# SECURITY OVERVIEW

Authentication, Authorization, and Data Protection

UPDATED: December 15, 2022

hyperview

590-1122 Mainland Street
Vancouver, BC. V6B 5L1

# SECURITY OVERVIEW

## CONTENTS

# Overview

This document describes authentication, authorization, and data protection techniques in the Hyperview cloud-based platform. Topics covered are:

- Trusted Platform Foundation
- Compliance
- Types of Data
- Customer Data Isolation
- Authentication
- Authorization
- Stored Credentials Security
- Data at Rest Security
- Data in Transit Security
- Data Collector Security
- Model and Device Definitions

# Trusted Platform Foundation

Hyperview is deployed on top of Microsoft Azure. It leverages the Azure application security services, continuous vulnerability scanning, and the work completed by Microsoft to certify their data centers for PCI, SOC and other certifications.

# Compliance

Hyperview is SOC 2 Type II compliant in accordance with American Institute of Certified Public Accountants (AICPA) standards for SOC for Service Organizations also known as SSAE 18. We perform annual audits with an accredited third-party auditor. Achieving this standard with an unqualified opinion serves as third-party industry validation that Hyperview provides enterprise-level security for customer's data secured in the Hyperview system. Compliance reports are available to customers on request.

# Types of Data

There are four types of data stored in the platform:

**User identification data.** Hyperview authenticates users based only on email, first and last name, password, and (optionally) phone number. As this data is owned by you, the customer, we delete all this data when you terminate your relationship with Hyperview.

**Customer application data.** This data is collected automatically by a Hyperview data collector or generated while using the application. Customer application data includes asset properties, data center floor layouts, sensor data, application usage logs, and asset change logs. Like identification data, you own this data as the customer, so it is deleted when you terminate your relationship with Hyperview.

**Platform health data.** This Hyperview-generated data is designed to improve the operations of the Hyperview application. Platform health data includes latency data and application errors. Hyperview owns this data, which it uses to improve the performance, stability, and usability of the platform.

**Enrichment data.** Hyperview provides this data to extend a user's data. Examples of enrichment data include device properties and images coming from the Hyperview Catalog service. Hyperview owns enrichment data, which is used to enrich and improve user-provided and collected data.

# Customer Data Isolation

Customer application data is stored in two databases.

**Customer Data Store**
Customer data is kept in a dedicated database that is isolated and dedicated for the customer instance. This database keeps all Asset Information, Users, Layout, credentials, etc.

**Time Series Data Store**
This database keeps anonymized numeric sensor readings over time. This database is isolated and dedicated for the customer instance. For example, a reading would be a timestamp, sensor identifier (UUID), Sensor Type (e.g., temperature) reading (e.g., 70).

# Bandwidth and Resource Utilization

Hyperview is designed in a manner that enables users to control sensor monitoring frequency, discovery frequency, and network targets. In addition, the software architecture is designed to efficiently use resources and is a core tenant of Hyperview's software development cycle.

While it is hard to estimate the bandwidth utilization without understanding the specifics of the deployment–such as number of assets, sensors, current network configuration, enabled features, and many other parameters–here are a few guidelines.

**Discovery**
Discovery is usually performed outside of business hours (i.e., daily after hours, once a week on weekends) for the main purpose of detecting changes and additions to the infrastructure environment. Customers have a lot of control on this process and can design it to meet their requirements.

**Server Monitoring**
If IPMI/BMC monitoring is all that is required, then the operating system (OS) is not touched or interfered with. The telemetry and discovery are completed on the management port using SNMP or IPMI.

If operating level information such as CPU, disk, and RAM utilization is also required, then SSH or WMI can be enabled and monitored. To enable or not enable this is entirely under the control of the user. This process is designed to have minimal impact on network and CPU utilization, with telemetry only using a few bytes per sensor.

**Power and Environmental Monitoring**
Power and environmental monitoring is out-of-band and targets specific sensors. Utilization impact is minimal per device and is limited to a few bytes per sensor.

**AssetTracker RFID**
The AssetTracker RFID module (Gen2) is mostly event-driven with a heartbeat sent every 60 seconds. If optional environmental sensors are deployed, they will add a few bytes per sensor approximately every 10 minutes, by default. The bandwidth impact is minimal and is out-of-band.

