User Documentation for the Netbox-Plugin DDDC

December 21, 2023

Version 0.95

Contents

Chapter 1

Introduction

Presented herewith is the User Documentation of the Netbox Device Detection and Device Characterization (DDDC) plugin. This plugin is part of the development of an asset discovery and asset characterization solution which has been developed within *Project 507 Part 2* by Fraunhofer IOSB on behalf of the German *Bundesamt für Sicherheit in der Informationstechnik (BSI*). The DDDC plugin takes input data from several sources and supports the processing and approval of these data to build a device database within the framework of Netbox. As such, it is intended to support IT security management task like device management, vulnerability management and patch management as well as intrusion detection systems.

The following topics are addresses within this documentation:

- Chapter 2 provides an overview of the implementation architecture and describes the basic concepts.
- Chapter 3 explains the different ways to import and process raw data.
- Chapter 4 contains detailed description of the different view and models which are implemented by the DDDC plugin.

Chapter 2

Project Overview

The goal of the DDDC project is the collection and the processing of device related data in order to build a device database that supports IT security operations like Intrusion Detection and Prevention, Vulnerability and Patch Management. Figure 2.1 provides an overview of the general architecture, which comprises the following building blocks:

Device Database
: This is the central component and serves as a repository providing device-related information. It is considered to be the source of truth with respect to the IT infrastructure, i.e. it describes the network as it should be.

Database Import Layer
: The import layer provides a set of functionalities to support the import of network related data of various formats.

Network Discovery and Device Characterisation
: This function block takes and analyses packet-related network data (either from online sniffing or stored in PCAP files) and generates a device and topology model of the network.

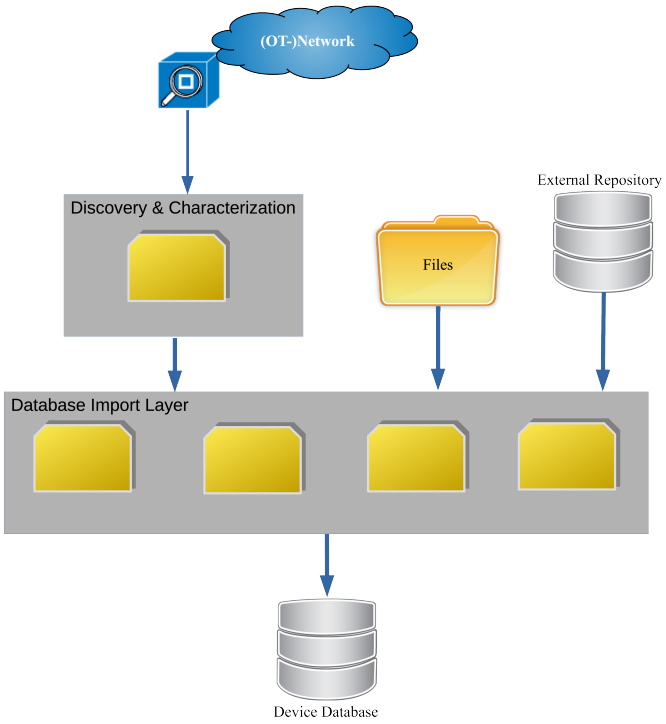

Figure 2.1: The general architecture

Figures 2.2 and 2.3 show the general architecture on the implementation level for the two operation modes (see below).

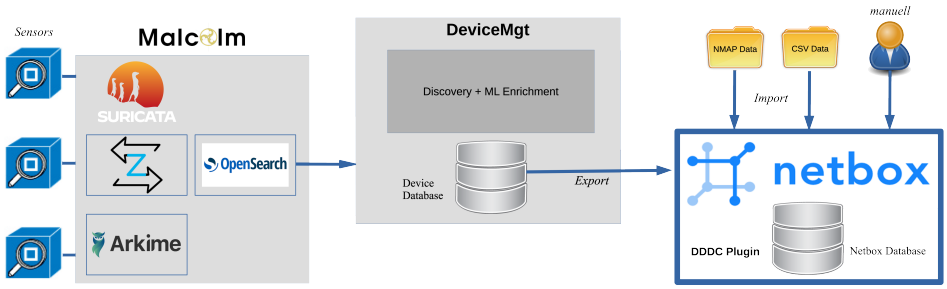

Figure 2.2: Implementation architecture in online mode

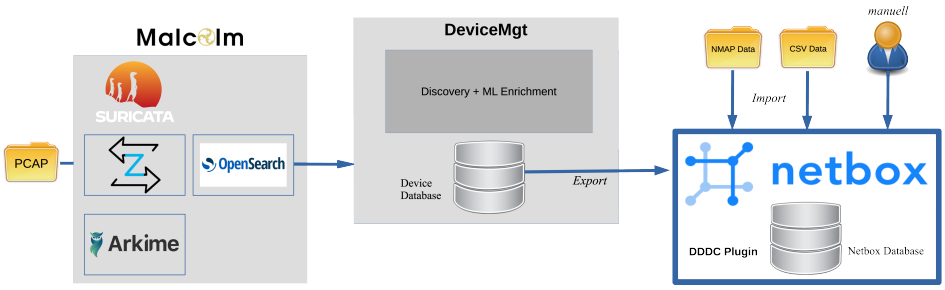

Figure 2.3: Implementation architecture in offline mode

Malcolm
: A network traffic analysis tool. It provides for full packet capture (Arkime) and packet analysis (Zeek). Resulting log messages are stored in and provided through an OpenSearch database.

Device Manager
: This tool reads the log messages from Malcolms OpenSearch database and applies different discovery methods to retrieve information about the existence and the characterization of devices.

Netbox with DDDC-Plugin
: Netbox is a network modelling and documentation solution and serves as the implementation base for the Device Database. The DDDC-Plugin adapts the Netbox Database to the Device Manager, providing the functionality of the Database Import Layer for inputting data from various sources (besides the discovery data from the Asset Manager, table-like data (CSV) and NMAP-XML output formats are supported). Imported data can be harmonised, spelling can be checked and the imports can be approved.

Two modes of operation can be distinguished:

Online Mode
: Malcolm listens to network traffic through its sensors and processes incoming network data continuously. The Device Manager periodically retrieves the respective log messages, updates its internal device model and exports it to the DDDC-Plugin within Netbox (see figure 2.2).

Offline Mode (PCAP Mode
: Offline or PCAP mode provides for the analysis of recorded network traffic, e. g. for forensic analysis (see figure 2.3).

2.1 Devices and Findings

Netbox's data model is based on the device concept. A device represents as a real world entity the building blocks of IT network structures (sometimes called asset). Additionally, the DDDC-Plugin uses the concept of communication, which represents a communication relation between two IP addresses (related to interfaces which in

turn are related to devices).

A finding is a data structure associated with some address information (IP address and/or MAC address) and represents a chunk of information derived by the discovery process in the Device Manager or by some data processed from CSV files or NMAP outputs. The relationship between address-specific findings and devices (one address maps to one device or more than one address maps to the same device) has to be defined by the user. The DDDC-Plugin distinguishes between `DeviceFinding` and `CommunicationFinding`. The processing of the data input and the construction of the device model within the Netbox Database can be outlined in three phases:

Phase 1
: Importing Data to Findings

Phase 2
: Associating Findings with Devices

Phase 3
: Applying Findings to Devices

2.1.1 Phase 1: Importing Data to Findings

The first phase (depicted in figure 2.4) covers the data import from external sources, preprocessing, mapping and the creation of respective finding objects. Three different sources of input data are currently supported:

- Device data from the device database of the device manager
- Raw data from any source in tabular form in CSV format
- Output data of NSE scripts in XML output format of NMAP

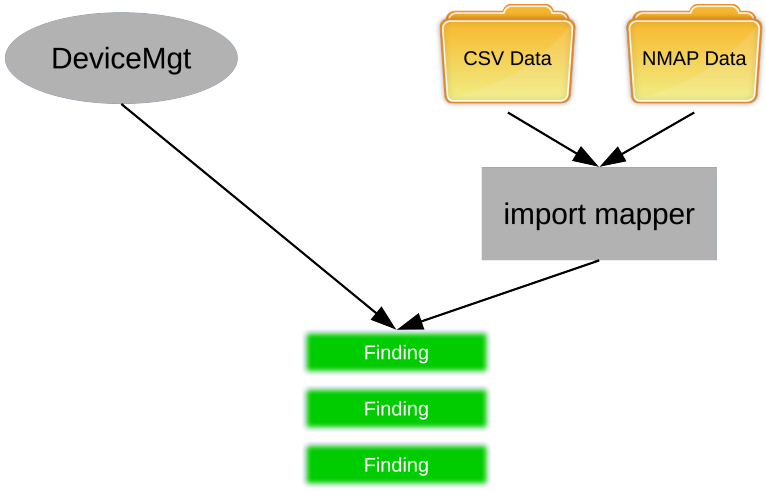

Figure 2.4: Phase 1: Importing Data to Findings

2.1.2 Phase 2: Associating Findings with Devices

The second step involves associating the finding with a device object. Therefore, the finding object must contain address information: an IP address and/or a MAC address, as this serves as the key information to reference a device object. Three scenarios have to be distinguished:

Operation Create Device

A new device object to be created, the device name and the interface name has to be provided. As a result, a new device object and a new interface object are created. If an IP or MAC address is available, additionally a new IP or MAC Address object is created automatically. Additionally, the new device object is linked to objects representing the Unspecified Device Type, Unspecified Manufacturer and Unspecified Device Role. These links are mandatory in the Netbox core model and the value is left to Unspecified until an appropriate finding is applied setting a proper value (see figure 2.5).

Operation Map to Device

This associates a finding with an existing device having the same address information. The finding object will be linked to the respective device object (see figure 2.6).

Operation Edit Device

This associates a finding with an existing device with different address information. This case applies for a device having more than one interface (more than one address). Using the operation Edit Device the device name has to be selected and the interface name has to be provided (see figure 2.7).

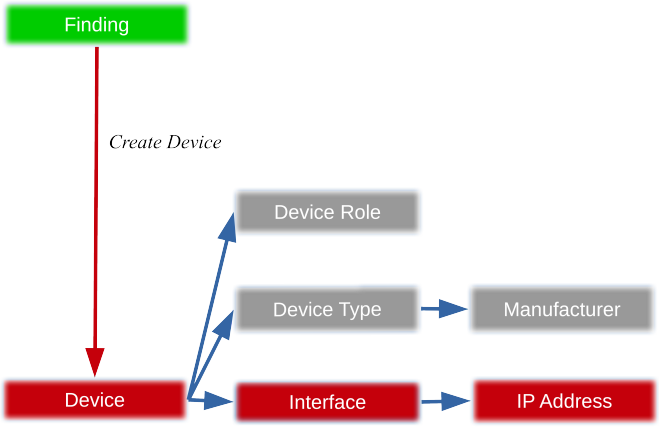

Figure 2.5: Phase 2: Create new device

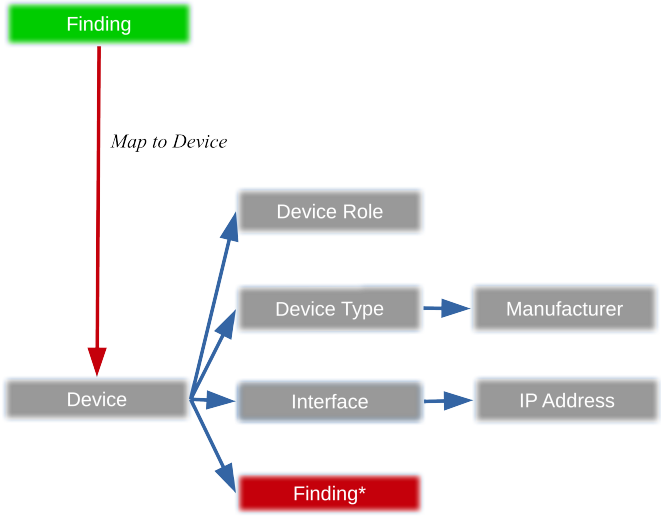

Figure 2.6: Add finding to existing device

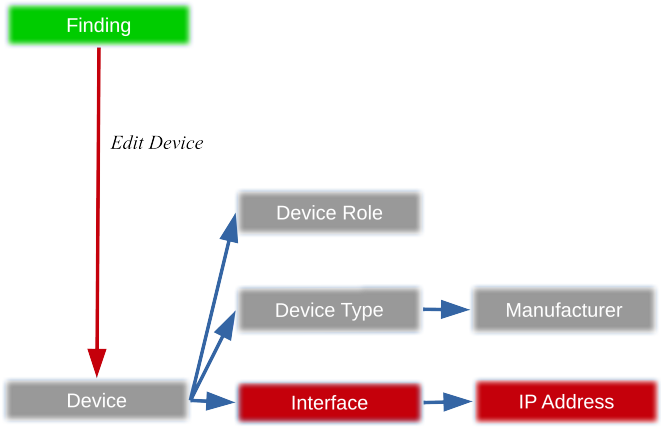

Figure 2.7: Add interface and address information to existing device

2.1.3 Phase 3: Applying Findings to Devices

In last step the findings associated with devices have to be processed. The user has to decided which findings have to be applied and which to be rejected. Here, at last the spellcheck and normalization takes place (see for 4.9.1.1).

Chapter 3

Importing Data

3.1 Importing Asset Manager Data

The Asset Manager Application processes log data from the Malcolm system and provides asset discovery and asset identification. The output consists of device and communication related findings.

