Identifying Unknown Malware Intrusion by Supercomputing

Taosha Wang
Ronny Ko

Limitation of All Security Software

cyber attacks and cyber security are in endless arms race

Software attack → software protection → smarter attacks → smarter protections → …… (never ends)

Smart Cars: even a single security breach can take away the driver's life!

Is there a protection scheme with a guarantee that no software attacks can possibly break…?

Tamper-proof security hardware

Malware Detection & Identification by Tamper-proof Hardware

TPM (Trusted Platform Module): Tamper-proof security chip

TPM is an independent chip from CPU

CPU communicates with TPM to leverage TPM's security services

TPM has its own registers and RSA private key burned at manufacture stage, which never gets revealed outside the TPM

<ins>TPM is mainly used for</ins>:

- Integrity Measurement for launched software binaries
- Remote Attestation

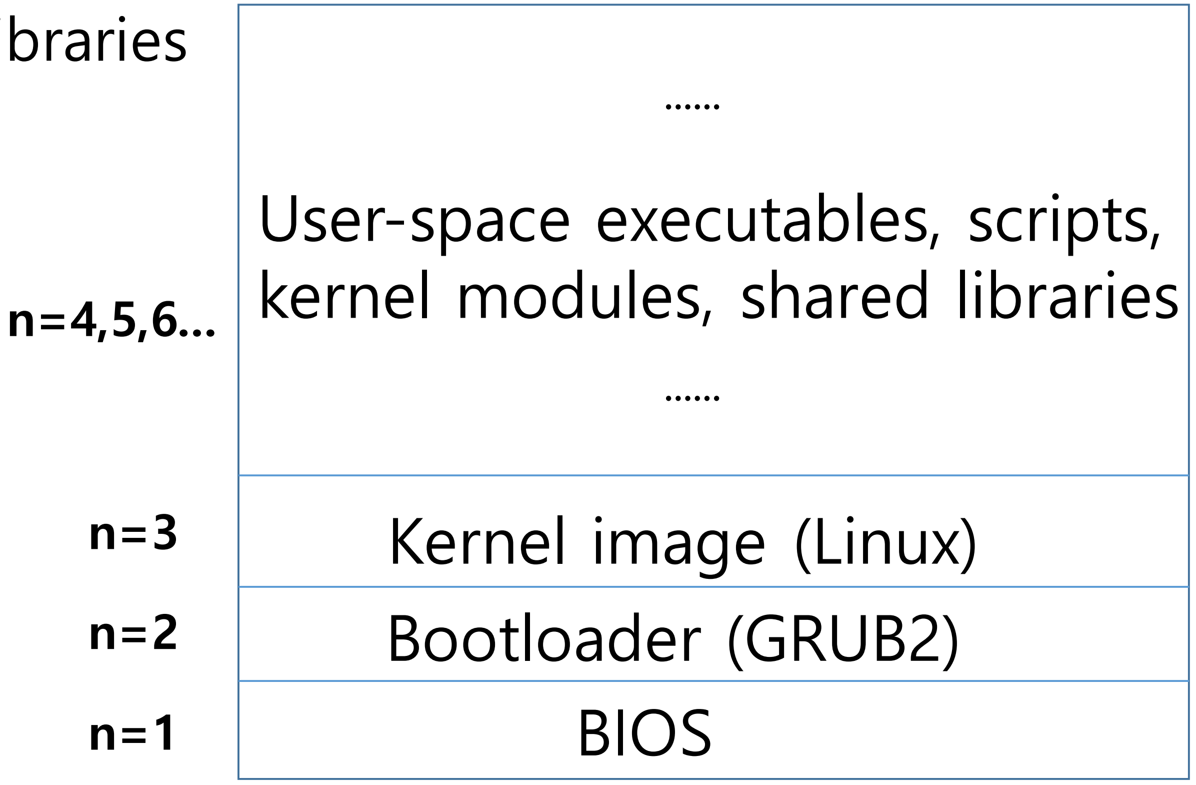
System Integrity Measurement

[System Boot-up Sequence]

