SGX Project Meeting #2

Agenda for July 7, 2016

What we have done

1. OpenSGX
   (a) Master branch was broken, spent time resolving the issue by reverting commits that introduced the bugs. Ended up studying the extended assembler and their implementation of SGX instructions in QEMU
   (b) Experimented with intra/remote attestation after the aforementioned issues were resolved. Ran the demo successfully
2. NGINX + LibreSSL
   (a) Created a script to build NGINX from sources and link it to libreSSL
   (b) Figured out how to generate self-signed certificates for testing the HTTPS enabled server
   (c) Started sifting through NGINX + LibreSSL source code to identify areas that need to be partitioned
   (d) Found a patch for Open/LibreSSL that separates the private key handling into a separate process (neverbleed). Interface provided to web-facing process is decrypt/encrypt, however, but perhaps the code may be a useful reference
3. Performance Evaluation
   (a) Re-read the Haven paper and read the paper released by the OpenSGX team from GeorgiaTech. Both employ a very similar technique to performance evaluation, they counted the number of cycles that their program consumes without SGX and compared it against the program with SGX instructions. They make an assumption about the number of cycles consumed by an SGX instruction and calculate the overhead based on that

What we want to discuss

1. OpenSGX
   (a) The Linux SDK from Intel has been released, we had a look at it and, while similar to the Windows SDK, it feels under-documented and there are very few samples for us to go by.
2. NGINX + LibreSSL
   (a) Going through the code and identifying manually areas that need to be partitioned is proving to be a difficult task. Having gone through the Wedge paper, we are aware of the possibility of memory-profiling and permissive modes enabling us to make this a more systematic task. We were curious as to what technique would be efficient given our time constraint.

3. Performance Evaluation

    (a) We are fairly confident that we can calculate overhead if we make assumptions, similar to the case of Haven, about the number of cycles that an SGX instruction consumes. However, we are still uncertain about how an end-to-end evaluation would be carried out.

what we are planning to do next week

1. Continue work on identifying the partition within NGINX + LibreSSL that would enable us to guard the private key.

2. Make concrete the code for provisioning the private key. We currently have a sample that is provided with the OpenSGX platform, but we need to tweak slightly to fit our needs