The Asset Management Application serves as an interface layer between the Malcolm system and the DDDC-Netbox plugin. It transforms the connection-oriented structure of the log data in the OpenSearch database of Malcolm into the device-oriented view used in the Netbox data model. This role of a mediator is depicted in the figures 2.2 and figure 2.3.

There are 4 possible ways to import from the Asset Manager:

Online Mode
: The Asset Manager and Malcolm are processing online network traffic received from sensor attached to the network. The output data of the Asset Manager is send to the DDDC-Netbox plugin periodically (e.g. every 10 minutes).

PCAP Mode
: Malcolm has processed a PCAP file and the resulting data is stored in the Open Search database of Malcolm. The Asset Manager is started and processes these data. At the end of processing, the output data is send to the DDDC-Netbox plugin.

Database Export
: The content of the Asset Manager's database can be exported to the DDDC-Netbox plugin by invoking the export function from the command line of the Asset Manager machine.

Netbox standard import
: In all 3 cases described above, the Asset Manager writes 2 data files (device finding and communication findings). These files may be imported using Netbox standard import functionality. This is described in section 4.1.2.

3.1.1 Configuring the Asset Manager

The Asset Manager is controlled by the configuration file

`/home/asset-manager/site/config.yaml`.

For the first three use cases (Online Mode, PCAP Mode and Database Export) the following address information of the Netbox server has to be provided:

The IP address of the Netbox host:

```
{netbox-exporter:netbox_host:  <IP Address> }
```

Username and Password of the Netbox user:

```
{netbox-exporter:netbox_username:  <user> }
{netbox-exporter:netbox_passwd:  <password> }
```

The interval for updating Netbox is configured by:

```
{asset-discovery:update_datastore_interval:<interval in seconds>}
```

3.1.2 Online and PCAP Mode

The Online Mode and the PCAP Mode are invoked from the Asset Manager by configuring appropriately and launching the application via the command line interface using the shell script `/home/asset-manager/etc/asset-manager/start.bash`. For more details see the respective documentation [1].

3.1.3 Database Export

The data export can be invoked manually from the command line interface using the shell script `/home/asset-manager/etc/netbox-exporter/export.bash`

3.1.4 Data Files for Netbox Import

The data files to be used for the Netbox Import are located in

`/home/asset-manager/logs/`.

The device findings can be found in the file

`/home/asset-manager/logs/ds-dump-findings.json`,

and the communication finding in the file

`/home/asset -manager/logs/ds-dump-l3_communications.json`.

3.2 Importing CSV data

A detailed description can be found in section 4.1.3.

3.3 Importing XML-based NMAP data

A detailed description can be found in section 4.1.3.

Chapter 4

Views and Models

This chapter offers an overview of the views provided by the DDDC-Plugin, highlighting that the majority of these views are directly derived from a corresponding model. "A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table." [4]. The most significant models introduced by this plugin include:

- DeviceFinding
- CommunicationFinding
- Communication
- Software
- ProductRelationship
- XGenericUri
- Hash
- FileHash

For each of these models, there is a set of essential views provided, which includes functionalities for adding, editing, displaying individual model items, and showcasing all model items in a tabular format. Each of these views is described in its respective section, along with specific additions and features. Moreover, extra attributes, views and functionality have been integrated into existing models within Netbox. These enhancements are described in their corresponding sections. The relevant models include:

- Device
- DeviceType
- Interface

4.1 DeviceFinding

This model is a collection of all device-specific attributes which may be imported and which are relevant to this project. Generally, a finding denotes an observation about a device or asset in the OT environment, originating from various sources like Device Discovery and Characterization, NMAP or operator lists. The core philosophy of this plugin is to initially represent these observations in a way that ensures unambiguous semantics. For instance, the manufacturer of a device may be referred to as 'vendor', 'manufacturer', 'producer', or other terms by different tools and conventions. When importing data from various sources, the first step is to map the items from those sources to the `DeviceFinding` model, where, for example, the attribute for the device's vendor/manufacturer/producer is labeled as 'manufacturer'. Once an observation has been standardized into the format of a `DeviceFinding`, the next step is to assign the finding to the associated device. To facilitate this process, the plugin offers specific functionalities, which will be detailed in this section.

In section 4.1.1, the attributes of the `DeviceFinding` are explained alongside the Add-View[1] provided by the plugin. The process of importing and mapping external data sources, such as NMAP and operator lists, to `DeviceFindings` is outlined in section 4.1.3. Section 4.1.4 offers a comprehensive description of the view that displays all `DeviceFindings` in a tabular format. This view not only allows users to associate findings with devices but also provides functionality for creating a new device within the NetBox database, especially if the device isn't already present, along with standard filter options.

4.1.1 Add-View and Detail-View

The Add-View is utilized to create a new `DeviceFinding` via the user interface. The user can access the Add-View through one of the following three options:

- Append the following url: `/plugins/d3c/findings/add/`
- Click on the green plus symbol located right beside the DeviceFinding-Item in the navigation bar (bottom left corner of figure 4.1)
- The DeviceFinding Table-View, as described in section 4.1.4, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.1.

[1] Usually, instead of manually adding a `DeviceFinding`, users are encouraged to automatically import device findings through the Import-View. Nevertheless, this view can still be helpful for testing purposes.

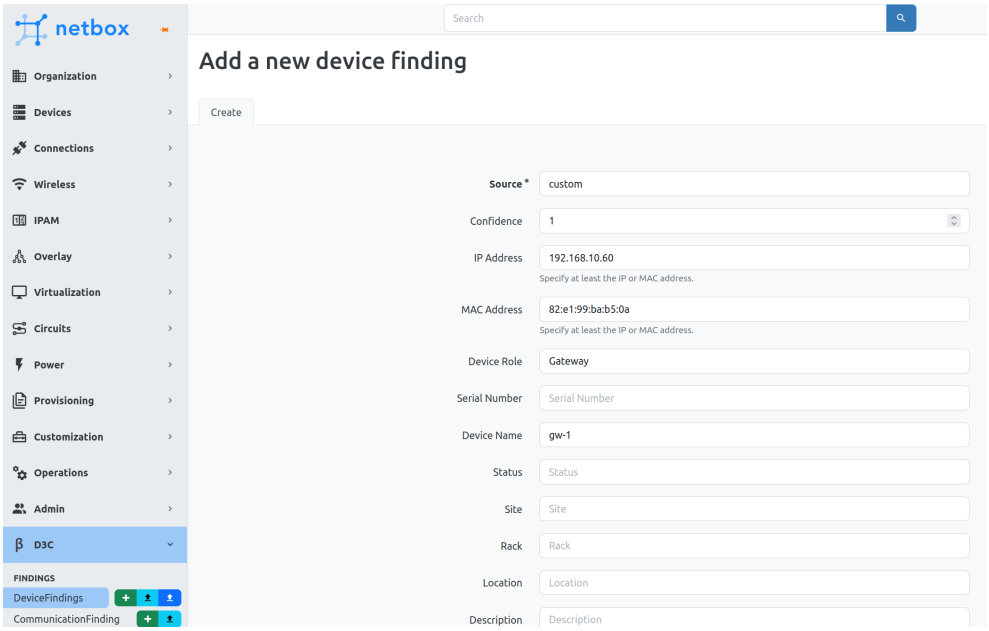

Figure 4.1: Add-View of a `DeviceFinding` (Top)

This view enables the user to enter a total of 30 attributes[2] for a `DeviceFinding`. The following enumeration provides explanations for all these attributes, which are integral to the data model of a `DeviceFinding` and can be customized using this view:

- `Source`: Mandatory text field for detailing the context or source of this finding. For instance, a value like "nmap" can be used to indicate that this finding is based on a nmap scan result.
- `Confidence`: Decimal value (e.g. 0.5) between 0 and 1 which express the confidence of this finding. **Warning:** Commata for decimal values are not supported.
- `IP Address`: The IPv4 address of one of the device's interfaces as text field (e.g. `192.168.0.10`). This attribute or MAC address must be set. This is mandatory because this information is later used to map the DeviceFinding to the corresponding Device object in Netbox.
- `MAC Address`: The 48-bit MAC address of one of the device's interfaces as text field (e.g. `00:B0:D0:63:FF:FF`). This attribute or IP address must be set. This is mandatory because this information is later used to map the DeviceFinding to the corresponding Device object in Netbox.
- `Device Role`: Text field specifies the functional role assigned to the device as defined by Netbox in [12].
- `Device Name`: Text field containing the descriptive name of the device as defined by Netbox in [9].

[2]It is essential to understand that the intention is not for a finding to define all these attributes. In fact, it's quite the opposite; we assume that a finding typically provides only a small subset of these attributes collectively. Therefore, most attributes are optional fields and not mandatory.

- `Serial Number`: Text field for the serial number of the device as defined by Netbox in [13].
- `Device Type`: Text field for the device model name, as defined by Netbox in [19], for the respective device type.
- `Description`: Text field for providing a device description, intended as an additional reminder alongside the device type name (e.g. CPU 414-3 PN/DP Zentralbaugruppe mit: Arbeitsspeicher 4 MB ...).
- `Device Family`: Text field specifies the family of a device type (e.g. `SIMATIC, SCALANCE`).
- `Manufacturer`: Text field for the manufacturer name accountable for producing this device type, as defined by Netbox in [18].
- `Is Safety Critical`: This text field should be set to `True` if the device is used for safety functionality, `False` if it is not, and it can also be left empty if the information is not known.
- `Network Protocol`: Text field specifies the network protocol used by one of the device services (e.g. `ipv4`). This attribute should be specified in combination with at least the IP address and probably the Transport Protocol, Application Protocol and Port. The concept of a service is defined by Netbox in [14].
- `Transport Protocol`: Text field specifies the transport protocol used by one of the device services (e.g. `tcp`). This attribute should be specified in combination with at least the IP address, and probably the Network Protocol, Application Protocol and Port. The concept of a service is defined by Netbox in [14].
- `Application Protocol`: Text field specifies the application protocol used by one of the device services (e.g. `http`). This attribute should be specified in combination with at least the IP address, and probably the Network Protocol, Transport Protocol and Port. The concept of a service is defined by Netbox in [14].
- `Port`: Text field specifies the port number used by one of the device services (e.g. `http`). This attribute should be specified in combination with at least the IP address, and probably the Network Protocol, Transport Protocol and Application Protocol. The concept of a service is defined by Netbox in [14].
- `Is Router`: This text field specifies whether one of the device's interfaces is a router interface or not. This attribute should always be specified in combination with at least the MAC address. Valid values are `Yes, No` and `Maybe`
- `Status`: Text field for the device's operational status as defined for Netbox in [16].
- `Site`: Text field specifies the name of the site in which the device is located as defined by Netbox in [15].
- `Rack`: Text field specifies the name of the rack within which the device is installed as defined by Netbox in [11].

- `Location`: Text field specifies the name of the location where the device resides within the assigned site as defined by Netbox in [8].
- `Article Number`: Text field specifies the stock keeping unit (SKU). It can be the same as model number (NetBox: part_number), especially when seller is the vendor itself.
- `Part Number`: Text field specifies the model number of the manufacturer as defined by Netbox in [10].
- `Hardware Version`: Text field specifies the hardware version of the device type.
- `Hardware CPE`: Text field specifies the Common Platform Enumeration (CPE) string of the device.
- `Software Name`: Text field specifies the name of a software component installed on the device.
- `Is Firmware`: Text field specifies whether the specified software name is a firmware component. This field should only be set if a software name is specified.
- `Version`: Text field specifies the version the specified software name. This field should only be set if a software name is specified.
- `Exposure`: Text field specifies the grade of exposure to other networks. Valid values are `Small`, `Indirect` and `Direct`[3].
- `Finding Status`: For internal purposes, it should always be set to the value `NEW`.
- `Tags`: Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`.

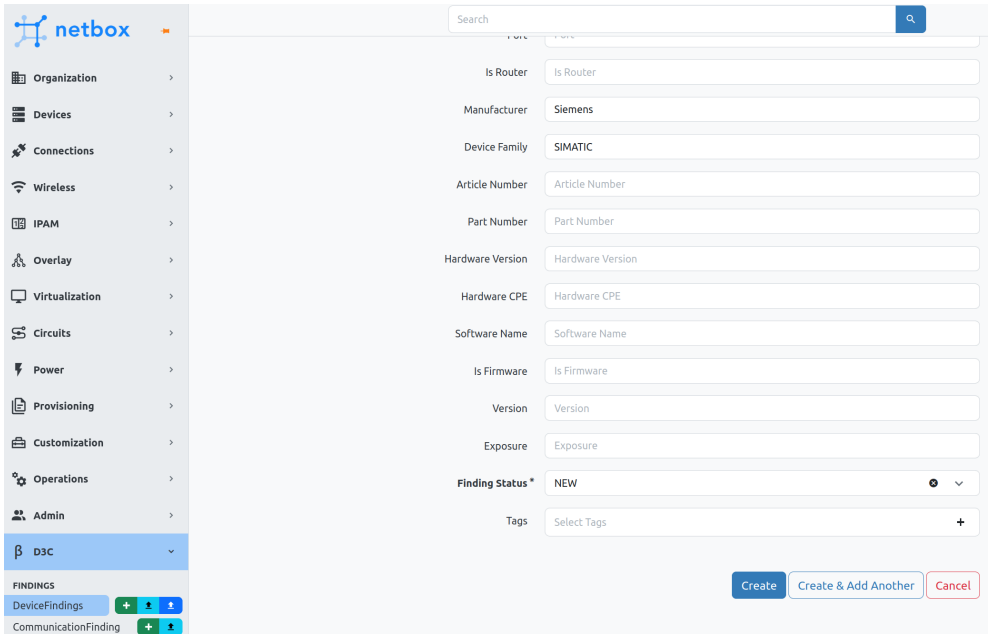

Figure 4.2: Add-View of a `DeviceFinding` (Bottom)

[3] A descriptions of the values can be found within Netbox under the respective custom field.