BIOS → bootloader (GRUB2)
→ OS kernel image (Linux, iOS, windows)
→ startup executables, scripts, modules, shared libraries
→ additional binaries

[TPM's PCR update rule]

$PCR_{n+1} = SHA1(PCR_n \mid hash_{n+1})$



[System Boot-up Software Stack]

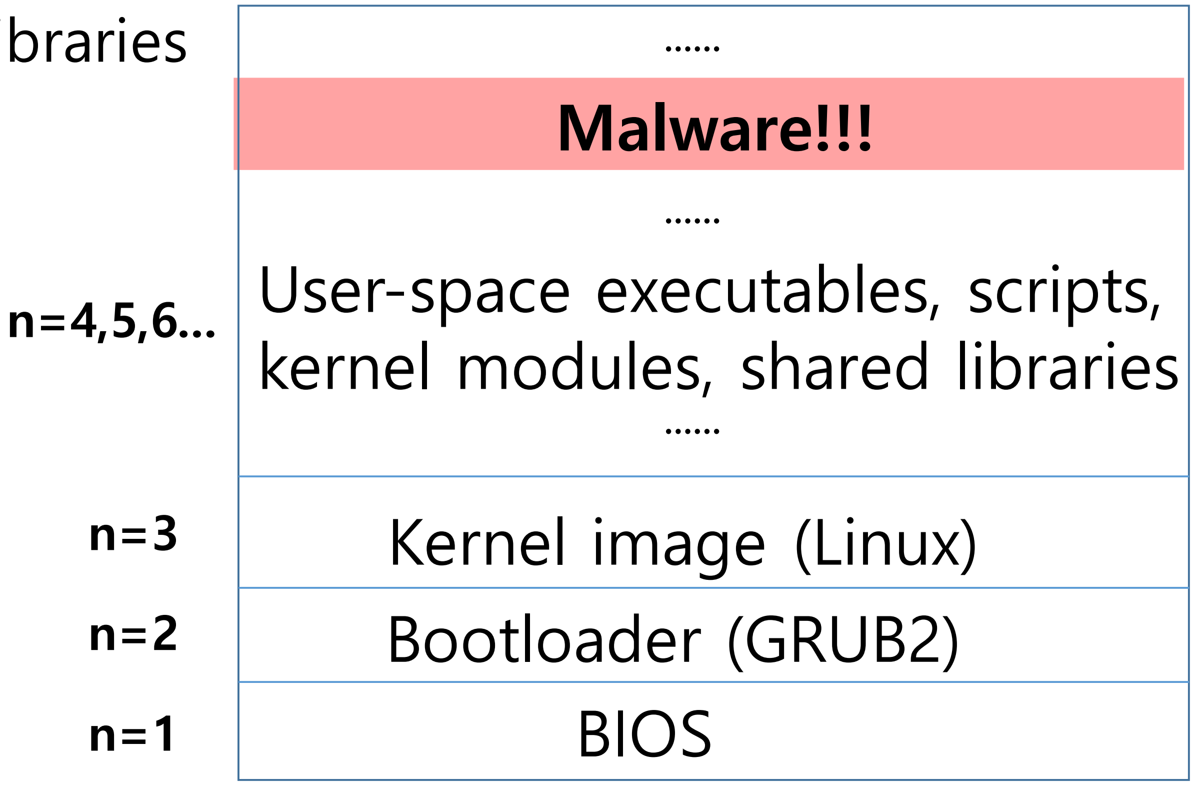
System Integrity Measurement

[System Boot-up Sequence]

BIOS → bootloader (GRUB2)
→ OS kernel image (Linux, iOS, windows)
→ startup executables, scripts, modules, shared libraries
→ additional binaries

[TPM's PCR update rule]

$PCR_{n+1} = SHA1(PCR_n \mid hash_{n+1})$



[System Boot-up Software Stack]

Malware Identification

[Malware Identification Pseudo Code]

```
void Recursive_Malware_Select (int depth)
{

    1. Continue recursion to generate every possible malware combination

    ----------------- CUDA / OpenACC section -----------------------------------
    2. for-loop: insert each malware selected in step 1 into every possible position
                 in the expected boot-up software stack log
    3. Simulate & compute the PCR value.
    4. Check if step 3's result is the same as the TPM-signed PCR value
       (if yes, break the for-loop / if not, go back to step 2)
    ----------------------------------------------------------------------------
}
```

