LINUX SECURITY SUMMIT NORTH AMERICA

Binding TDISP & Platform Attestation Reports for Confidential VMs

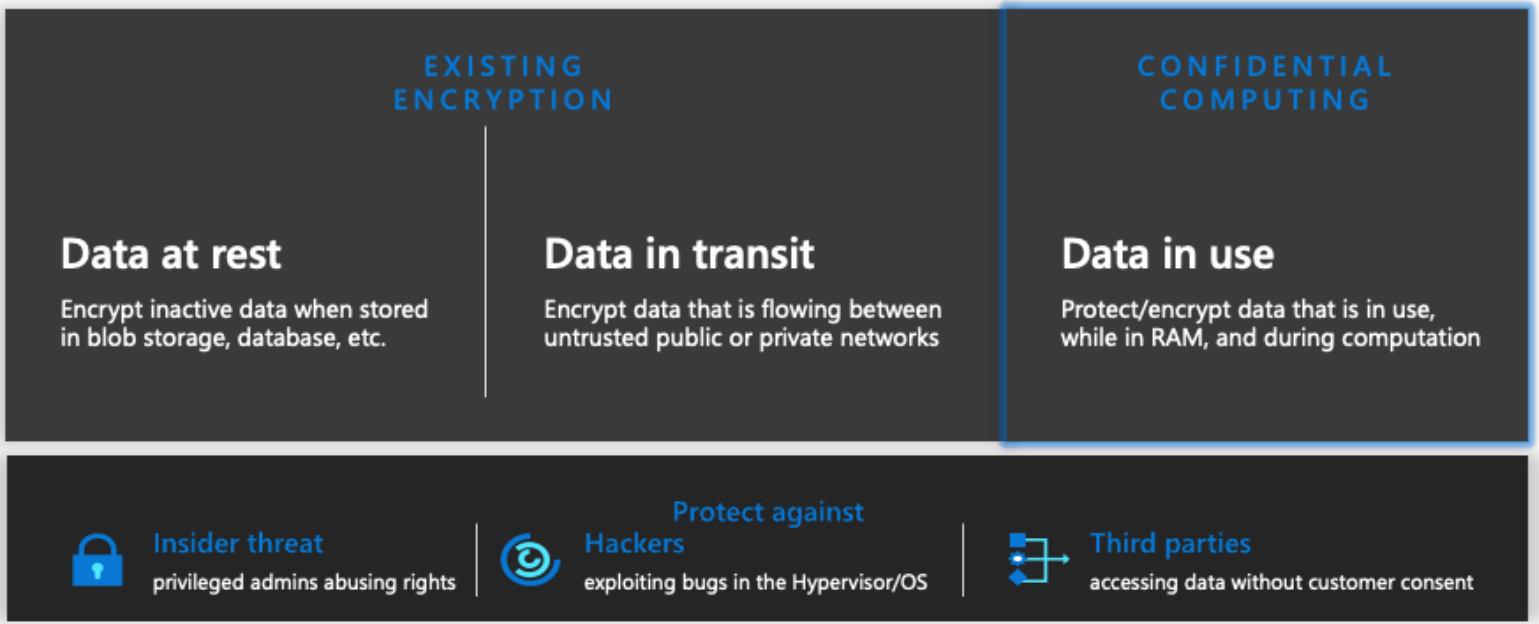
Anna Trikalinou, Microsoft

THE LINUX FOUNDATION

Agenda

- Confidential Computing 101
- TDISP
  - I/O in CC without TDISP
  - I/O in CC with TDISP
- Attestation
  - Platform Attestation Evidence
  - TDISP Device Attestation Evidence
- Binding Evidence
  - Proposal
  - Open questions

Confidential Computing 101



The ***protection of data in use*** by performing computation in a hardware-based, attested Trusted Execution Environment (TEE).