After setting the values for a Device Finding, it can be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.2. The creation can also be canceled using the corresponding button. The creation of a `DeviceFinding` will fail if any of the mandatory fields, `Source`, `Finding Status`, or either of the attributes `IP Address`or `MAC Address` is missing. Additionally, it will fail if a confidence value outside the range of 0 to 1 is set. The form will highlight the input boxes accordingly if any error occurs. If the creation is successful, the user will be navigated to the detail view of the DeviceFinding, as illustrated in figure 4.3. In this figure, a `DeviceFinding` with the ID 173 was created. The `Source` attribute was set to 'custom', and the confidence value is 1. The service fields indicate an HTTP service. However, no device is associated with the `DeviceFinding`, but an IP and MAC address are set accordingly to enable a lookup for the corresponding device. The lookup mechanism will be described in section 4.1.4. No other attributes of the DeviceFinding were set, as shown in figure 4.3.

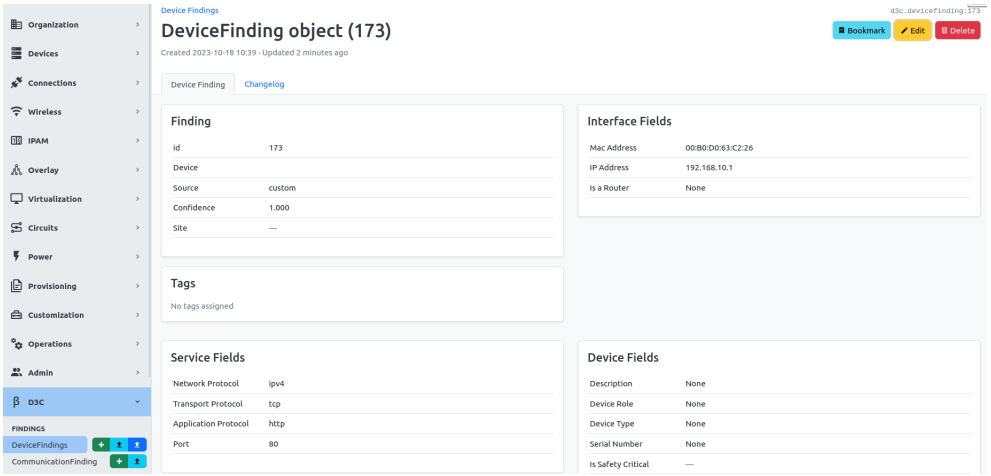

Figure 4.3: Detail-View of a single `DeviceFinding`

4.1.2 Standard Import-View

One option for importing findings information from the Asset Manager (as described in section 3.1) involves utilizing data files and the Netbox standard bulk import function, outlined in [3]. This function can be invoked from the navigation bar or from the Table-View of the `DeviceFindings` or the `CommunicationFindings`. The Import-View is shown in figure 4.4. An example of an import file is shown in listing 4.1. In addition, examples for possible input data can be found under `https://github.com/DINA-community/DDDC-Netbox-plugin/tree/main/data`. Furthermore, in case of a csv file the delimiter ";" has to be used.

```
[
    {"oui": "PHOENIX CONTACT Electronics GmbH",
    "source": "MacDetect",
    "manufacturer": "PHOENIX CONTACT Electronics GmbH",
    "device_type": "MGUARD RS4004 TX/DTX",
    "confidence": "1.0",
```

```
"mac_address": "a8:74:1d:8c:FF:FF",
"ip_address": "192.168.11.1"},
{"source": "AssetDiscovery",
"confidence": "1.000",
"ip_address": "192.168.11.1",
"mac_address": "a8:74:1d:8c:FF:FF"},
{"port": 123,
"confidence": "1.000",
"network_protocol": "ipv4",
"transport_protocol": "udp",
"source": "AssetDiscovery",
"ip_address": "192.168.11.1",
"application_protocol": "ntp"}]
```

Code Listing 4.1: Example of the device findings import file in JSON format

The allowed attributes for import are identical to those of the `DeviceFinding` object, which are described in section 4.1.1. Moreover, the Asset Manager utilizes the following attributes to import data into Netbox: `source`, `ip_address`, `mac_address`, `manufacturer`, `network_protocol`, `transport_protocol`, `application_protocol`, `port`, `device_role`, `confidence`, `device_type` and `is_router`.

For the import of the data files the tab 'Upload File' has to be selected (the file has to reside on the machine, on which the browser is running). Within this tab, the data file and the data file format, which is 'JSON' in the figure, has to be selected. How to locate the data file for the asset manager is described in section 3.1.4. The 'Submit'-Button starts the import.

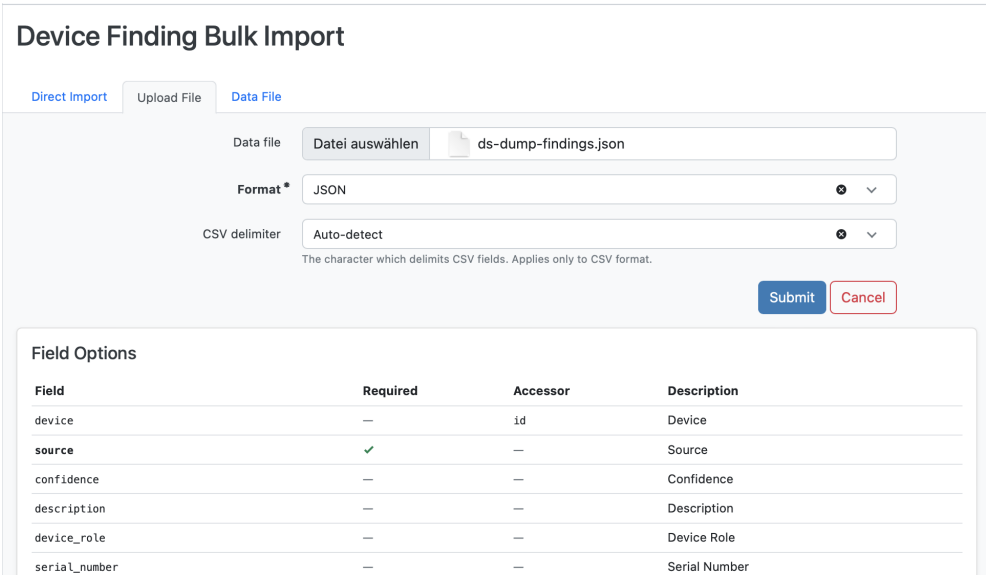

Figure 4.4: The `DeviceFinding` bulk import

4.1.3 Import and Mapping-View

When DeviceFinding data is available in NMAP-XML format or in CSV files that don't precisely adhere to the Netbox model, the *Import and Mapping-View* allows

flexible mapping of such input data to the Netbox model. The actual translation from the source data to the `DeviceFinding` is controlled by templates. Each field of the `DeviceFinding` model has one template that determines how the field is filled from the source data. A set of templates that specifies how a source file is translated into Findings is called a *Mapping*. To open the *Import and Mapping-View*, a user can either:

- Append the following url: `plugins/d3c/findings/import/`
- Click the dark-blue (rightmost) up-arrow beside the DeviceFinding-Item in the navigation bar (bottom left corner of figure 4.5)

Importing and mapping DeviceFindings is a multi-step process. After initially opening the importer, the user is presented with the form shown in figure 4.5. Here, the user can either paste raw data into the *Raw Data* text box, or select a data file by clicking the *Browse* button. The data format must also be set to either *CSV* or *NMAP* in the *Format** option. The *Validate* button uploads the data and performs an initial validation. Examples for possible input data can be found under `https://github.com/DINA-community/DDDC-Netbox-plugin/tree/main/data`

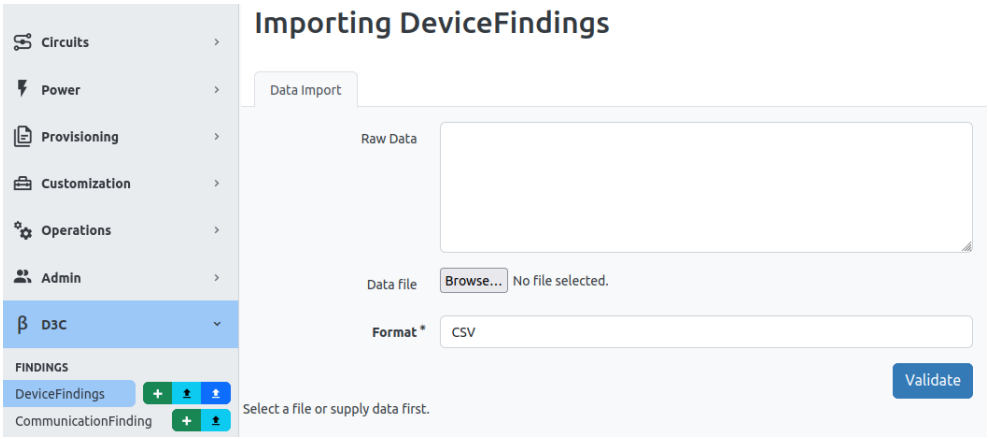

Figure 4.5: Initial `DeviceFinding` Import and Mapping View

After the inital validation, the full mapping form is shown and the data can be mapped to the `DeviceFinding` model. The mapping form has 5 groups of controls, shown in figure 4.6

Input Data The input data that needs to be mapped to the `DeviceFinding` model. Either *Raw Data* or a *Data file* can be supplied. If both are supplied, the *Data file* take precedence. If a *Data file* is given, after validation, the contents of the file are transferred to the *Raw Data* box.

Command Buttons These buttons control the import. Their function is explained below.

Load/Save Mapping These controls allow for the storing and retrieval of mappings (sets of templates), so they can be re-used for later imports. To store a mapping, give a name in the *Save Mapping* text field and click the *Save Mapping* button. If a mapping with the given name and given *Format* already exists, it

is overwritten. To load a mapping, select the mapping to be loaded in the *Existing Mappings* dropdown box and click *Load Mapping*. Any current templates will be replaced with the loaded templates. To remove a stored mapping, select the mapping to be removed in the *Existing Mappings* dropdown box and click *Delete*.

Detected Headers Here the headers are listed that have been detected in the CSV file and that can be used in mappings.
Note: Only columns that have actual data in them are listed.

Templates The templates that control how data is transferred from the source data to the Device Findings. After clicking the *Validate* command button, the templates are applied to the first data row in the source data and the result is shown behind the template.

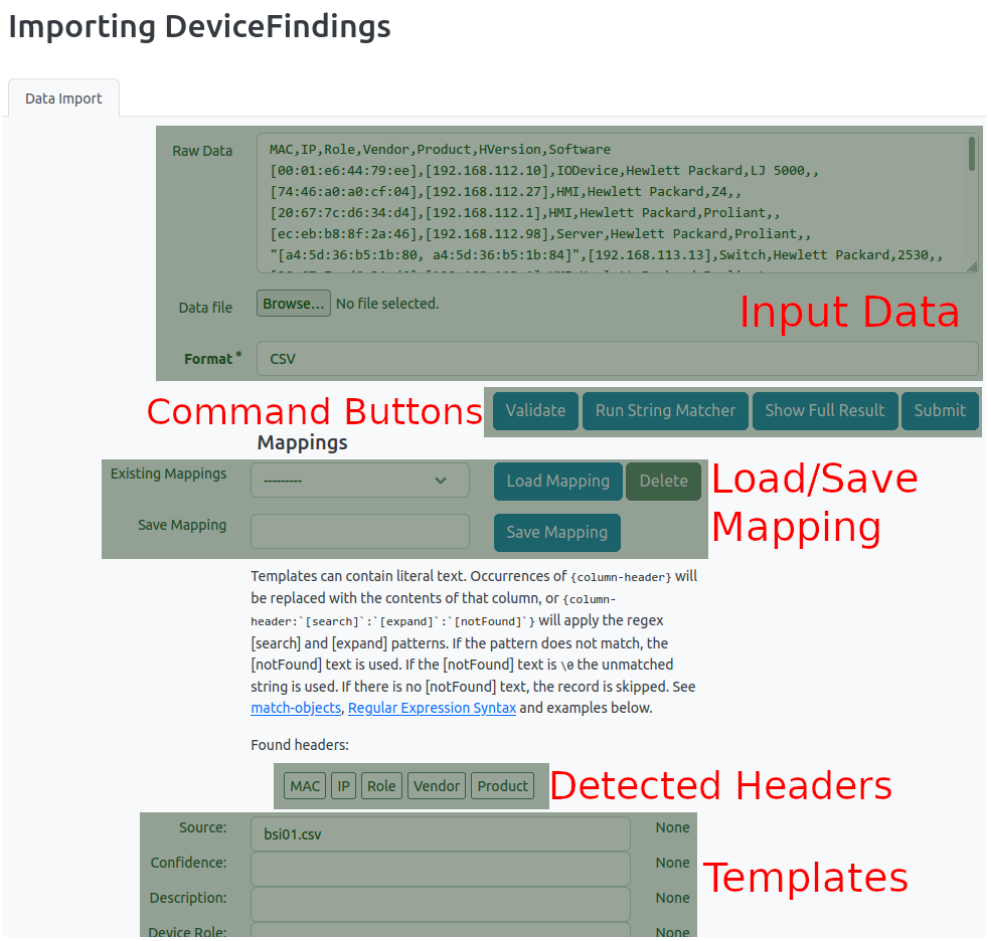

Figure 4.6: Full `DeviceFinding` Import and Mapping View

The command buttons have the following function:

Validate Applies the current mapping to the first row of data and shows the result next to the mappings. This does not change any data and is safe to execute.

