CYBERNETICA

Machine Learning on Confidential VMs

Technical Report

Version 1.0
20.03.2025
D-16-539
Public
© 2015 - 2025 Cybernetica AS

Table of Contents

1. Introduction

This document describes a high level architectural solution for AI-enabled network threat detection. The focus is on the architectural peculiarities which emerge form the use of Trusted Execution Technologies (TEE). The solution is designed to run in the cloud which is an untrusted environment. When a user uploads sensitive data to the cloud they need to trust the cloud service provider (CSP) that the data cannot or will not be accessed by non-authorised parties, including CSP administrators. TEE technology enables the end user to cryptographically enforce access control policies. The context of network threat detection is used to provide a more tangible example for the reader. However, the concepts will likely also apply to workflows of similar form in other domains.

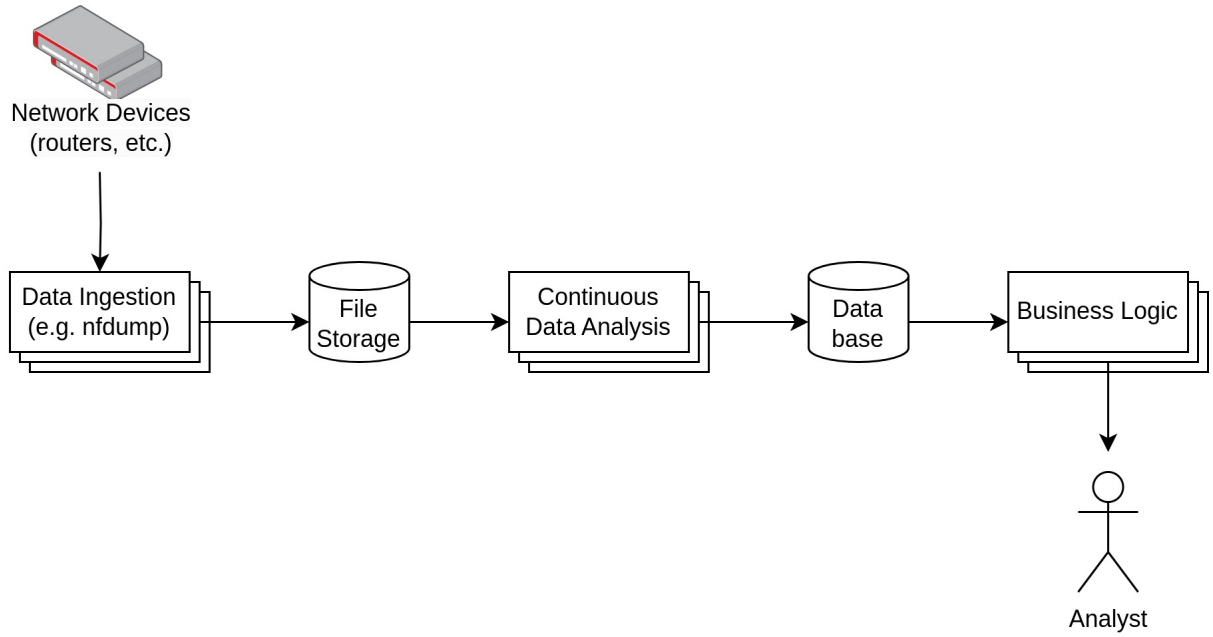


Figure 1. Data flow through the network threat detection solution, starting from network devices and ending at the incident responder.

The threat analysis solution consists of the following components:

Network Devices
: These devices export raw network traffic data, very likely in huge quantities. This data is the basis for the threat detection.

Data Ingestion
: Tools like `nfdump` receive traffic data from the network devices, do some simple preprocessing, and store the data in simple files for later processing.

File Storage
: Detailed traffic data is stored temporarily in files.

Continuous Data Analysis
: These tools analyse all the detailed traffic data and condense it into more valuable

structured data. This could also include a continuous training phase for an artificial intelligence (AI) model, and real-time monitoring of the traffic using the same AI models.

Database

The database persists any information which has required significant computing resources. This includes AI models and an alert history.

Business Logic

Applications which make use of the persisted information. This could be an alert monitoring service which informs the analyst when an anomaly has been detected. The component also comprises a web service which provides usual graphical representations of the network data, which the analyst can use to understand the context and drill into the details of the network traffic.

Analyst

The analyst uses the threat detection solution to quickly react to incoming cyber threats.

The solution is designed to run in a cloud environment, and the solution data (network traffic data, AI models, anomaly history, etc.) has to remain protected from the CSP. The solution data might contain sensitive data such as business secrets or potential vulnerabilities, and necessitating the prevention of any leak.