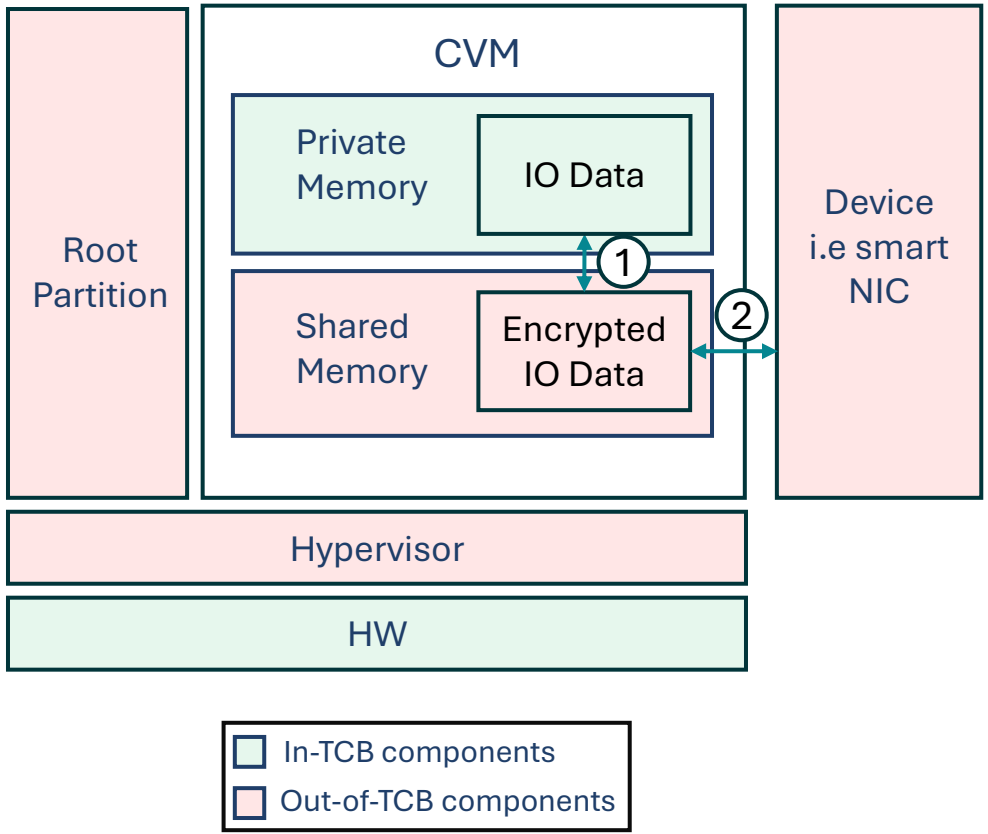
Verifiable assurance for data integrity, code integrity, and data confidentiality

TEE Device Interface Security Protocol (TDISP)

- TDISP is a specification written from PCI SIG
- Defines how to:
  - Establish trust between CVM and a device (device attestation)
  - Secure the interconnect between the host and the device
  - Attach/detach a device interface to/from a CVM
- To support TDISP, we need:
  - Device changes, HW & FW
  - Host CPU changes (1st generation w. TDISP support: Intel GNR, AMD Turin)
  - Hypervisor, Host OS, CVM changes
  - Lots of infrastructure changes

I/O in Confidential VM without TDISP (1/2)

- In CVM arch., memory is divided into:
  - **Private memory**, which is only accessible to in-TCB components (i.e. CVM and trusted HW)
  - **Shared memory**, which can be accessed by out-of-TCB components (i.e. HV, Host, devices)
- Without TDISP, the device cannot be added to the CVM TCB and can only access the CVM's shared memory.
- To do I/O, i.e. smart NIC:
  1. The device driver encrypts(/decrypts) & copies I/O data from private memory to shared (/from shared memory to private)
  2. Device accesses encrypted data via shared memory



I/O in Confidential VM without TDISP (2/2)

- Other types of I/O devices:
  - GPU:
    - GPU needs to perform computations on the plaintext confidential data (i.e. training); cannot process encrypted data
    - CC principle is that only in-TCB components can access plaintext confidential data
    - NVIDIA Hopper offers a Confidential Computing architecture without TDISP to add the GPU into the Platform TCB and allow the GPU to work on confidential data
  - Crypto Accelerator:
    - Needs to process plaintext data; cannot double-encrypt
    - Will also require a custom Confidential Computing architecture to add the accelerator into the Platform TCB