Run String Matcher Tries to find common Device Families, Device Types, Article Numbers and Versions in each data line, and adds them as extra CSV columns.

This does not change any data and is safe to execute.

Show Full Result Applies the defined mappings to the entire data file and shows the full results as a table below the mappings. This does not change any data and is safe to execute.

Submit Submits the data and mapping and generates Device Findings from them.

The templates that determine the actual translation from the source data to the `DeviceFinding` are listed in the form. Each field of the `DeviceFinding` model has one template that determines how the field is filled from the source data. Templates can contain literal text with placeholders. Placeholders take one of the following forms:

`{column-header}`
: The placeholder will be replaced with the content of the column with the name `column-header`. If there is no data in this column for the current record, the result is left empty.

`{column-header:‘[search]‘:‘[expand]‘}`
: Will apply the regular expression `search` to the column `column-header`. The result of this regular-expression search is applied to the `[expand]` template and the placeholder is replaced with the result of this expansion. If the regular expression does not match the content of the column, the entire record is skipped and now `DeviceFinding` is produced for this record.

`{column-header:‘[search]‘:‘[expand]‘:‘[notFound]‘}`
: Same as above, but when the regular expression does not match, the literal text in `[notFound]` is used instead. If the `[notFound]` text is `\0` then the (unmatched) content of the column `column-header` are used as if the template was `{column-header}`.

The ‘ character in the templates is the `back-tick` character [24]. More information on regular expression syntax can be found in [22] and more information on match expansion can be found in [21]. Below the templates is a table with examples. Clicking the *Show Full Result* button will apply the mapping to the full data set and show the results in a table below the templates. A column named *Error* is added to this table, listing the regular expressions that failed to match and that did not have a `[notFound]` entry.

4.1.4 Table-View and Device Lookup

This view presents all DeviceFindings in tabular form. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/findings/`
- Click on the DeviceFinding item inside the navigation bar as illustrated in figure 4.7. Right next to the item, three buttons with different colors and symbols are available.

Figure 4.7: Navigation bar with the 'DeviceFinding' item allowing the navigation to the `DeviceFinding` Table-View. Right next to the item, three buttons with different colors and symbols are available: (Green) Navigation to the Add-View described in section 4.1.1 (Light blue) Navigation to the default Import-View all Netbox models provide. (Dark blue) Navigation to the Import and Mapping-View described in section 4.1.3.

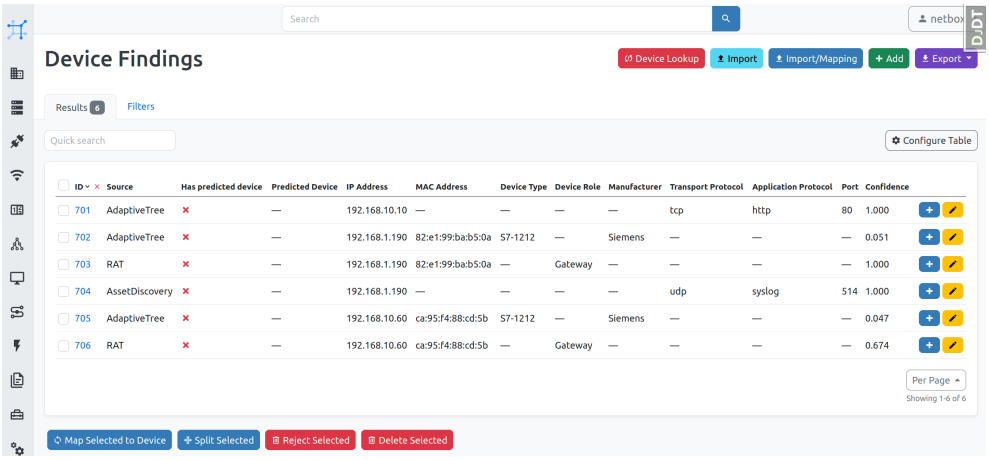

Figure 4.8: Table-View of `DeviceFindings`

This view primarily serves the purpose of assigning findings to devices in the Netbox device database. It is illustrated in figure 4.8. At the top right corner, there are five buttons, each of which is described below:

- Device Lookup: This button initiates a lookup for all `DeviceFindings` displayed in this table. It searches for a device in the Netbox device database based on the IP and/or MAC addresses, if available. Please note that this lookup is resource-intensive and may take some time to complete.
- Import: Navigation to the default Import-View all Netbox models provide.
- Import/Mapping: Navigation to the Import and Mapping-View described in section 4.1.3
- Add: Navigation to the Add-View described in section 4.1.1
- Export: Providing default export functionality for Table-Views in Netbox.

The first action a user should take when navigating to this view is to perform a Device Lookup. Using the IP and/or MAC Address information provided by each `DeviceFinding`, the associated device inside Netbox is identified. This action often leads to adjustments in the view, as illustrated in figure 4.9. If a device is found for a `DeviceFinding`, a green checkmark is displayed under 'Has predicted device', and the specific device name is shown under 'Predicted Device'. In figure 4.10, for

three DeviceFindings, a Device named 'Device 1' was found. Afterwards, the user has the opportunity to check if this mapping is correct. If the device prediction for a `DeviceFinding` is correct, the user can map it to the device by selecting the finding via the checkbox at the beginning of the row and clicking the blue 'Map Selected to Device' button in the lower-left corner. Afterward, the mapped DeviceFindings are no longer displayed in the Table-View and can be further processed directly in the Device-View as described in section 4.9.

Tip: If all predictions are correct, simply type 'True' inside the 'Quick search' textbox and select all findings by clicking on the checkbox in the header, as illustrated in figure 4.10

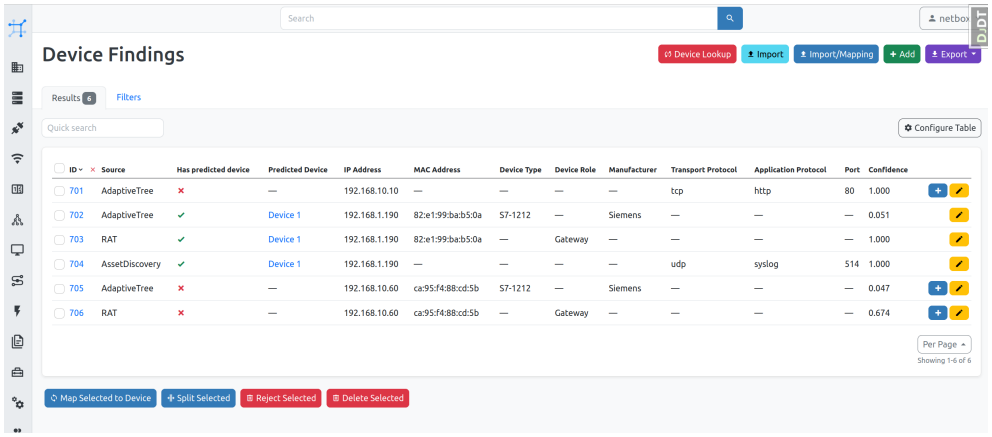

Figure 4.9: Table-View after a Device Lookup.

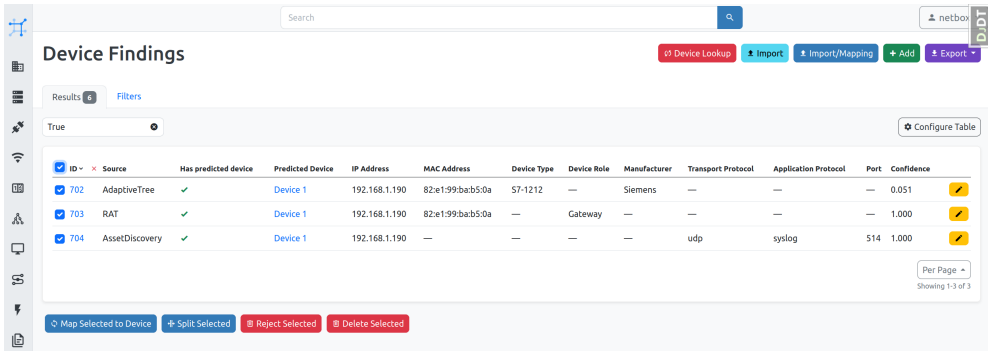

Figure 4.10: Using the "Quick search" to filter for all `DeviceFindings` with a predicted Device.

Figure 4.11 illustrates the Table-View after mapping has been performed. In most cases, the Device Lookup will not find for every `DeviceFinding` a corresponding device. In this case the user has multiple options to handle such a `DeviceFinding`:

- Create Device: If the user realizes that the device is missing in the database, they can manually add the device via the standard views in Netbox or use the procedure described in section 4.1.4.1.
- Edit Interface for Device: The user may realize that the IP or MAC Address for

the corresponding interface of the device is misconfigured in Netbox. In such cases, the user can manually modify the interface through the standard views in Netbox or follow the procedure described in section 4.1.4.2.

- Reject or Delete the `DeviceFinding`: The user can delete or reject the DeviceFinding using the corresponding red button at the bottom of the view. The user should always prefer to reject a DeviceFinding. This has the advantage that an identical DeviceFinding appearing a second time with the same values will be ignored.

4.1.4.1 Create Device based on DeviceFinding

If a user realizes that a device needs to be created based on a `DeviceFinding`, they can utilize the Table-View. For instance, consider `DeviceFinding` with the ID 705, as shown in figure 4.11. This `DeviceFinding` suggests the presence of a Siemens S7-1212 device with the IP address 192.168.10.60. If this device is missing in the database, the user can click the blue plus button in the corresponding row. Upon clicking the button, the user will be presented with a view similar to figure 4.12. In this view, the user can input a device and interface name, such as 'Device 2' for the device name and 'eth0' for the interface name. Optionally, the user can add a comment for the new device. Clicking the 'Create' button will generate the Device[4] with the specified name and, if available, set the interface with the MAC and IP Address.

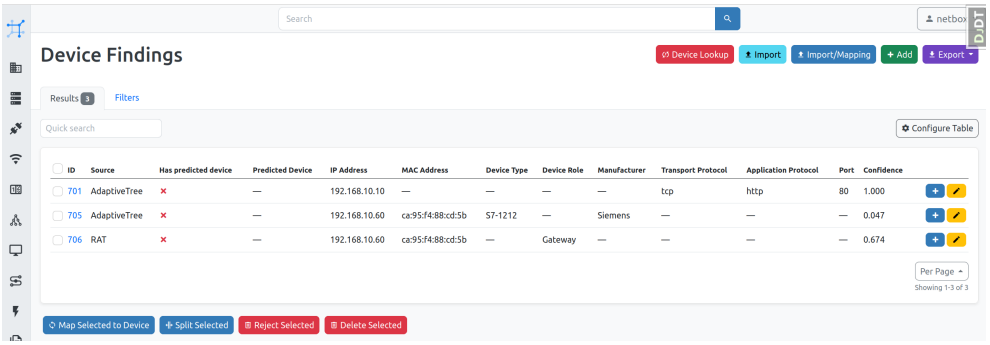

Figure 4.11: Table-View after clicking on the "Map Selected to Device' button and deleting the filter

[4]The Netbox Device model includes mandatory fields, such as Device Type. When creating a device with this view, all mandatory fields are initially set to 'Unspecified'.

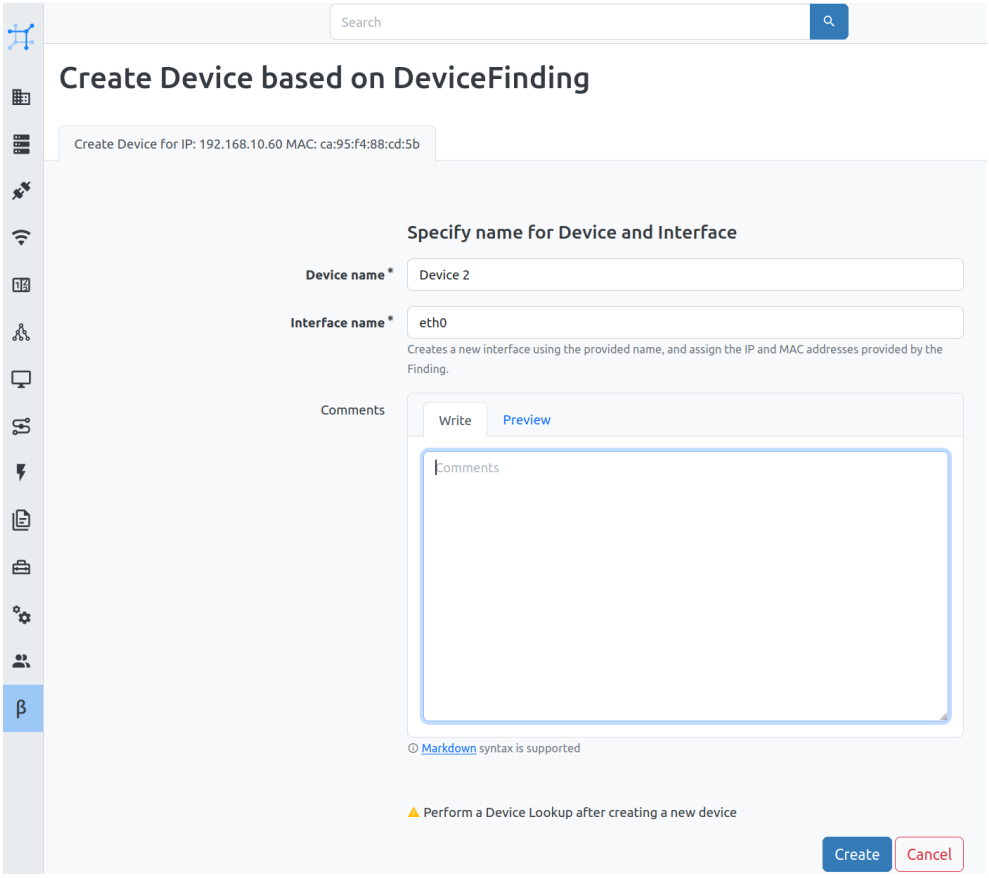

Figure 4.12: Create-View based on a `DeviceFinding` via the Table-View

After creating the new device, the user is returned to the Table-View, which appears identical to its previous state. To view the new device and its corresponding mapping, the user must click on the 'Device Lookup' button. The result will resemble something like what's shown in figure 4.13. The `DeviceFinding` with ID 705 can now be mapped to the device, as well as the `DeviceFinding` with ID 706.