**Alarm and Trap Aggregation**
Alarm and trap aggregation is event-driven, and is dependent on how chatty the devices are, as well as the infrastructure environment. It is under the user's control as to use or not use alarm and trap aggregation, and when to deploy it.

**Firmware Update**
Firmware updates apply to compatible devices and is designed to be light on bandwidth, with various layers of caching and validation to ensure efficient use of network resources while maintaining security. Firmware update jobs are designed to target specific devices at a specific time, which all under the user's control.

**Data Collector Deployment**
It is recommended that users deploy the data collectors on the monitoring networks and to not intermix monitoring, management, and production traffic.

# Authentication

Hyperview customers can create their own dedicated user accounts in the platform or use an enterprise single sign-on with Azure AD/Microsoft 365.

# Authorization

Hyperview has the following user roles:

- **Administrator**
- **Data Center Manager**
- **Power User**
- **Reporting User**
- **Read Only User**

User roles define what users can do in the application. In addition, an Administrator can define access controls on specific assets to limit what other users can see and interact with. For instance, a group of users can be restricted from viewing a certain location.

# Stored Credentials Security

Hyperview users can add access credentials for managed devices. For example, Hyperview will use a configured SNMP community string to access sensors like current utilization from a managed Power Distribution Unit (PDU). Device credentials are stored in the Hyperview database and encrypted using the Advanced Encryption Standard (AES). This data is always stored and transported encrypted even over HTTPS.

For data collection, it is recommended to use unprivileged or read-only user levels for device credentials.

# Data at Rest Security

The application is installed in Microsoft Azure and uses the at-rest data protection capabilities of Azure Cloud.

The Data Collector is installed by the customer on a physical or virtual machine. If at-rest data protection is required for Data Collectors, technologies like BitLocker or dm-crypt can be employed.

# Data in Transit Security

All interactions with the platform from a user (using a web browser) or from a Data Collector (as it communicated with the APIs) is secured with HTTPS.

# Data Collector Security

The Hyperview Data Collector can be installed on one or more Windows or Linux physical or virtual servers. It collects and relays data back to the Hyperview platform, which lets you monitor and perform operations against the collected data.

The Data Collector must be registered with the platform before it can relay information. Registration is a special handshake that creates a unique identifier to access the platform. The process can only be triggered from the machine that hosts the Data Collector and requires a unique key that is generated by the platform for that Data Collector.

Once registered, the Data Collector secures the key in a configuration file that is stored locally in a secure manner. It will then poll the platform for data collection jobs. Note that all communication between the Hyperview platform and the Data Collector must be initiated by the Data Collector.

# COLLECTED DATA

Three types of data are collected by Hyperview: Data needed to identify assets (e.g. make, model and serial number), data to monitor the utilization of assets (e.g. power draw or CPU utilization); and data to graph the relationship of an asset in relation to other assets (e.g. network connectivity and power connectivity).

With that in mind data collection follows these design tenants to ensure security and privacy:

The customer is in control of what protocols to enable.

The customer is in control of the privilege level of the user used to collect data. Data collection will function with read only access levels.

The customer is in control of what networks and IP ranges to discover and monitor.

The customer is in control on where to place a data collector in the network and which networks it will have access to.

Only data needed to perform the above three tasks is collected.

Below is a list of data types that is discovered and monitored by protocol and, where applicable, by device type. Please note, the list below may change based on the capabilities of the target systems, vendors, firmware versions, allowed access, and protocol versions.

While every effort has been made to ensure the list is comprehensive, Hyperview is a continuously developed product that operates in a multi-vendor environment where networked assets are constantly changing and where vendors extend standard protocols or implement them in a non-standard way.

As a result, there may be some variation in what data is collected and monitored between vendors and generations of hardware/firmware.

We encourage users to review the Hyperview Model and Device Definitions section for more information on the topic.

**BMC-based Protocols: IPMI, iLo and iDRAC**

- Firmware version
- Make and model of the device
- Motherboard serial number
- Device serial number
- Temperature Sensors
- Voltage Sensors
- Current Sensors
- Fan Sensors
- Chassis Physical Security Sensor
- Processor type, make and model
- Power Supply Sensors
- Power state Sensor
- Memory type, make and model
- Drive Slot configuration
- BMC LAN information
- Management Subsystem Health
- Motherboard Battery
- Field Replaceable unit state

**Cisco UCS Manager**

- Inventory (e.g. Servers, blades and network assets)
- Inventory identification (e.g. Managed Models, Serial numbers and blade location)
- Inventory configuration (e.g. Storage, CPU and memory)
- Sensors (e.g. Power and temperature)