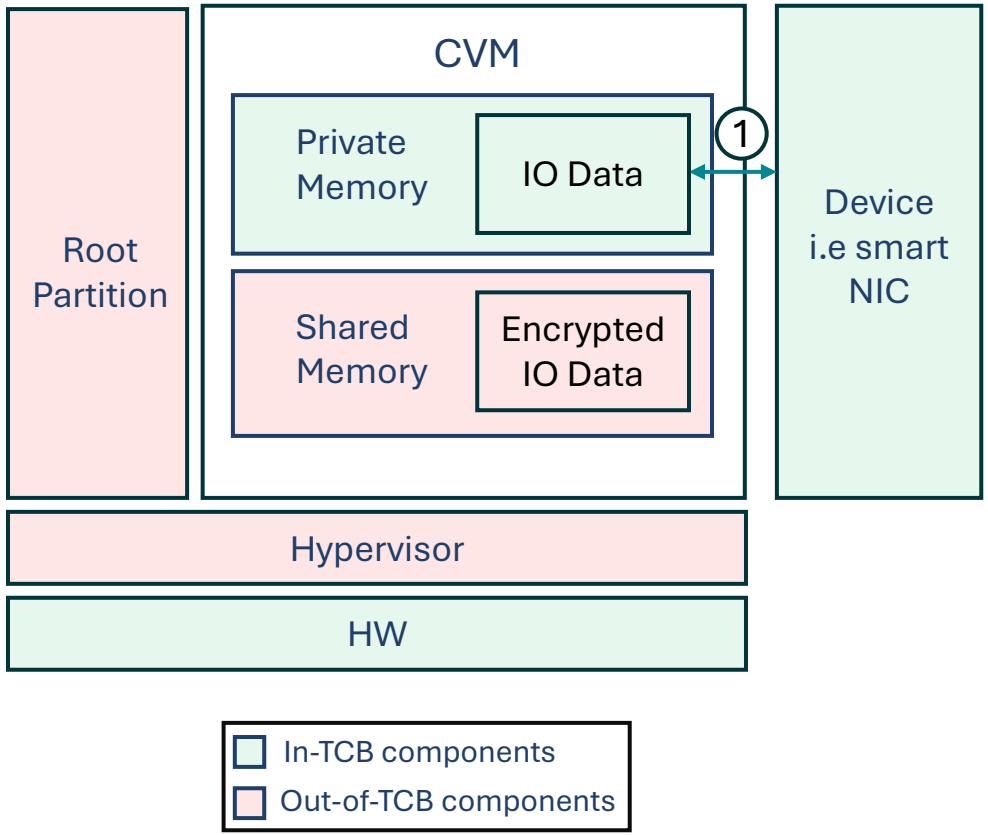
I/O in Confidential VM without TDISP - Impact

- For smart NIC type of devices:
  - CPU encrypts/decrypts all data exiting/entering the TEE
    - Consumes CPU cycles, instead of working on CVM workload
  - Can impact IOPS and reduce performance
  - Can increase cost
- For GPU & Crypto Accelerator type of devices:
  - Need to define a TDISP-like Confidential Computing architecture to add the device into the CVM's TCB, so that the device can process confidential data

<u>Impact</u>: Confidential Computing becomes more complex, less performant and more expensive than General Purpose Computing

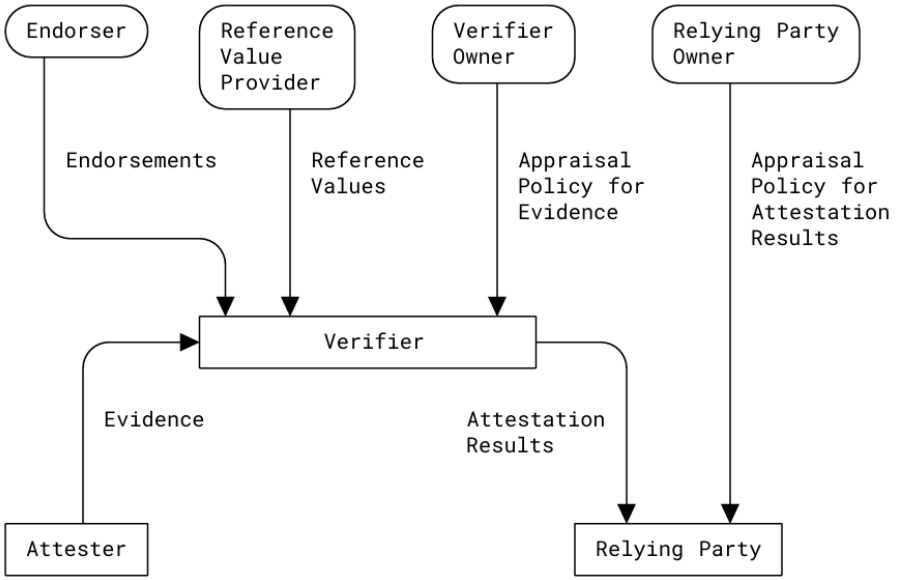
I/O in Confidential VM with TDISP

- With TDISP, the CVM can do device attestation and choose to add the device into the CVM's TCB
- An accepted device can access the CVM's private memory
- To do I/O, i.e. smart NIC:
  1. The accepted device can access the I/O data directly from the CVM's private memory. No CPU encryption/decryption is needed.
- **CVM performance is closer to GP**
  - Impact of IDE and other non-TDISP CVM features, i.e. memory encryption & integrity, IPI.
- **Less cost and complexity to support other accelerators**