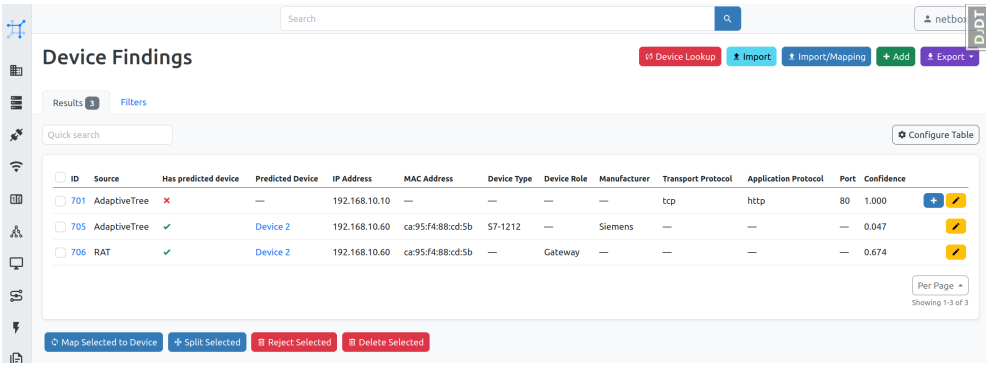

Figure 4.13: Table-View for `DeviceFindings` after a DeviceLookup.

4.1.4.2 Edit or create Interface based on DeviceFinding

If a user realizes that the interface of a device needs to be adapted or a new interface has to be created according to a `DeviceFinding`, they can utilize the Table-View. For instance, consider the `DeviceFinding` with the ID 701, as shown in figure 4.13. The user may notice that the device with the IP Address 192.168.10.10 is also belonging to 'Device 2', but the IP Address belongs to a missing interface. In such cases, the user can click on the yellow button on the right of the row. This action will take the user to a view similar to the one shown in figure 4.14. In this view, the user can edit the IP and/or MAC Address of the `DeviceFinding`. Moreover, the user can select the device associated with the `DeviceFinding` and enter the interface name for the new interface (e.g., 'eth1' as illustrated in figure 4.15). As depicted in the figures, if the interface name doesn't exist, it is created, and the respective IP and/or MAC address from the `DeviceFinding` is assigned to it. If the MAC Address of the interface hasn't been set, it is updated; however, if it's already set, a new IP is generated and then assigned to the interface. After clicking on the 'Save' button, the interface is created and the user is then navigated to the Table-View. A Device Lookup must be performed before the changes to the 'Predicted Devices' attribute can be seen. This is illustrated in figure 4.14 by the orange text.

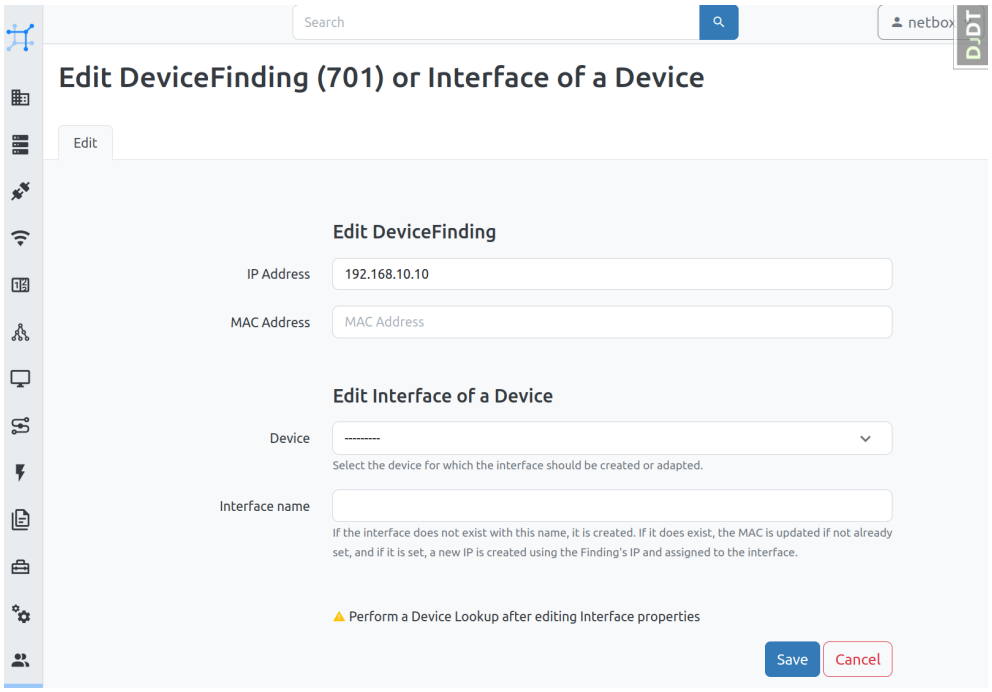

Figure 4.14: Edit Interface of Device based on `DeviceFinding` (1).

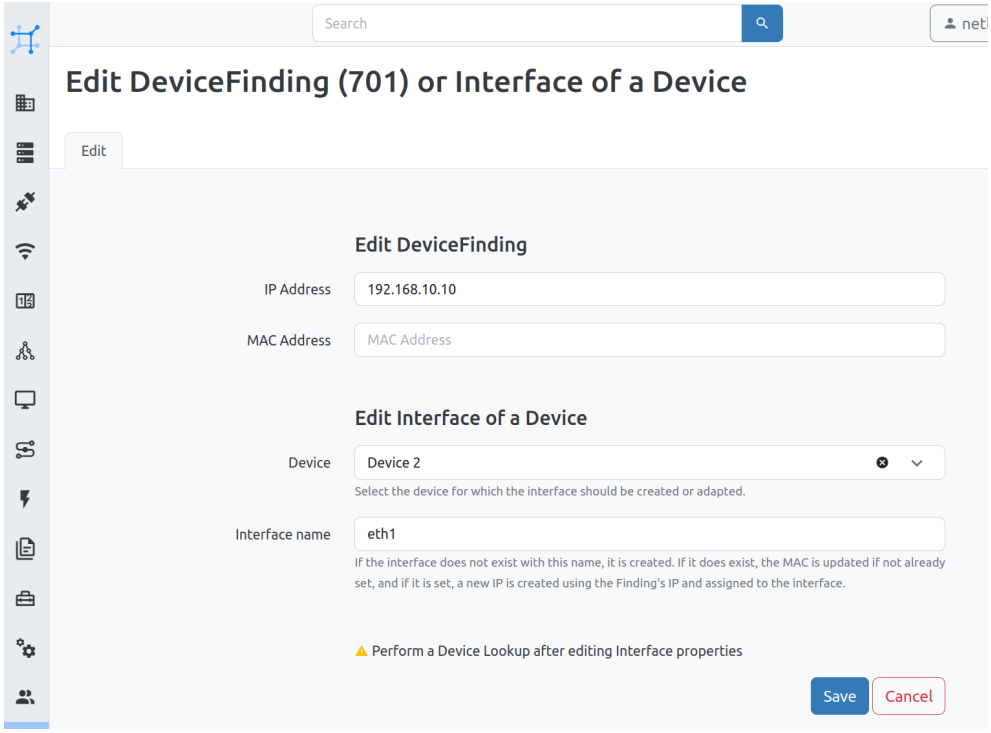

Figure 4.15: Edit Interface of Device based on `DeviceFinding` (2).

4.1.4.3 List of IP and MAC Addresses

The DDC3-plugin supports only single values for an IP or MAC address. However, as emerging use cases involve multiple IP and MAC addresses, this view provides a 'Split Selected' button, as illustrated in figure 4.16. This feature assumes that the first item in the IP Address list and the first item in the MAC Address list logically belong to the same interface, followed by the second item of the IP list to the second item of the MAC list, and so forth. Clicking the 'Split' button clones the corresponding finding according to the length of the IP/MAC lists. Based on the `DeviceFinding` depicted in figure 4.16, containing two valid IP and MAC address values, the utilization of the 'Split Selected'-button generates two new Findings derived from this initial Finding. The preceding Finding is automatically rejected, if the creation was successful, as illustrated in figure 4.17.

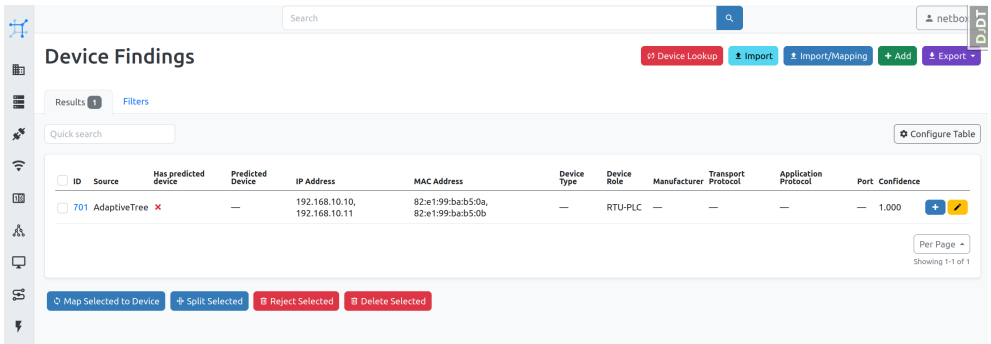

Figure 4.16: Split lists of IP and MAC Address values `DeviceFinding` (1).

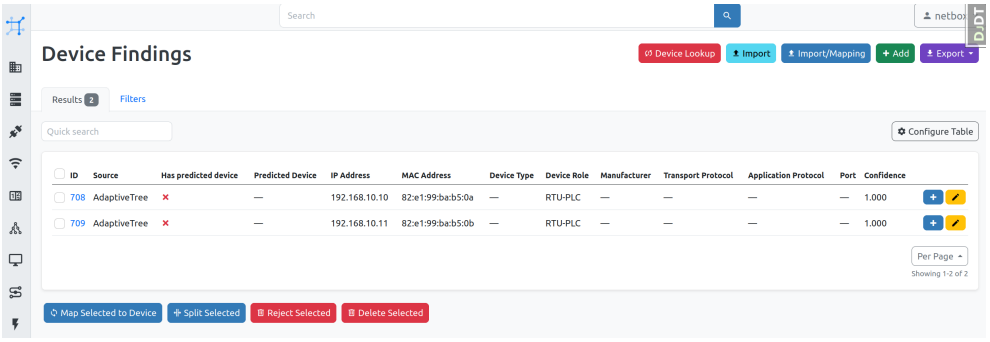

Figure 4.17: View after splitting the `DeviceFinding` (2).

4.2 CommunicationFinding

A `CommunicationFinding` denotes a observation about a communication relation between two addresses (at least specified by a pair of IP address and port number).

4.2.1 Table-View

This view presents all `CommunicationFindings` in tabular form. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/communication_finding/`
- Click on the `CommunicationFinding` item inside the navigation bar as illustrated in figure 4.18. Right next to the item, two buttons with different colors and symbols are available.

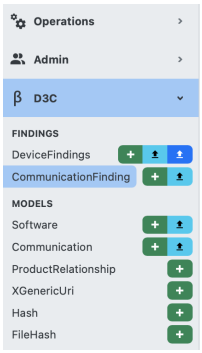

Figure 4.18: Navigation bar with the 'CommunicationFinding' item allowing the navigation to the `CommunicationFinding` Table-View. Right next to the item, two buttons with different colors and symbols are available: (Green) Navigation to the Add-View described in section 4.2.1.2 (Light blue) Navigation to the default Import-View all Netbox models provide.