2. Trusted Execution Environments

Sensitive data at rest and in transit can be protected with commonly available industry-grade encryption. However, in traditional applications sensitive data in use — encryption keys, business secrets — is not encrypted. While the operating system can offer some level of protection, nothing protects the data from the components (hypervisor, OS) or participants (administrator) being malicious or compromised. TEE technology provides an extra layer of protection for sensitive data while it is being processed. The working memory of the software which is protected by TEE technology is encrypted and any tampering will be detected.

One important area for TEE technology is cloud computing where organisations process sensitive data in the cloud. TEE technology protects sensitive data from the cloud operator itself, thus providing a basis for compliance with various data protection regulations. Remote users such as workload owners can use *Remote Attestation* to cryptographically verify that the correct software is running on the cloud servers without physical access to the cloud servers. As a result, a remote user can trust that the remote software runs the expected code in a secure environment. Additionally, *Remote Attestation* provides the user with a means to create a secure communication channel. The user can then upload their sensitive data over the newly-created secure communication channel for further processing.

There are two kinds of TEE technologies:

***VM-based* TEE technologies**
: A VM with all its components is protected from its environment. This means that regular applications can be protected with little or no changes needed. Examples of VM-based TEEs are Intel® TDX and AMD SEV-SNP.

***Process-based* TEE technologies**
: A submodule of an application is protected from its environment. This means that applications need to be divided into trusted and untrusted components. The most important example here is Intel® SGX.

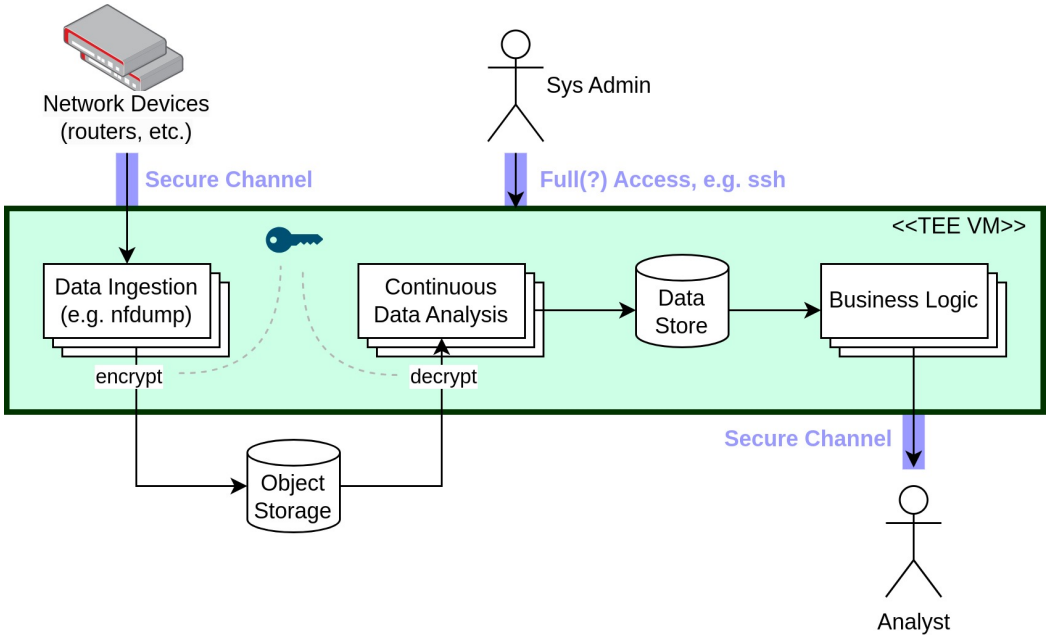
In the remainder of the document we will mostly consider VM-based TEE technologies, as they make the use of existing software much easier than process-based TEE technologies.

3. TEE Architecture

This section constructs the architecture step by step, moving from a simple approach to a more realistic one by improving on shortcomings from the previous one.

3.1. Single Virtual Machine Without Persistence

The solution lives in a single virtual machine which is protected by a VM-based TEE technology (TEE-protected VM, TPVM). Stored data is only accessible as long as the TPVM lives, i.e. any reboot results in full data loss. From the TEE standpoint this is the easiest solution, as there is just a single TEE with a trivial lifecycle.



