cs.kuleuven.be
imec-DistriNet, KU Leuven
Heverlee, Belgium

ABSTRACT

The disappearance of Intel SGX on the most recent generation of consumer processors has opened a debate on the value of client-side enclaves. In this short paper we illustrate the usefulness of enclaved execution on consumer devices by highlighting a subset of the unique use cases this technology can enable. Firstly, allowing client-side browser languages to execute inside trusted enclaves can open new innovative possibilities for web applications. Secondly, the ability for consumers to share and donate CPU cycles with integrity and confidentiality guarantees allows for improvements to niche fields such as volunteer computing. Thirdly, the possibility is created for mutually distrusting actors in peer-to-peer networks to bundle computational resources. Furthermore, decentralized peer-to-peer systems can offer energy-efficient alternatives in blockchain consensus protocols. To conclude, we also discuss Digital Rights Management (DRM), one of the more controversial applications of consumer-side enclaved execution.

CCS CONCEPTS

• **Security and privacy → Software security engineering**.

KEYWORDS

Trusted Computing, Intel SGX

ACM Reference Format:
Kobe Vrancken and Frank Piessens. 2022. Do we need consumer-side enclaved execution?. In *Proceedings of the 5th Workshop on System Software for Trusted Execution (SysTEX '22 Workshop).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Recent releases of processor hardware show that Intel has deprecated Intel SGX on consumer processors. Instead, focus has narrowed towards the more popular cloud server use case. This decision makes sense. Applications that focus on cloud computing have been dominating the application-focused part of the trusted computing research space. They have been explored thoroughly throughout previous years. In this short paper, we would like to make a case against dropping support for consumer-side enclaved execution by exploring some of the many use cases that it can offer.

2 CASE STUDIES

2.1 Trusted web pages

Previous research [5][4] has enabled execution of client-side web languages like JavaScript or WebAssembly inside SGX enclaves. This enables web applications to move trusted code to the client-side.

As an example, consider the problem of input validation. In order to enhance user experience, a website should immediately notify a user whenever they enter an invalid value into an input field. Typically, some client-side JavaScript will verify entered values and display appropriate error messages. From a security perspective these client-side checks are, of course, meaningless. User input will always need to be validated a second time at the server-side. Using enclaves, validation can be moved fully to the client-side. The input values are sent to the enclave instead of the server. The enclave validates the value and immediately provides feedback to the client. When all values are entered and correct, the enclave can send them to the server using encryption, preventing any modification of the validated values by the client.

2.2 Domain-specific web applications

The use of trusted client-side web languages can also enable functionality for web applications that would otherwise be impossible to achieve. Take for instance the example of a chess website. A website such as Lichess [3] offers their visitors the ability to evaluate chess positions using a chess engine. These chess engines evaluate the chess board and provide a metric for players that indicates if either the black or the white player has an advantage in the game. The evaluation is often used by players to analyze their own decisions and strategy after completion of a game.

Chess engines are very computationally intensive. The more resources a chess engine has available, the more deeply can it analyze a position. Lichess and other websites offer visitors the ability to evaluate a chess position in the cloud, albeit at a limited depth. If more depth is required, it is also possible to enable local client-side evaluation of the chess position. Since all the work is now off-loaded to the visitor, the visitor can freely decide on the evaluation depth. Consider a chess game played between two players. Commonly, both players will perform a client-side analysis of the game. Both client machines will thus perform exactly the same computations, assuming they choose the same evaluation depth. This is redundant but necessary. Neither client can trust the result of the other client, thus sharing or collaborating on the chess evaluation could lead to manipulated and incorrect results. Now assume that both clients have an SGX-enabled processor and that the chess engine evaluation is executed within a protected enclave. Due to the integrity of computation offered by a trusted enclave there would be no way for a client to manipulate the result of the chess engine. This allows players to share their local evaluations with one another. A protocol could be devised allowing the players to collaborate on an evaluation thereby enabling them to evaluate the positions even deeper. Furthermore, the result of local evaluations could now be published directly on the chess website, allowing any future spectators that explore chess games to use these analyses without needing to spend any computational resources. While the chess engine use-case is

SysTEX '22 Workshop, @ASPLOS, Lausanne, Switzerland
2022.

a very specific example, it highlights that consumer-side enclaved execution can prove to be a valuable resource for domain-specific problems.

2.3 Pay-by-computation

Another use case, explored in [4], is the pay-by-computation model for the web. Currently, web applications often rely on the use of web advertisements as a form of payment for services rendered. Enclaves can offer an interesting alternative. Visitors of websites can donate CPU cycles. This allows websites to distribute workloads among website visitors thereby potentially saving on infrastructure costs. For websites with heavily parallelizable workloads this can open up new possibilities. An architecture where most of the computational load is moved to visitors can provide unlimited scaling opportunities given the workload scales linearly with the amount of users. In short, consumer-side enclaved execution can allow for new scaling possibilities for certain classes of websites and can offer alternatives to the traditional advertisement-focused payment model of the web.

2.4 Volunteer computing

The sharing or donating of CPU cycles is also a key element in *volunteer computing*. Volunteer computing is a branch of computing where a group of volunteers offer CPU time for free to a project. BOINC [1] is the largest and most famous platform that supports volunteer computing. On this platform scientists can register projects that require a large amount of computational resources. Any volunteer with a working computer can download the BOINC client and donate CPU cycles to any of these projects. To ensure correctness of computation some BOINC projects need to send the same workload to multiple volunteers. This ensures that no single volunteer can cheat the system by providing false computation results. Enclaves could eliminate this redundancy. By running the workload inside an attested enclave on the machine of an untrusted volunteer, the integrity of computations can be guaranteed. It becomes unnecessary to send the same workload to multiple volunteers. This elimination of redundancy in volunteer computing by relying on enclaved execution has been explored in [9] and [4].