4.2.1.1 Edit-Button

The Edit button located on the right side of the line in the Table View offers the standard Netbox functionality for editing an object.[5]

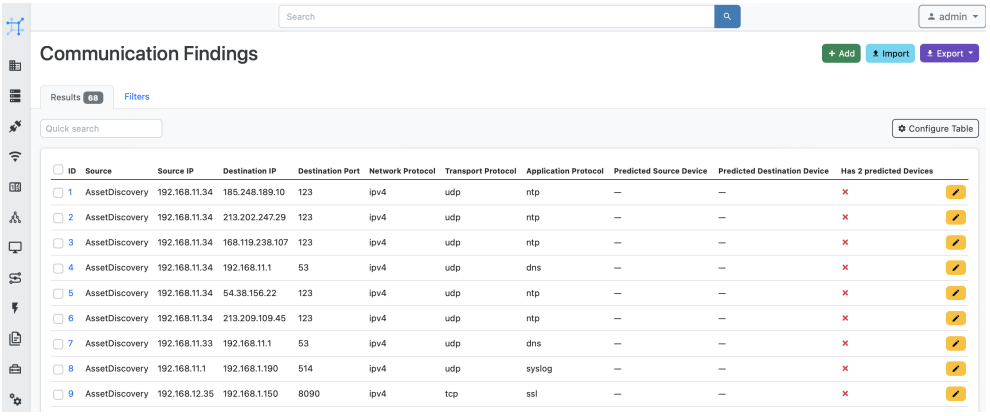

Figure 4.19: Table-View of `CommunicationFindings`

4.2.1.2 Add-Button

The Add-Button (accessible from the Navigation bar or from the Table-View) allows for the creation of a new `CommunicationFinding` object.[6]

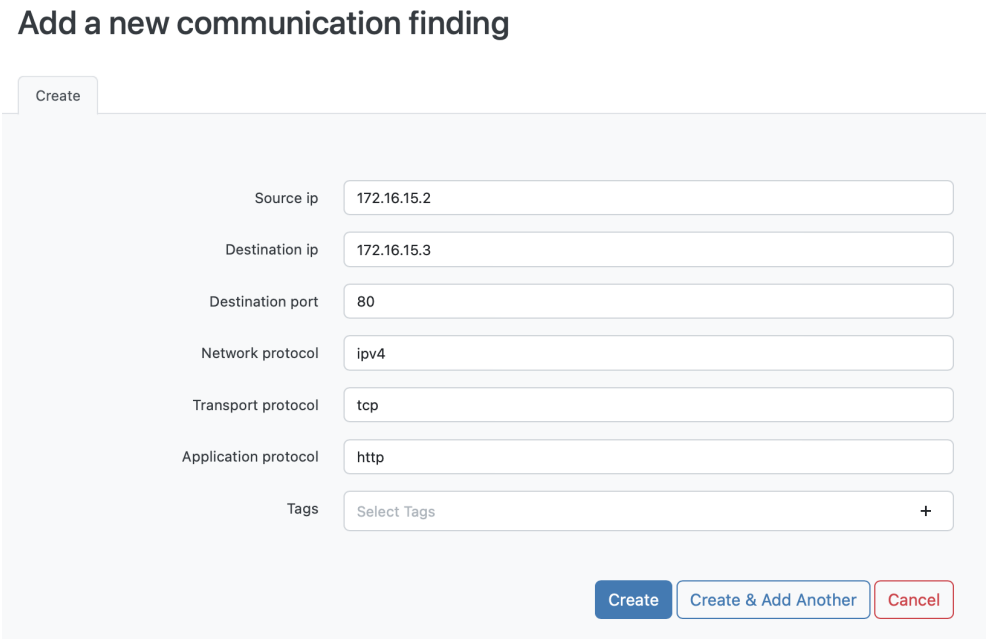

Figure 4.20: Add-View for the Creation of a new `CommunicationFinding`

[5] Editing of `CommunicationFinding` objects is not considered to be useful in normal operation mode.

[6] Creation of new `CommunicationFindings` is not considered to be useful in normal operation mode.

4.2.2 Standard Import-View

One option to import the findings information from the Asset Manager (as described in section 3.1) is the use of data files and the Netbox standard bulk import function. This function can bei invoked from the navigation bar or from the Table-View of the `DeviceFindings` or the `CommunicationFindings`. The Import-View is shown in figure 4.21

For the import of the data files the tab 'Upload File' has to be selected (the file has to reside on the machine, on which the browser is running). Within this tab, the data file and the data file format, which is 'JSON', has to be selected. How to locate the data file for the asset manager is described in section 3.1.4. The 'Submit'-Button starts the import.

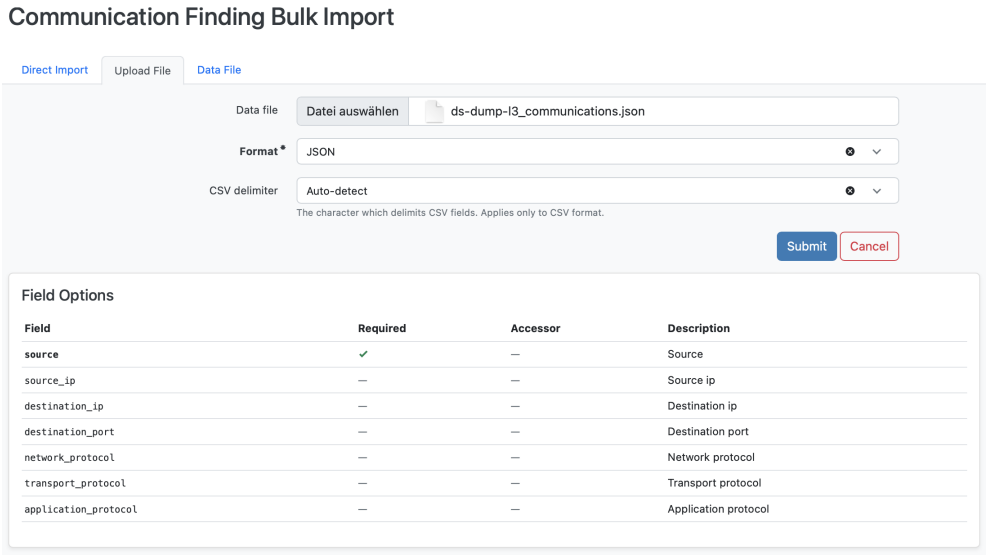

Figure 4.21: The `CommunicationFinding` bulk import

4.2.3 Detail-View

The user can access the Detail-View through one of the following two options:

- Append the following url: `/plugins/d3c/communication_finding/<ID>/`
- Click on the `CommunicationFinding` ID (the most left column in the table view)

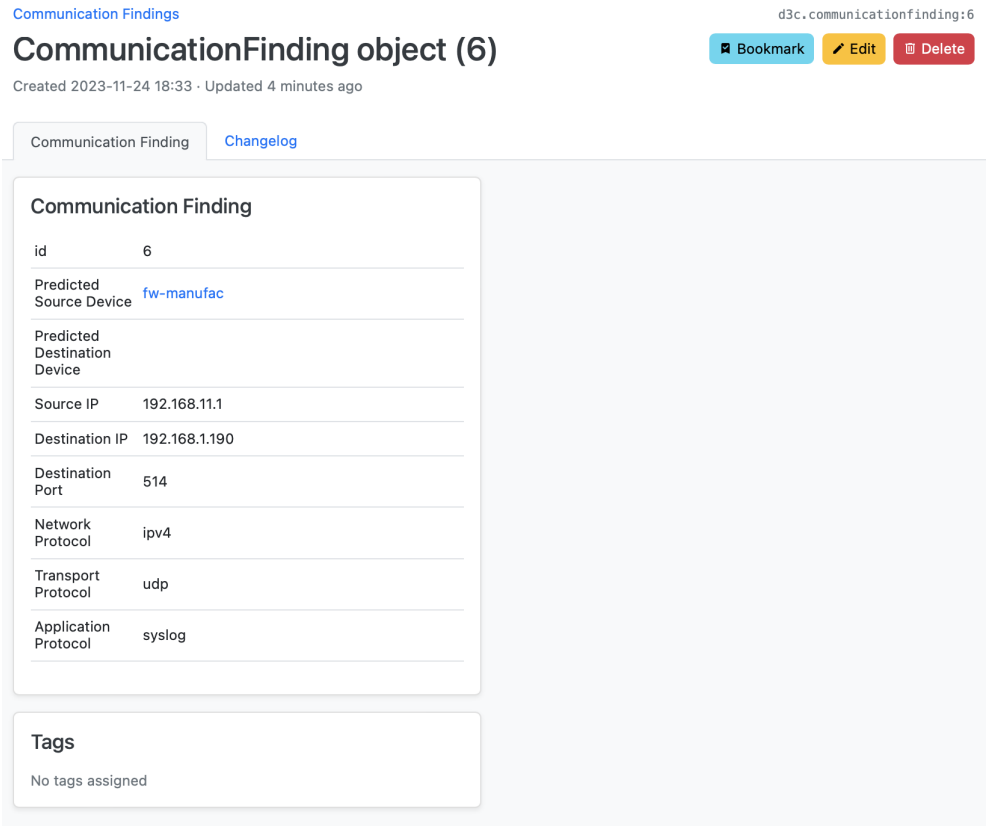

Figure 4.22: Detail-View of a single `CommunicationFinding`

The following are the attributes of the Detail-View:

- `Predicted Source Device`: The device object having the source IP address associated to one of its interfaces, if such is exists.
- `Predicted Destination Device`: The device object having the destination IP address associated to one of its interfaces, if such exists.
- `Source IP`: The IPv4 address of the source of this communication relation.
- `Destination IP`: The IPv4 address of the destination of this communication relation.
- `Destination Port`: The port number on the destination side of the communication relation (i.e. the server port).
- `Network Protocol`: The network protocol used by this communication relation.
- `Transport Protocol`: The transport protocol used by this communication relation.
- `Application Protocol`: The application protocol used by this communication relation.
- `Tags`: Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`.

4.2.3.1 Edit-Button

The Edit-Button in the top-right corner provides the standard Netbox functionality of object editing.[7]

4.2.3.2 Delete-Button

The Delete-Button in the top-right corner provides the standard Netbox functionality of object deletion.[8]

4.2.4 Mapping CommunicationFindings to Devices

Upon creating a device, the `CommunicationFindings` are updated, and the attributes `Predicted Source Device` and `Predicted Destination Device` are set if their corresponding IP addresses match those associated with a device[9]. Moreover, if both devices are found, the flag `Has 2 Predicted Devices` is set. An example is shown in figure 4.23.

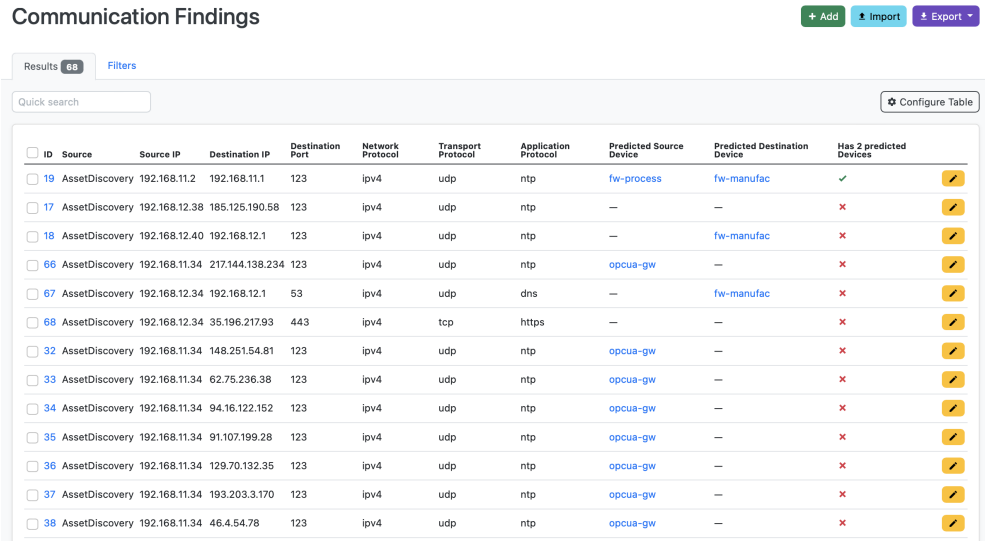

Figure 4.23: The list of `CommunicationFindings` with predicted devices

The filter tab (figure 4.24) enables the filtering of the list of `CommunicationFindings`. The example demonstrates selecting all `CommunicationFindings` that have both predicted devices set.

Remark: Netbox filtering is based on pure string matching, no wildcards are supported. Of particular importance for network management applications like this, there is no knowledge concerning the IP address space, i.e. it is not possible to filter on certain IP address ranges.

[7] Editing of `CommunicationFinding` objects is not considered to be useful in normal operation mode.

[8] Deletion of any finding objects destroys the `CommunicationFinding` processing within this plugin and should be avoided.

[9] All `DeviceFindings` should be processed before the assigning `CommunicationFindings`. This ensures that the majority of devices are known.