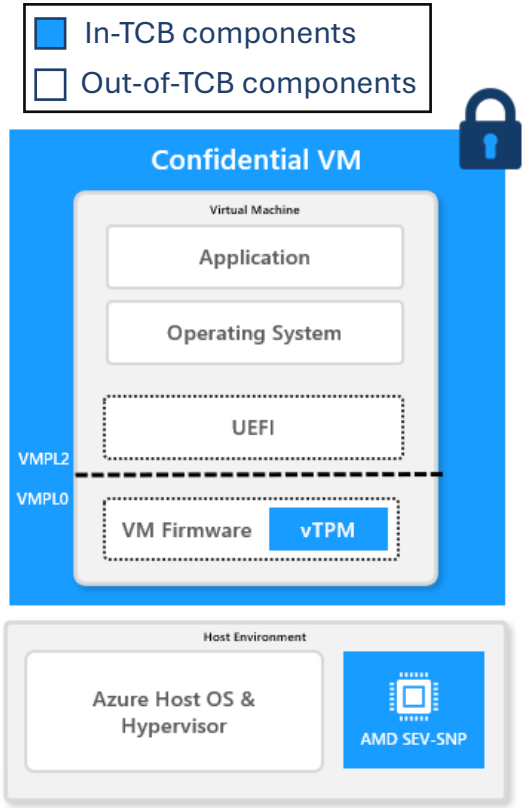


Attestation in Confidential Computing

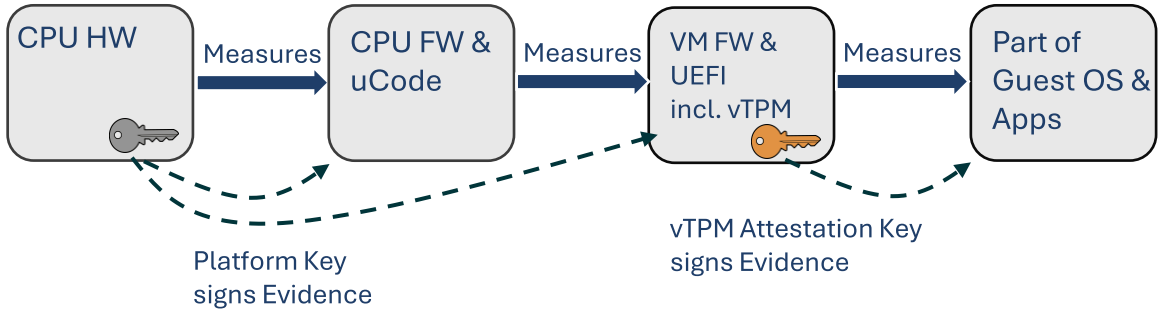
- CC offer verifiable assurance via Attestation
  - “Attestation is the process by which an **Attester** produces **Evidence** (believable, verifiable information about its own state), which is then evaluated by a **Verifier**. The Verifier appraises this Evidence using trusted inputs like **Endorsements** and **Reference Values**, and produces an **Attestation Result**. This result is then used by the **Relying Party** to make a trust decision about the Attester.” according to Remote ATtestation procedureS (RATS) Architecture
  - Platform Attestation and TDISP Attestation are distinct