2.5 Peer-to-peer infrastructure

In previous examples we have explored the donation or selling of CPU cycles from one or more volunteers or website visitors to a single party. There is an even more general case to be considered whereby a group of mutually distrusting peers group their computational resources in a peer-to-peer fashion. To illustrate consider a compute-heavy task that need to execute periodically and that can be parallelized, e.g. compiling or rendering. Whenever a peer needs to execute a compute-heavy task, this work can be distributed evenly among a group of peers. Enclaved execution is used to execute the distributed workload on each peer. This avoids a bottleneck on the local machine and speeds up the workload. Each peer can make use of this computational power provided by the group. In return, during idle moments, each peer executes parts of jobs for other peers. A credit system could even be devised to make sure this happens fairly. In [12] the example of distributing compilation jobs between a group of mutually distrusting peers using enclaved execution was explored in depth.

2.6 Blockchain

In the context of decentralized peer-to-peer systems, blockchains are another prime example. As a more energy efficient alternative to Proof-of-Work, Intel themselves have proposed a consensus method called proof-of-elapsed-time (PoET) [7] used in the hyperledger Sawtooth. PoET relies on Intel SGX enclaves to stochastically elect a leader in a distributed consensus algorithm. To become a viable alternative to Proof-of-Work, consumer-side enclaved execution is thus essential.

2.7 Digital Rights Management

A more controversial application of consumer-side enclaved execution is *Digital Rights Management* (DRM). Using enclaves it is possible to decrypt and load code at runtime, thereby obfuscating the code that is executing. Companies can use this to hide their algorithms to prevent, for instance, reverse engineering. This is used in practice in DRM applications.

As an example, to play Ultra HD Blu-Ray videos today, an SGX enabled processor is required [6]. The removal of SGX support in the latest release of Windows 11 saw consumers unable to play Ultra-HD Blu-Ray video [10]. Another example is WideVine [8], Google's content protection system for premium media used by Google Play, YouTube, Google Fiber, Netflix, Hulu, Amazon, etc. This system offers three protection levels, two of which rely on consumer-side trusted execution environments for protection of the underlying media. In addition to media protection, researchers [2] have also proposed the usage of DRM to prevent cheating in online games.

3 CHALLENGES

In this section we highlight a small subset of challenges that occur for client-side enclaved execution. For one, applications such as trusted client-side form validation require widespread adoption of the technology. Furthermore, there is a great heterogeneity in consumer-side trusted execution environments. The security guarantees provided by Intel SGX are different from those, for instance, provided by ARM TrustZone or AMD SEV. Furthermore, in terms of security, one should consider possible risks to the client. Enclaves could potentially be used as a way to hide malware on a consumer device [11]. Tackling these and other challenges related to client-side enclaved execution will remain important in future research.

4 CONCLUSION

In this short paper we have highlighted several applications of enclaved execution on consumer hardware. This small set of examples already shows that widespread support of enclaves on consumer devices can offer possibilities for innovation in many areas of computing. The technology can be used in innovative ways to enhance existing applications. However, not all applications are without controversy. Applications that use SGX to obfuscate instead of innovate shine a bad light on the consumer-side enclave ecosystem. The question remains whether the positive innovations that consumer-side enclaved execution can offer outweighs the potentially controversial applications of that same technology.

REFERENCES

[1] David P Anderson. 2004. Boinc: A system for public-resource computing and storage. In *Fifth IEEE/ACM international workshop on grid computing*. IEEE, 4–10.

[2] Erick Bauman and Zhiqiang Lin. 2016. A case for protecting computer games with SGX. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*. 1–6.

[3] Thibault Duplessis. 2010. Lichess. https://lichess.org/. Accessed: 2022-01-25.

[4] David Goltzsche, Manuel Nieke, Thomas Knauth, and Rüdiger Kapitza. 2019. Acctee: A webassembly-based two-way sandbox for trusted resource accounting. In *Proceedings of the 20th International Middleware Conference*. 123–135.

[5] David Goltzsche, Colin Wulf, Divya Muthukumaran, Konrad Rieck, Peter Pietzuch, and Rüdiger Kapitza. 2017. Trustjs: Trusted client-side execution of javascript. In *Proceedings of the 10th European Workshop on Systems Security*. 1–6.

[6] Intel. 2017. *High Dynamic Range (HDR) on Intel Graphics*. Technical White Paper. Intel. 13 pages. https://www.intel.com/content/dam/support/us/en/documents/graphics/HDR_Intel_Graphics_TechWhitePaper.pdf

[7] Intel. 2017. PoET 1.0 Specification. https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html. Accessed: 2022-02-17.

[8] Alex Lee. 2018. *WideVine DRM: Getting Started with Devices*. Technical Report. Google. 20 pages. https://web.archive.org/web/20190504082905/https://storage.googleapis.com/wvdocs/Widevine_DRM_Getting_Started_Devices.pdf

[9] Jonathan M McCune, Bryan J Parno, Adrian Perrig, Michael K Reiter, and Hiroshi Isozaki. 2008. Flicker: An execution infrastructure for TCB minimization. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*. 315–328.

[10] Jimmy Pezzone. 2022. Intel SGX deprecation impacts DRM and Ultra HD Blu-ray support. https://www.techspot.com/news/93006-intel-sgx-deprecation-impacts-drm-ultra-hd-blu.html. Accessed: 2022-01-28.

[11] Michael Schwarz, Samuel Weiser, and Daniel Gruss. 2019. Practical enclave malware with Intel SGX. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 177–196.

[12] Kobe Vrancken, Frank Piessens, and Raoul Strackx. 2019. Securely deploying distributed computation systems on peer-to-peer networks. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 328–337.