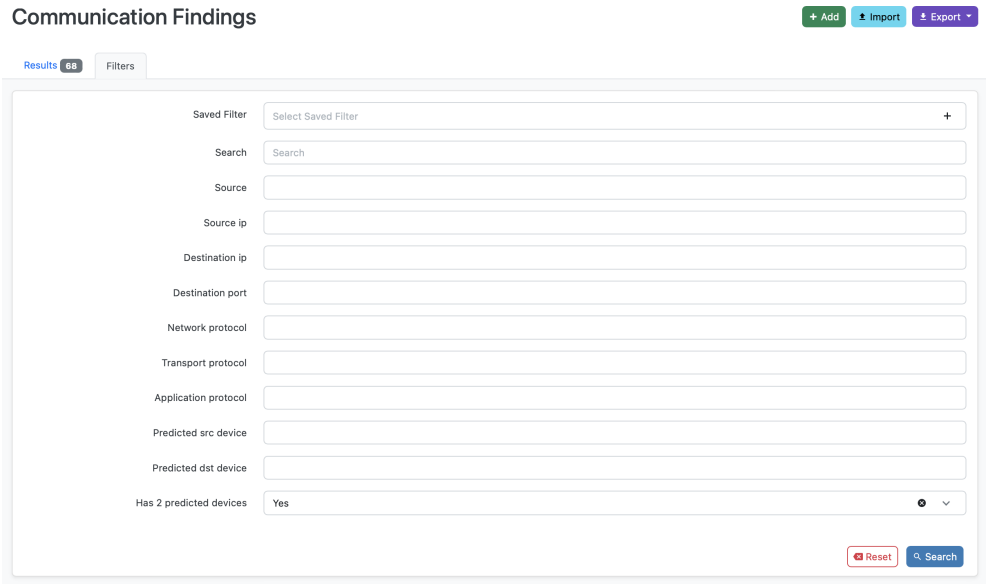

Figure 4.24: The filter view to search for `CommunicationFindings` having 2 predicted devices

From the filtered view, specific lines or all lines can be selected. The blue button Map Selected Device in the bottom line invokes the mapping. This perfoms the following actions:

- From the data in the `CommunicationFinding` object a `Communication` object is created and linked to the respective device objects.
- the `CommunicationFinding` object is marked DONE (i.e. the attribute `Finding Status` is set to DONE).

The red button Reject Selected rejects the selected `CommunicationFindings` (i.e. the attribute `Finding Status` is set to REJECT).

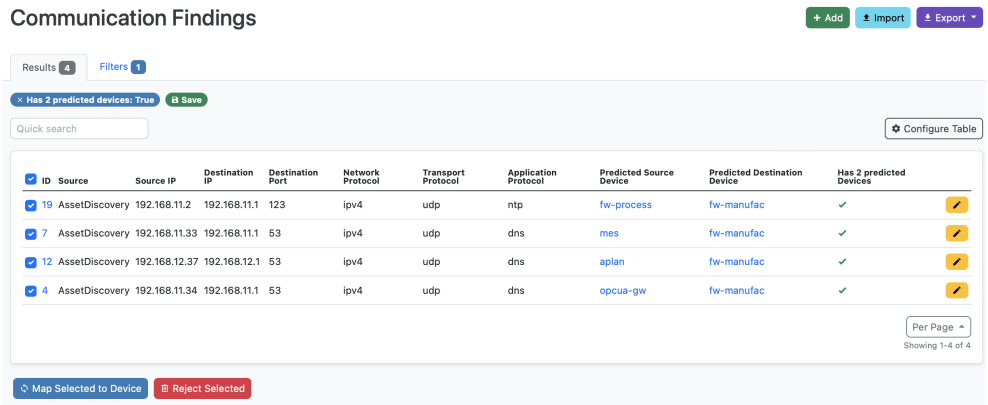

Figure 4.25: The list of `CommunicationFindings` having 2 both source and target device predicted

4.3 Communication

`Communication` objects and the respective model represent communication relations between a pair of addresses (IP address and Port number). Additionally information concerning the used protocol may be provided.

4.3.1 Table-View and Detail-View

The Table-View presents all `Communication` objects in tabular form. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/communication/`
- Click on the `Communication` item inside the navigation bar as illustrated in figure 4.26.

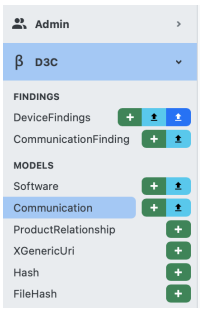

Figure 4.26: Navigation bar with the 'Communication' item allowing the navigation to the `Communication` Table-View. Right next to the item, two buttons with different colors and symbols are available: (Green) Navigation to the Add-View described in section 4.3.2 (Light blue) Navigation to the default Import-View all Netbox models provide.

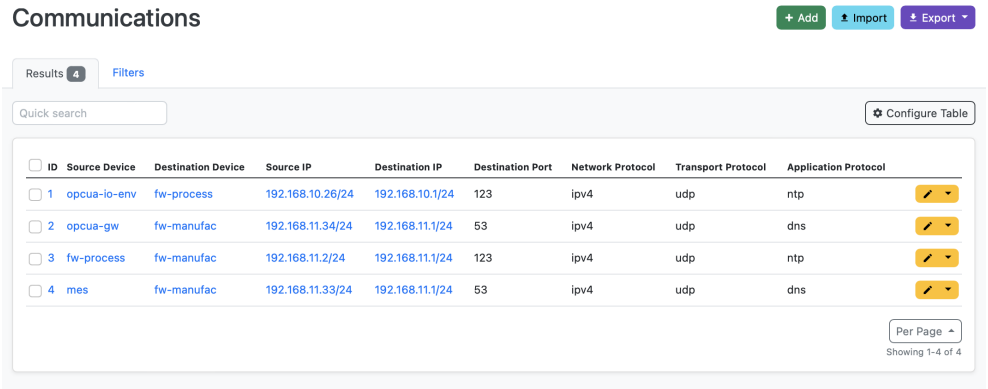

Figure 4.27: Table-View of the `Communication` objects

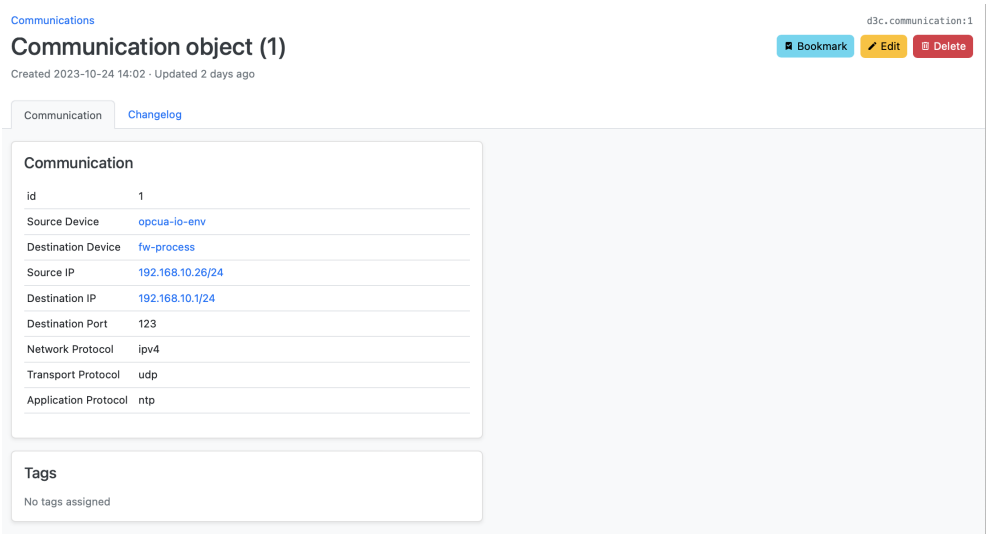

Figure 4.28: Detail-View of the `Communication` object

The Detail-View can be invoke through the following options and is shown in figure 4.28:

- Append the following url: `/plugins/d3c/communication/<ID>/`
- Click on the Communication ID (the most left column in the table view)

The following list shows the attributes of the `Communication` object.

- `Source Device`: The source device of this communication relation
- `Destination Device`: The destination device of this communication relation
- `Source IP`: The IPv4 address of the source of this communication relation.
- `Destination IP`: The IPv4 address of the destination of this communication relation.
- `Destination Port`: The port number on the destination side of the communication relation (i.e. the server port).
- `Network Protocol`: The network protocol used by this communication relation.
- `Transport Protocol`: The transport protocol used by this communication relation.
- `Application Protocol`: The application protocol used by this communication relation.
- `Tags`: Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`.

4.3.2 Add and Edit-View

The user can create a new `Communication` object using the Add-View, which can be invoked through one of the following options:

- Append the following URL: `/plugins/d3c/communication/add/`

- Click on the green plus symbol located right beside the 'Communication' Item in the navigation bar, illustrated in figure 4.26.
- The `Communication` Table-View, as described in section 4.3.1 features a green action button labeled 'Add' that allows to navigate to the view shown in figure 4.27

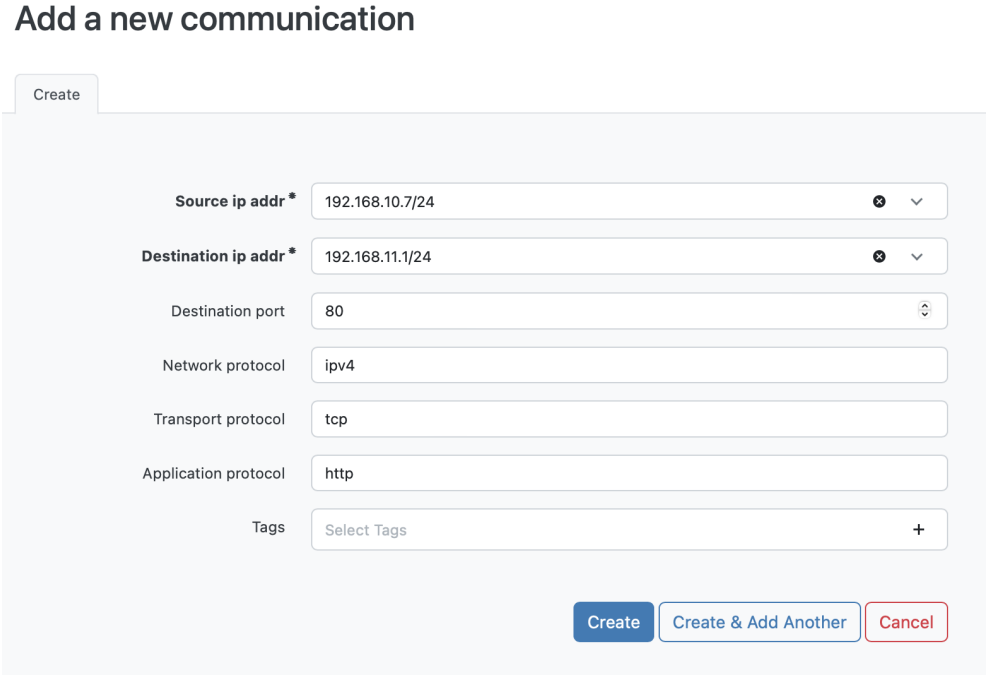

Figure 4.29: Add-View of the `Communication` object

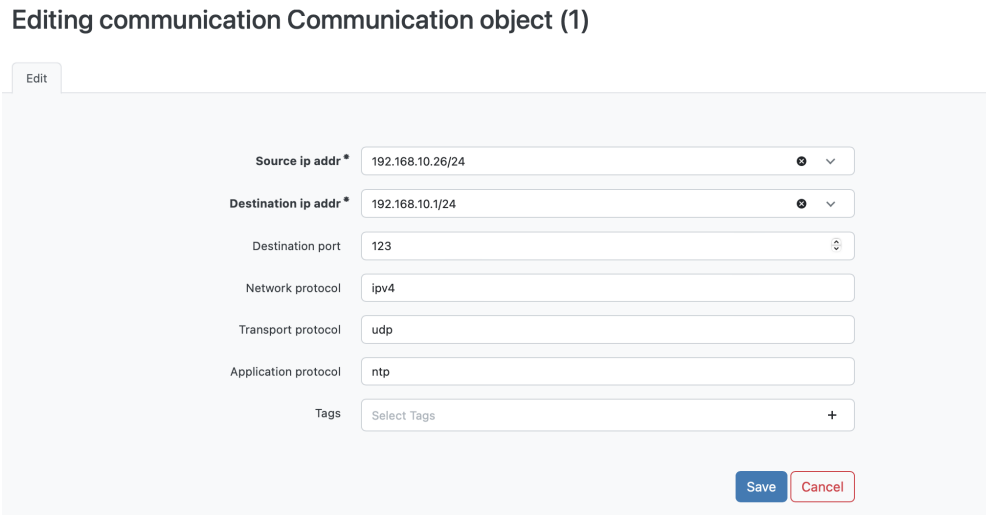

Figure 4.30: Edit-View of the `Communication` object

The Add-View is show in figure 4.29. Source and destination address information is mandatory. The related devices are determined automatically to preserve consistency.

The Edit-View is shown in figure 4.30, the attributes are the same as above.