Remote Attestation

<u>Server</u> **<u>Client</u>**

{r: random number} $\xrightarrow{\textbf{(step 1)}}$

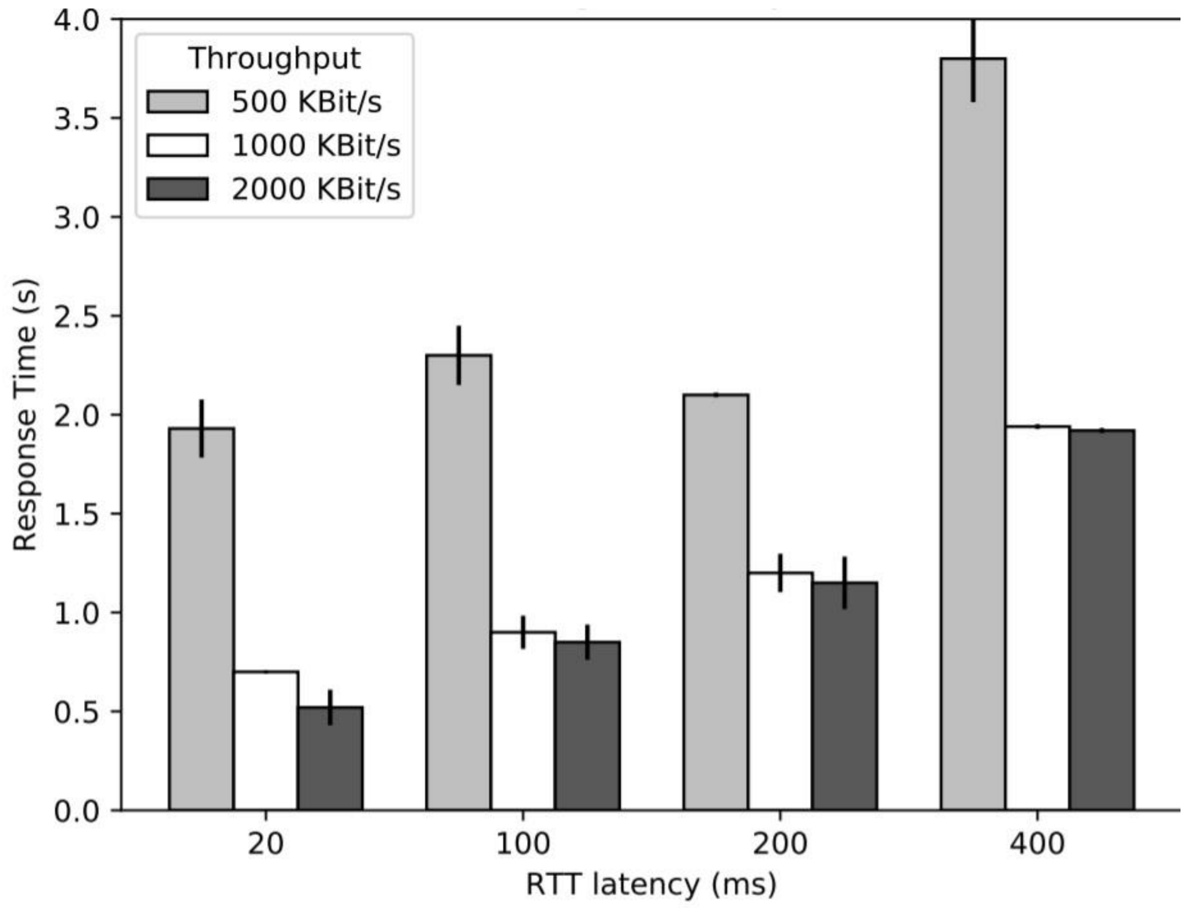
$\xleftarrow{\textbf{(step 2)}}$ {PCR | r}
{Sign$_{RSA}$(PCR | r, privateKey$_{TPM}$) }