Platform Attestation Evidence

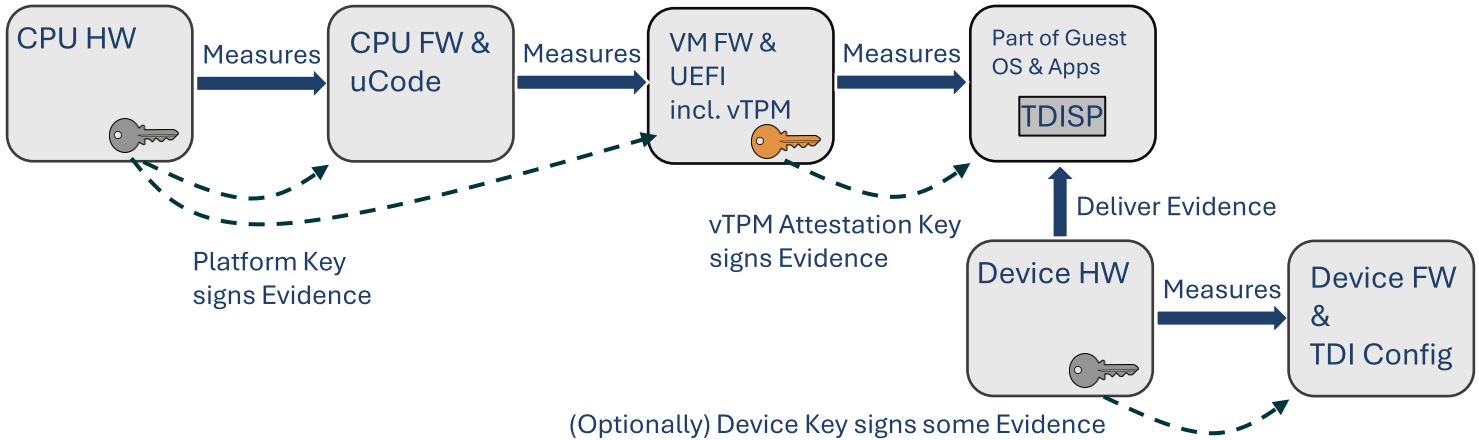


- Platform Attestation Evidence describe the in-TCB components of the CVM
  - Uses a chain of trust where one component measures the next and the root-of-trust is the CPU HW
  - A Verifier can check that the Guest Evidence is bound to the CPU Platform key, which proves that the CVM is running in a Confidential environment.
  - Guest kernel & drivers are measured at boot. Runtime attestation relies on IMA & systemd and their policy.
  - Contains no TDISP device information



TDISP Attestation Evidence (1/2)

- TDISP Device Attestation Evidence describe the in-TCB components of a particular TDISP device
- The TDISP Device Attestation Evidence contain:
  - Device certificates, Measurements & Device Interface Report
- The Evidence is transferred from the Device to the CVM via the CPU FW/HW.
  - Since the CVM (VM FW or Guest OS) collects the Platform and TDISP device evidence, the CVM can verify that both evidence are mapped to the CVM itself.



TDISP Attestation Evidence (2/2)

- **However, a remote Verifier cannot know if the TDISP Device Evidence originated from the same CVM as the Platform Evidence**
  - No existing binding between Platform and TDISP Device Evidence
  - An attacker could replay old TDISP Attestation Evidence with a CVM that doesn't have that TDISP device and cause a Relying Party to disclose secrets to the CVM that wouldn't otherwise
  - i.e. A Relying Party may have a policy to release proprietary training data to a CVM that has an attached TDISP GPU device.

Binding Evidence – Goal

- Goal: Binding method is the same across OSes and CSPs
  - Becomes easier for anyone to implement their own Verifier ☺
- Proposed Solution:
  - CVM collects TDISP Device Attestation Evidence and uses evidence to extend a vTPM NVIndex
  - NVIndex gets signed using vTPM Attestation Key, which is already part of Platform Attestation Report

TPM NVIndex 101

- TPM 2.0 Spec. defines that each TPM/vTPM implementation needs to have at least 6 KB of Non-Volatile Storage
  - Stores TPM-internal data (i.e. hierarchy seeds, counters, clocks, etc.)
  - Stores user-defined data and accessed via an NVIndex
- NVIndex is a handle (aka pointer) to a specific NV memory region. It includes:
  - Size of data
  - Attributes
    - Define static behavior and access rules of the NVIndex.
    - Defined at NVIndex creation time and cannot be changed afterwards.
    - i.e. If NV data acts as a counter, raw data or PCR, which hierarchy can read/write, etc.
  - Access control policy
    - Define dynamic authorization rules on who can access the NVIndex
    - i.e. require a password, key, specific PCR value to access NVIndex

Binding Proposal (1/3)