Secure Communication Channels

Network devices should send data only over an authenticated and encrypted communication channel to ensure that they only communicate with the TPVM. Likewise, analysts should use authenticated and encrypted communication channels to ensure that they see the original threat analysis. TEE technology allows to create such communication channels through the *Remote Attestation* process. The exact technical form of the communication channel itself is flexible. A brief overview of examples is given in Figure 2. One suitable solution for network devices might be the integration of *Remote Attestation* with a VPN. Analysts might also connect via a VPN, or use the Custom PKI approach. In both cases, existing software on network devices or the analyst's device does not need any changes to support *Remote Attestation*.

Encryption of Data

Large amounts of network data will be generated for active networks. It will be better to store this data outside the TPVM's RAM, e.g. in an *Object Storage*. The data needs to be encrypted as it leaves the security perimeter of the TEE. Care must be taken to use suitable encryption schemes to guarantee the integrity when reading the data later[1].

The TPVM should therefore create a root key and further derive unique encryption keys for each encrypted BLOB.
For the sake of simplicity, this scenario assumes that the processed data in the *Data Store* fits into the VM's RAM.

Software Updates

The TPVM in this example should ideally live for an extended amount of time. This means that software needs to be updated live within the VM. This might be possible via automatic software updates. But this probably also requires admin access (not for the CSP administrators) to diagnose problems which emerge due to updates or long uptime.

Pros

The solution is relatively simple to set up. This may result in easier operation.

Cons

When the TPVM terminates, all data and thus all insights are lost.

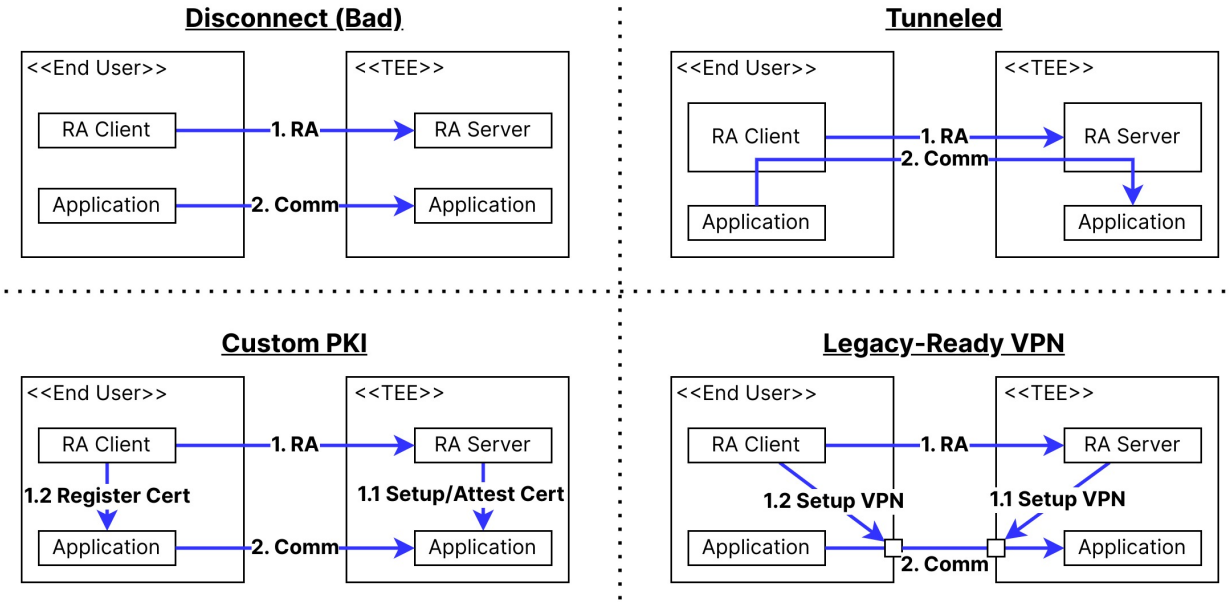
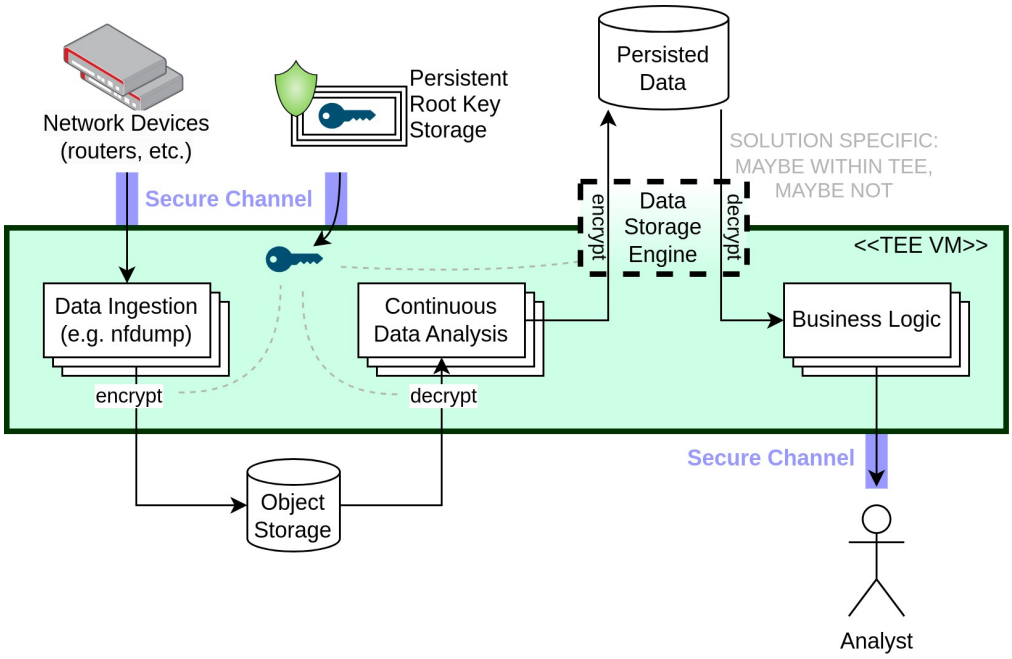