(step 3)

Verify$_{RSA}$(PCR | r, publicKey$_{TPM}$)

Experiments

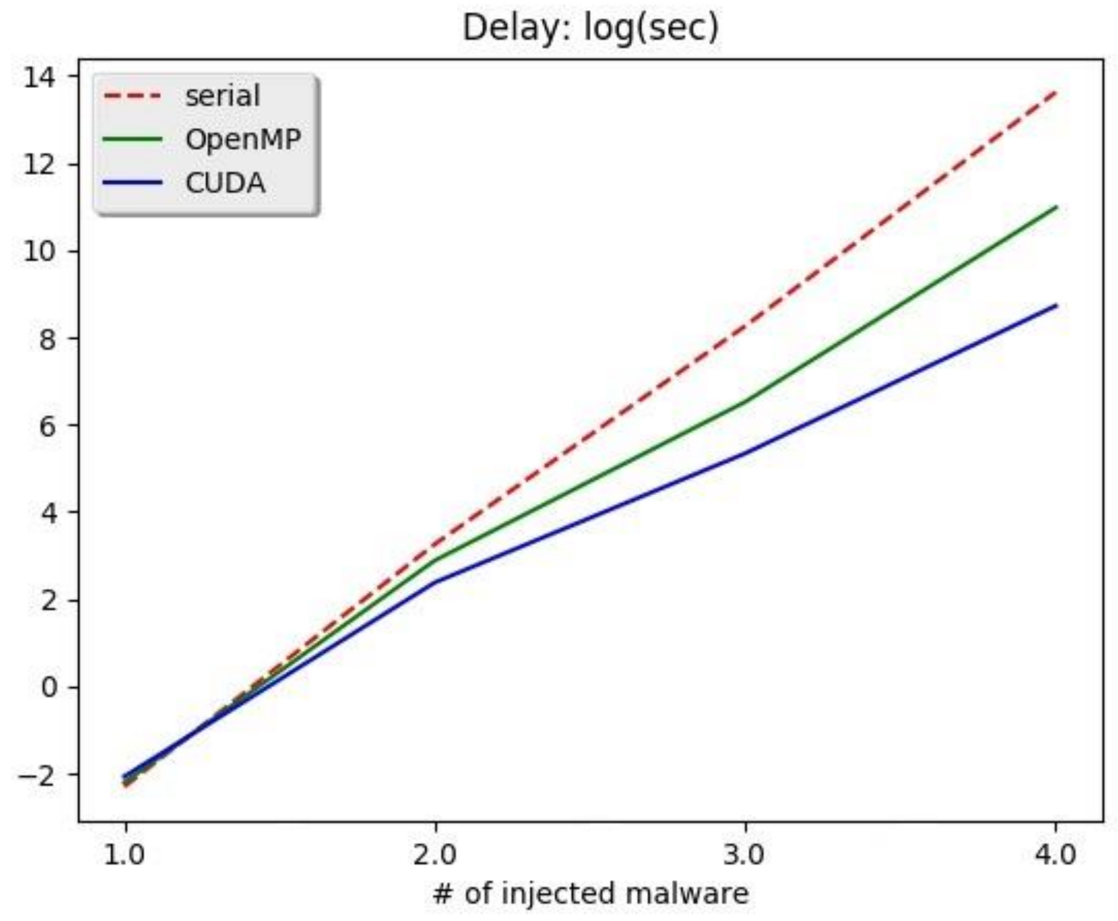
Remote Attestation



- 200 binary files loaded during boot-up

- The attestation delay becomes higher with the increasing network latency, but decreases with the increasing bandwidth.

Malware Identification with Parallel-computing



- Measuring execution time for multiple SHA1 computations (200, 200^2, 200^3, 200^4)

- OpenACC: upto 10x speedup
- CUDA: upto 100x speedup

- Practical enough up to the scenario of 4 malware injection

Conclusion

Leveraging parallel computing enables identifying a reasonable number of concurrently intruding malware even without the launch log file.

Future research:

- How to identify self-modifying malware?
- How to handle multi-core CPU’s non-deterministic binary launch sequence?