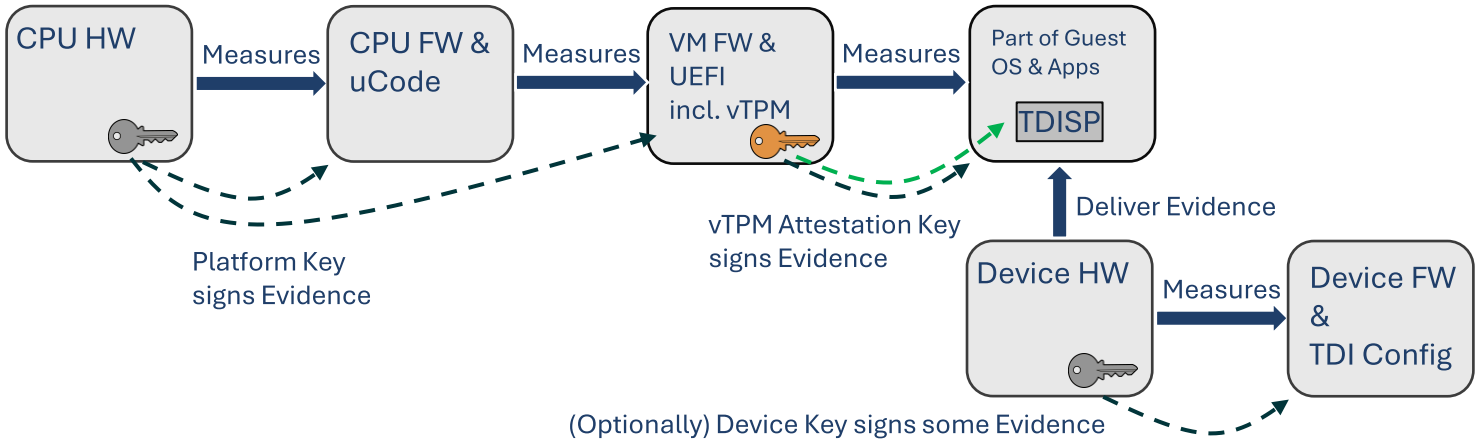
- Define a PCR-like NVIndex that gets extended with the hash of each TDISP Device Attestation Evidence
- Create a file to store Event Log Records related to this NVIndex
- Every time a new TDISP Device Attestation Evidence is retrieved by the CVM, the CVM would do:

  Digest ← Hash(TDISP Device ID, Device Certificate, Measurements, Interface Report)
  NVIndex_New_Value ← Hash(NVIndex_Old_Value | Digest)
  Create a new Event Log Record <NVIndex, extend event, digest>
  Append the new Event Log Record to the NVIndex Event Log file

- Every time the CVM wants to invoke a remote Verifier to perform attestation, the CVM would do:

  NVIndex_Quote ← Sign(NVIndex, vTPM Attestation Key)
  Send to remote Verifier:
  - (existing) Platform Attestation Evidence and all supporting certificates
  - (existing) All TDISP Device Attestation Evidence
  - NVIndex_Quote
  - NVIndex Event Log file

Binding Proposal (2/3)



- By generating the NVIndex Quote using the vTPM Attestation Key we bind the Platform Attestation chain of trust to the TDISP Attestation

Binding Proposal (3/3)

- NVIndex needs to be defined by VM FW using Platform hierarchy
  - Platform hierarchy is under the control of the platform manufacturer (TPM) or VM FW (vTPM)
  - Platform hierarchy is needed to use TPMA_NV_POLICY_DELETE attribute, which restricts everyone else (i.e. Guest) from deleting and redefining the NVIndex and its data
    - This would prevent an attacker with Guest OS kernel privileges from deceiving the remote Verifier into approving a malicious TDISP device
- Other NVIndex attributes to use:
  - TPMA_NV_POLICY_READ, TPMA_NV_OWNERWRITE, TPM_NT = 0x4 (extend-only), TPMA_NV_CLEAR_STCLEAR (reset on reboot)
- Use Canonical Event Log Record format for Event Log format

Open Questions

- Maintain history of all TDISP devices that were evaluated, including rejected TDISP devices and accepted TDISP devices with outdated evidence
  - Each TDISP device evidence is a few KBs in size
  - A Relying Party might use this information in its policy
    - i.e. a CVM that has previously accepted a device with vulnerable FW might have become compromised, even though the device FW got updates to a non-vulnerable version
  - Still, can cause unnecessary bloating for long lived CVMs and attestation requests to the remote Verifier
  - Need some policy over log and NVIndex reset
- Decide on location and permissions of Event Log file and NVIndex extend
  - Isolated to kernel drivers or allow applications?

Thank you!

LINUX SECURITY SUMMIT

NORTH AMERICA