Figure 2. A secure communication channel can be created in various ways. It can even be integrated in applications which do not understand the concept of remote attestation itself. However, one should always ensure that the *Remote Attestation* process is tied to the creation of the secure communication channel, as otherwise the other end of the communication channel might not be the expected protected software.

3.2. Single Virtual Machine With Persistence

The major drawback of the previous approach is the lack of persistence. Without persistence the computations will have to be restarted after e.g. a hardware failure, which can lead to significant downtime.



To achieve data persistence, the TPVM needs to be able to decrypt the encrypted data after a reboot.

Root Key

Intel® SGX, the predecessor technology of Intel® TDX, had a feature to create such a unique key for an enclave, where the CPU itself served as a kind of persistent root key storage. However, with Intel® TDX, this functionality is not accessible. Hence, the root key needs to be stored in a place outside the TPVM, as otherwise it would be erased after the VM terminates. If the solution is used exclusively within the same organisation, then one solution is to keep the root key on-premises. During a VM reboot, the root key can be re-provisioned to the TPVM as a part of *Remote Attestation*. If the solution is used by mutually distrusting organisations, a more complex setup is required, as is mention in the section on multi tenancy.

Database Encryption

Processed data likely has a more structured form than raw input data streams. Hence we assume that a more advanced data storage type is used than for object storage — for example, a RDBMS or a Key-Value store. There are various approaches to how to encrypt data in a data store, which in this case can be meaningfully categorised as follows:

Data Storage Engine inside the TEE

Approaches such as *Transparent Data Encryption* (TDE — handled within the data storage engine) or *Full Disk Encryption* (FDE — handled at the block storage layer) ensure that all data is encrypted, and that the database can see all the data and thus is not restricted in the kinds of queries it can perform. The drawback is that administrators

need to access the TPVM even to perform routine database maintenance tasks.

Data Storage Engine outside the TEE

In this case encryption happens at the application layer, i.e. in the *Continuous Data Analysis* and *Business Logic* boxes. This is called *Client Side Encryption* and means that the data storage engine itself only receives the ciphertext version of sensitive data. Since the data storage engine lives outside the TEE, the administrator has regular access to perform routine database maintenance tasks. This approach can, however, require additional work if the application layer needs to perform queries with joins on or direct comparison of the encrypted sensitive data.

Ephemeral VM

Data and root key are persisted outside the TPVM. A VM crash will only result in the loss of temporary data which can likely be easily recomputed. Ideally, one could use this approach for software update purposes: whenever any component of the solution is updated, a new TPVM is created which then replaces the old TPVM. But the differing characteristics of the solution's components makes this problematic: a critical security update for a *Data Ingestion* component should not cancel a long-running AI training phase from the *Continuous Data Analysis* component. This is addressed in the next section where the various components are moved into dedicated TPVMs.