**SSH for Linux/Unix**

- Installed Software list
- CPU information
- CPU utilization
- RAM information
- RAM utilization
- Network hardware information (e.g IP and MAC)
- Disk capacity sensor
- Storage hardware information
-

**Windows WMI**

- Installed Software list (apt/rpm tool)
- CPU information
- CPU utilization
- RAM information
- RAM utilization
- Network hardware information (e.g IP and MAC)
- Disk capacity sensor
- Storage hardware information
- Hyper-V virtual machine list

**SNMP**

SNMP is a definitions-based protocol and supports a large variety of devices and manufacturers. Below is a list of the general kinds of data and sensors discovered and monitored by Hyperview by device type. Please note that the intention is to give the reader a good idea on what is monitored. The specific list of sensors read on an asset depends on the manufacturer, asset type, the capabilities embedded in the SNMP implementation and the device definition provided by Hyperview.

**Network Switch**

- Switch port configuration
- MAC Addresses per port
- Identification and inventory information
- VLAN information

**Busway**

- Frequency
- Phase Current
- Phase Power
- Phase VA
- Phase Voltage
- Phase Voltage Total
- Total Current
- Total Load
- Total Power
- Total VA
- Total VAR

**Chiller**

- Cooling Demand
- Cooling Load
- Cooling Output
- Humidification Active
- Dehumidification Active
- Sensor Failure
- Power
- Return Temperature

**CRAC**

- Air Flow
- Cooling Demand
- Cooling Load
- Cooling Output
- Dehumidification Active
- Dew Point
- Enter Water Temperature
- Fan Failure Boolean
- Fan Speed
- Humidification Active
- Power
- Return Humidity
- Return Temperature
- Supply Humidity
- Supply Temperature

**CRAH**

- Air Flow
- Cooling Demand
- Cooling Load
- Cooling Output
- Dehumidification Active
- Fan Speed
- Humidification Active
- Power
- Return Humidity
- Return Temperature
- Supply Humidity
- Supply Temperature

**Environmental Monitor**

- Air Flow
- Ambient Temperature
- Analog Input
- Camera Motion
- Dew Point
- Door Lock Status
- Door Status
- Door Switch
- Dry Contact
- Humidity
- Leak Alarm Zone
- Leak Cable Current
- Leak Unit Distance
- Leak Unit Status
- Leak Zone Distance
- Leak Zone Status
- Light Level
- Power
- Pressure
- Smoke Alarm
- Sound Level

**Generator**

- Battery Current
- Battery Voltage
- Frequency
- Fuel Level
- Oil Pressure
- Phase Current
- Phase Power

- Phase VA
- Phase Voltage
- Plant Battery Voltage
- Running
- Total Current
- Total Power
- Total VA
- Total VAR

**In Row Cooling**

- Airflow
- Cooling Demand
- Cooling Load
- Cooling Output
- Dehumidification Active
- Fan Speed
- Humidification Active
- Power
- Return Humidity
- Return Temperature
- Supply Humidity
- Supply Temperature

**PDU**

- Breaker Current
- Emergency Power
- Frequency
- Panel Current
- Panel Phase Current
- Panel Power
- Panel VA
- Phase Current
- Phase Load
- Phase Power
- Phase Rotation Loss
- Phase VA
- Phase Voltage
- Power Factor
- Total Current
- Total Load
- Total Power
- Total VA
- Total VAR

**Power Meter**

- Frequency
- Phase Current
- Phase Power
- Phase VA
- Phase Voltage
- Total Current
- Total Power
- Total VA
- Total VAR

**Rack or Tower UPS**

- Battery Capacity
- Battery Current
- Battery Discharging
- Battery Voltage
- Bypass Frequency
- Output Power
- Output Source
- Output Total Load
- Output Total Power
- Output Total VA
- Output VA
- Phase Bypass Current
- Phase Bypass Power
- Phase Bypass Voltage
- Phase Input Current
- Phase Input Frequency
- Phase Input Power
- Phase Input Voltage
- Phase Output Current
- Phase Output Load
- Phase Output Voltage
- Power Factor
- Runtime Remaining
- Usable Power

**Rack PDU**

- Current
- Outlet Power
- Outlet Status
- Power Factor
- Total Current
- Total Load
- Total Power
- Total VA
- Usable Power

**Transfer Switch**

- Selected Source
- Source 1 Circuit Breaker 1 Open
- Source 1 Failure Status
- Source 1 Frequency
- Source 1 Overload
- Source 1 Over Voltage
- Source 1 Phase Current
- Source 1 Phase Voltage
- Source 1 Under Voltage
- Source 2 Circuit Breaker 1 Open
- Source 2 Failure Status
- Source 2 Frequency
- Source 2 Overload
- Source 2 Over Voltage
- Source 2 Phase Current
- Source 2 Phase Voltage
- Source 2 Under Voltage
- Total Load
- Total Power
- Total VA

**UPS**

- Battery Capacity
- Battery Current
- Battery Discharging
- Battery Status
- Battery Voltage
- Bypass Frequency
- Frequency
- Output Current
- Output Source
- Output Total Load
- Output Total Power
- Output Total VA
- Output Voltage

- Phase Bypass Current
- Phase Bypass Power
- Phase Bypass Voltage
- Phase Input Current
- Phase Input Frequency
- Phase Input Power
- Phase Input Voltage
- Phase Output Current
- Phase Output Load
- Phase Output Power
- Phase Output VA
- Phase Output Voltage
- Power Factor
- Runtime Remaining
- Total VAR

**Utility Breaker**

- Breaker Position
- Frequency
- Phase Current
- Phase Power
- Phase VA
- Phase Voltage
- Total Current
- Total Power
- Total VA
- Total VAR

## MODBUS AND BACNET

Modbus and BACnet protocols are monitoring only. Because of the nature of these two protocols Hyperview includes a constructor that allows customers to define exactly what they want to monitor. For more information, visit https://docs.hyperviewhq.com/.

# Model and Device Definitions

A device can be a server, CRAC, PDU, UPS, or any IP accessible device monitored by Hyperview. To correctly model the device, Hyperview needs to understand the device type, manufacturer, and model. This is then mapped to applicable attributes, sensor data and communication protocols. To arrive at its **Model and Device Definitions**, Hyperview uses one or more of the following knowledge sources:

Model Data

Device Definitions

Traps

## MODEL DATA

**Model Data** allows Hyperview to identify what a device is, down to the manufacturer and model. Apart from identification information, it allows Hyperview to classify a device based on extended attributes, such as space utilization, expected power utilization, network-related information, and any other data that may be relevant (depending on the device type).

The Hyperview model database contains attributes and images of thousands of devices.

If Hyperview fails to identify a device based on model data, it attempts to perform generic detection to identify what the device is. If that fails, the device is added and marked as “Unknown” in the system.

## DEVICE DEFINITION

The **Device Definition** subsystem allows Hyperview to identify a device and react dynamically to the different variations of it. It will also enumerate and pull data from the various sensors available. Sensor data is then mapped to instrumentation points in the Hyperview data model.

Within Hyperview, there are two high level protocol types:

Standards-based Protocols

Definition-based Protocols

Standards-based protocols (e.g., WMI, IPMI and VMWare) are typically from a single vendor and can be described with a standard algorithm with little to no variation across devices.

Definition-based protocols (e.g., SNMP and SSH) can have many variations between devices, with the only common element being the protocol or language to query the devices. SNMP, for example, can be implemented on power, network, and facility devices. Each one of these devices may have its own level of support for the protocol and its own unique sensor map.

Within definition-based protocols, there are two subcategories:

Discoverable Protocols

Non-discoverable Protocols

Discoverable definition-based protocols include SNMP and SSH. This means that Hyperview can use the Identification information in the model data database to automatically identify and model the device. It will then use the definition to map and query the device sensors. (The definition is based on a Domain Specific Language invented by Hyperview for this purpose.)

Non-discoverable definition-based protocols include Building Automation and Control (BACnet/IP), and Modbus/TCP. These protocols are mostly used for building management and industrial control applications. Since each installation for these systems can be unique, Hyperview allows the user to describe the sensors and their mappings using the Hyperview GUI.

## SNMP TRAPS

**SNMP Traps** are *push notifications* that are sent when a significant event happens, such as a UPS losing power and switching to battery. Trap support and capabilities vary across devices and manufacturers. A trap definition catalogs the various notifications a certain device can send, along with their severity and interpretation.

Trap notification events are mapped to the context of the device using the trap definition. The events are subsequently passed on to Hyperview's notification subsystem.