Pros

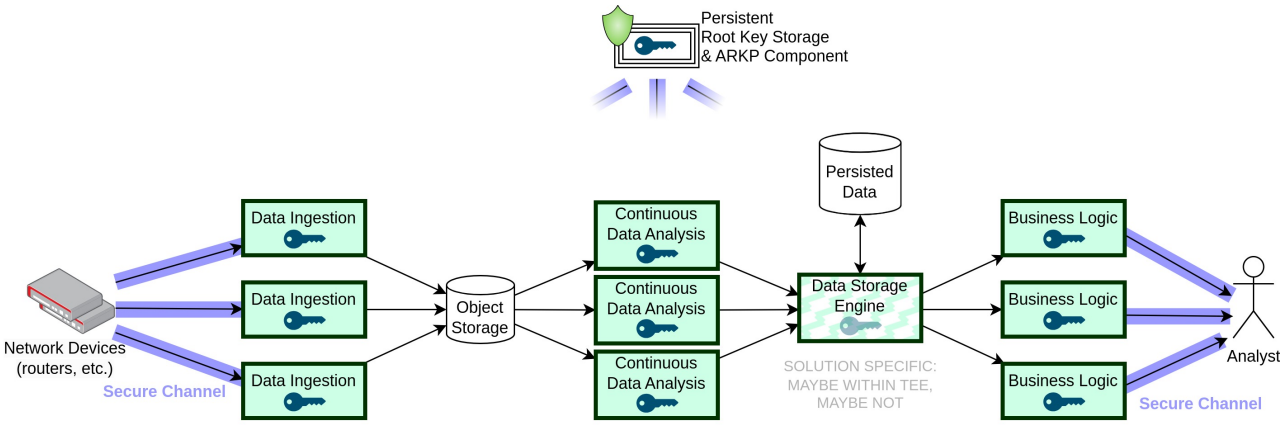
The solution survives reboots and is still rather simple.

Cons

The various software components likely differ in their lifecycle characteristics. If one component requires a VM reboot it also disrupts other software components.

3.3. Multiple Virtual Machines With Persistence

The structure of the solution in this section is more granular: each component of the solution runs within its own TPVM.



Automatic Root Key Provisioning

The root key is now required by more VMs to write and read encrypted persisted data. Whereas in the previous approach a manual root key provisioning method was sufficient, it is highly recommended here to have an automated system (the *ARKP Component*) to link the *Root Key* with the TPVMs. This implies that there needs to be an integrated, automatic process for identifying valid TPVMs during remote attestation. One solution is to compute the respective measurement values of a given TPVM as part of the (trusted) build process and store them in a (trusted) registry. Build system integration allows for high velocity development where changes materialise continuously in newly deployed TPVMs.

VM-to-VM Communication

Data needs to move between TPVMs. There are various ways to achieve this, depending on the needs of a given component.

Communication via Persisted Storage

The TPVMs all share the same root key. They can use the same encryption scheme for a given piece of data to write the data in one VM and read the data in another VM. Since the *ARKP Component* provisions the root key only to trusted VMs, the data cannot be read by rogue components.

Direct Communication

Services may exchange data directly via HTTP APIs or similar TLS-based communication channels. One way to secure this channel is to use a custom PKI where the *ARKP Component* acts as a CA and signs X.509 certificates of the TPVMs during the provisioning phase. Whatever strategy is used, one must make sure that it supports *mutual authentication*, as otherwise the receiving end may accept malicious input which may hide important attacks.

Pros

The fine-grained separation of software components allows for easier horizontal scaling. Availability characteristics are also likely improved over the single-VM approaches.

Cons

The orchestration of the software components is more complex and likely requires additional tooling.

3.4. Tenancy Considerations

The solution may be used to analyse the network traffic of multiple companies to get a broader picture of attack patterns and emerging threats. At this point, the solution is processing sensitive data from multiple parties. This has the following implications:

No administrator access

An administrator having access to a TPVM for maintenance tasks is comparable to a backdoor enabling the exfiltration of any participating company's sensitive data from the solution. Hence the TPVMs need to be able to operate fully autonomously.

The *Persistent Root Key Storage* needs to be trusted by everyone

The complete root key being accessible to a single entity is comparable to a backdoor enabling the decryption of the sensitive data exchanged between the TPVMs. Hence the *Persistent Root Key Storage* needs to be built on concepts such as secret sharing or (software) HSMs where control is split among multiple parties.

The TPVM measurement endorsement system needs to be trusted by everyone

On a related note, it is necessary to ensure that only those TPVMs can be registered as valid in the *ARKP Component* which have been approved by all relevant parties. Otherwise, a single user might be able to create a malicious TPVM which can export the root key after the root key provisioning phase.

If each step in the chain of trust adheres to *Multi Party Authorisation* principles, then the solution is owned by multiple parties in terms of trust. Cybernetica is working on innovative solutions in this domain.

[1] Details can be found in a paper by Briongos et al., or our own Intel® SGX security vulnerability overview, section 2.3.