4.3.3 Communication-View of the Device Object

The Detail-View of the Device object has two tabs related the communication relationships:

- 'Client Communications' show those communication relations in which the current device takes the client role (see figure 4.31)
- 'Server Communications' show those communication relations in which the current device takes the server role (see figure 4.32)

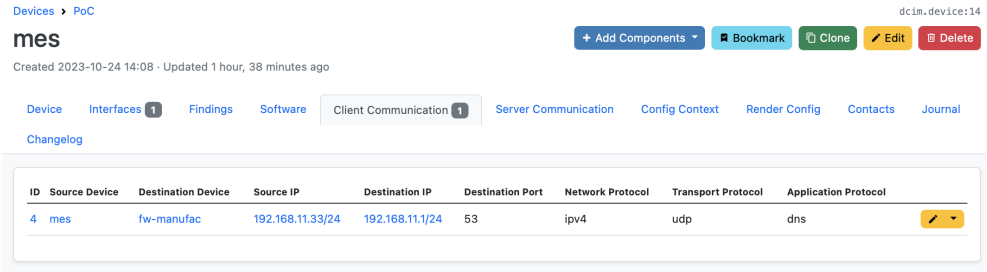

Figure 4.31: Communications with the Device in the client role

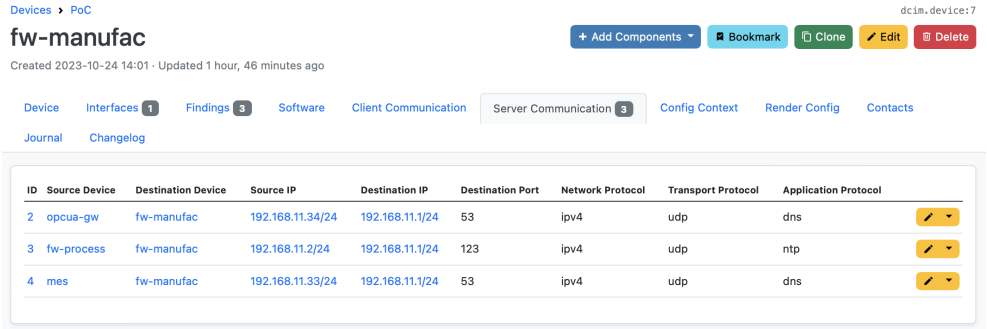

Figure 4.32: Communications with the Device in the server role

4.4 Software

Software is a crucial part when modeling assets. One reason for modeling software components is their necessity with respect to the vulnerability handling process on the operator side in an industrial environment. Netbox does not provide a software data model by default. Therefore, the DDDC-Plugin introduced such a model. In the following subsections, the Add, Detail, Table, and Edit-View are described.

4.4.1 Add, Edit and Detail-View

The user can create a new software object using the Add-View, which can be accessed through one of the following three options.

- Append the following url: `/plugins/d3c/software/add/`
- Click on the green plus symbol located right beside the Software-Item in the navigation bar, illustrated in figure 4.33.

- The Software Table-View, as described in section 4.4.2, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.34.

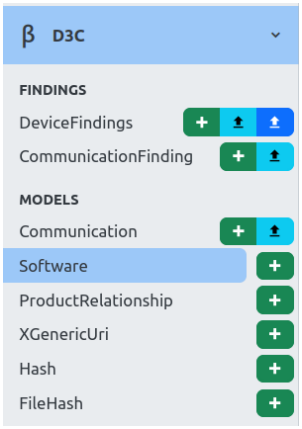

Figure 4.33: Navigation bar including the 'Software' item, which allows users to navigate to the Table-View, described in section 4.4.2. Right next to the item, the green button with the plus icon enables the user to navigate to the Add-View.

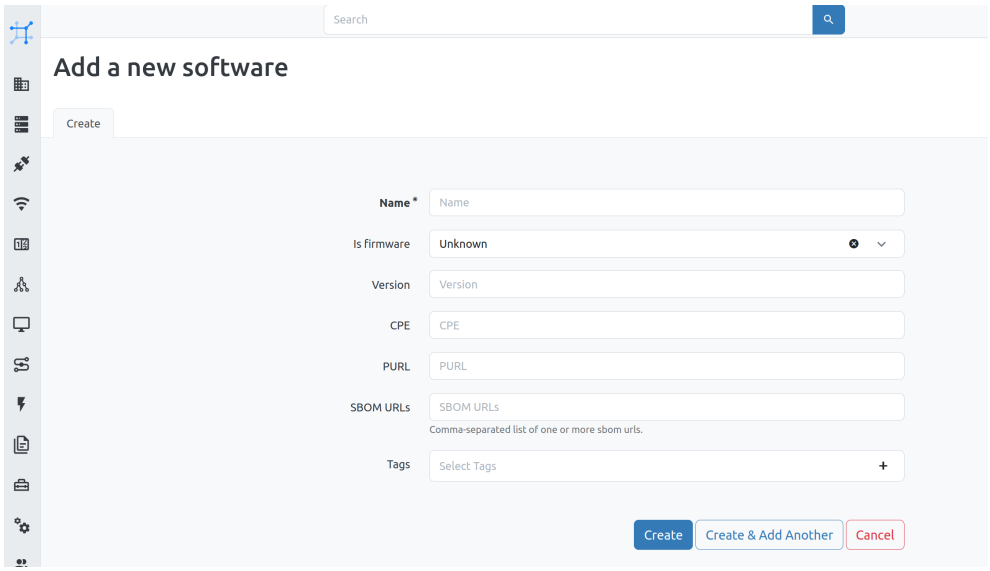

Figure 4.34: Add-View of Software

This view enables the user to enter the following attributes w.r.t. a software object:

- `Name`: Text field specifies the software name. This field is mandatory.
- `Is Firmware`: Choice field which is optional. Can be set to 'Yes', 'No', or 'Unknown'.
- `Version`: Text field specifies the version of the software. This field is optional.
- `CPE`: Text field specifies the Common Platform Enumeration (CPE) string of the software. This field is optional. If the user provides a value, the form validates

whether the string constitutes a valid CPE.

- `PURL`: Text field specifies the package url (PURL). This field is optional. If the user provides this string, the form validates whether the string constitutes a valid PURL.
- `SBOM URLs`: Text field specifies one or more SBOM URLs. Multiple URLs must be entered with commas separating them. This field is optional. Each provided string is validated to determine if it constitutes a valid URL.
- `Tags`: Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`.

After setting the values for a Software, it can be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.34. The creation can also be canceled using the corresponding button.

The creation will fail if the software name is not specified or the values for CPE, PURL or SBOM are invalid. The form will highlight the input box accordingly if any error occurs. If the creation is successful, the user will be navigated to the detail view of the Software object, as illustrated in figure 4.35. In this figure, a Software with the ID 1 was created. The `Name` attribute was set to `Windows`, the version to '10', and a `CPE` was specified. In addition to the already described attributes of software, the following three fields can be found on the Detail-View:

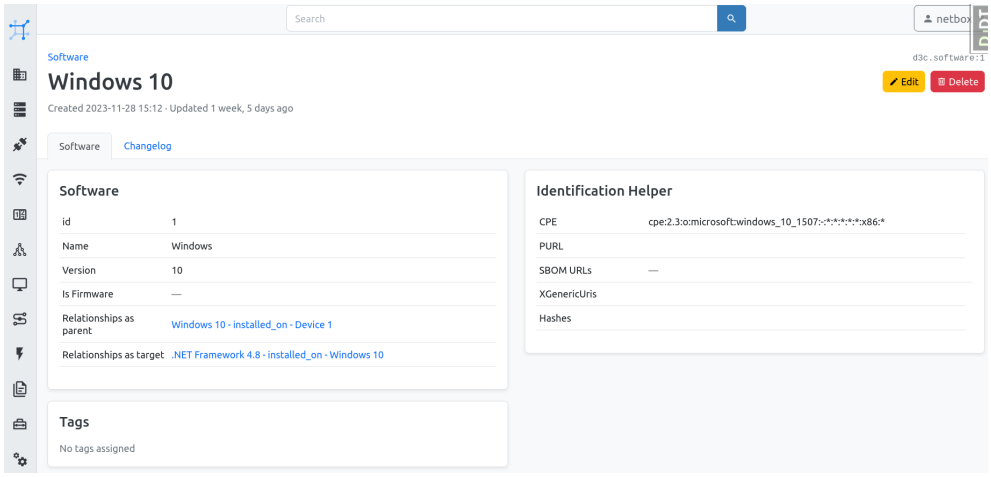

Figure 4.35: Detail-View of Software

- `Relationship as parent`: All `ProductRelationships` specified with this software as the parent object are listed in this field. `ProductRelationships` are described in section 4.5
- `Relationship as target`: All `ProductRelationships` specified with this software as the target object are listed in this field. `ProductRelationships` are described in section 4.5
- `XGenericURIs`: All generic URIs (XGenericURIs) for this software are listed in this field. XGenericURIs including their creation are described in section 4.6.

- `Hashes`: All Hashes specified for this software are listed in this field. Hashes including their creation are described in section 4.7.

On the top right corner of the Detail-View in figure 4.35, the following two action buttons are available:

- Edit: A click on this button navigates the user to the edit page of a software object, as illustrated in figure 4.36. The Edit-View is identical to the Add-View, with the only difference being the presence of filled-out attributes. Changes can be saved by clicking the 'Save'-button.
- Delete: A click on this button deletes the object after a warning message is displayed to the user.

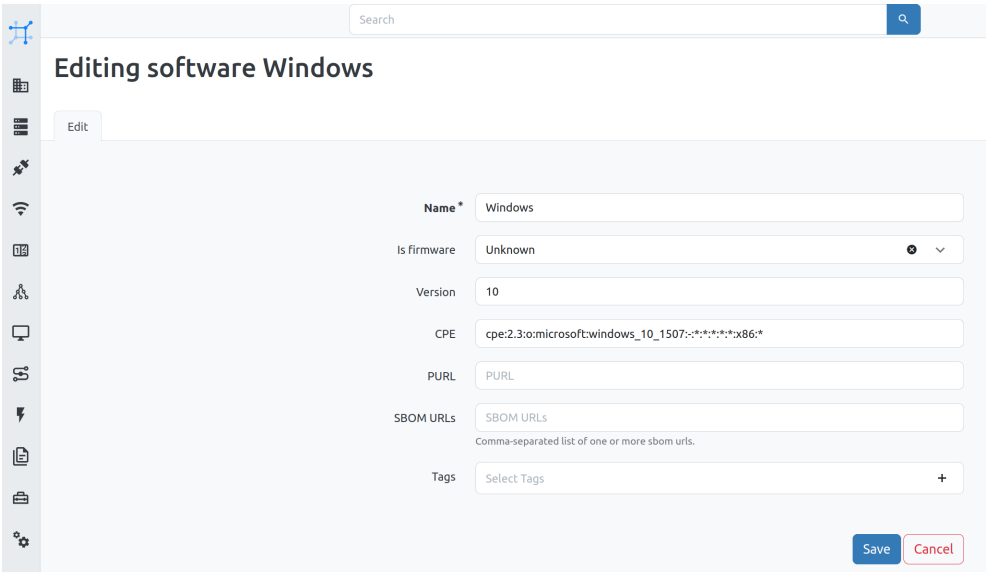

Figure 4.36: Edit-View of Software

Furthermore, there is an additional tab named 'Changelog' available. Opening this tab provides the user with an overview of all changes made to this software item.

4.4.2 Table-View

The Table-View of the software data model is illustrated in figure 4.37. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/software/`
- Click on the Software-item inside the navigation bar as illustrated in figure 4.7.

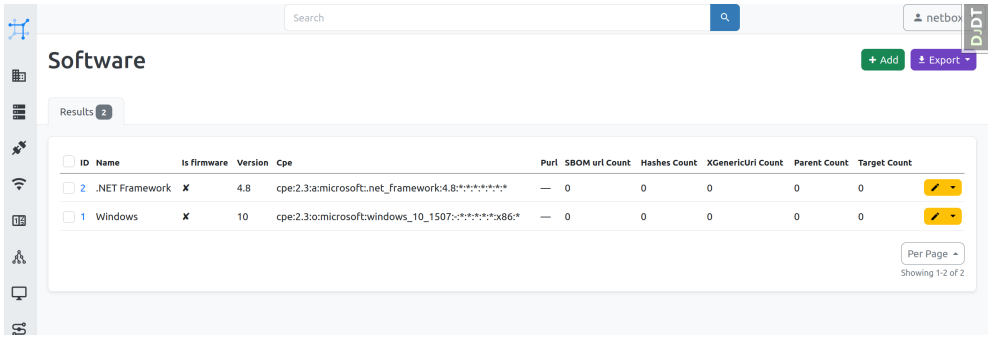

Figure 4.37: Table-View of Software

In figure 4.37, two software objects are displayed. For each object, the corresponding attributes are shown. If an attribute allows only one value, such as `Name`, `Is Firmware`, `Version`, `CPE` and `PURL` the actual values are displayed. For all other attributes, only the count of available values is presented. At the end of each row, the yellow button with the pen icon allows users to navigate to the Edit-View of the software object. Clicking the arrow pointing downward right next to the pen icon provides options to either delete or view the 'Changelog' for the item.

In the top right corner, two buttons are available: One button to add a new Software object through the Add-View, and another button providing standard export functionality for tables in Netbox.

4.5 ProductRelationship

With the addition of a software model described in section 4.4 for an asset database, it is also crucial to model dependencies between devices and software, such as Software 'x' being installed on Device 'y'. Furthermore, software components can, of course, be installed alongside other software components and can have dependencies on each other. The `ProductRelationship` data model allows for the representation of relationships between device and software components. The modeling methodology employed in the DDDC-Plugin is founded on the machine-processable format for security advisories known as the Common Security Advisory Framework (CSAF). Instead of referring to devices and software components, CSAF classifies any deliverable identifiable by a name as a `Product`. The following relationship types between Products are defined by CSAF in [23]:

- `default_component_of`: Indicates that the entity labeled with one Product ID is a default component of an entity with another Product ID.
- `external_component_of`: Indicates that the entity labeled with one Product ID is an external component of an entity with another Product ID.
- `installed_on`: Indicates that the entity labeled with one Product ID is installed on a platform entity with another Product ID .
- `installed_with`: Indicates that the entity labeled with one Product ID is installed alongside an entity with another Product ID.

- `optional_component_of`: Indicates that the entity labeled with one Product ID is an optional component of an entity with another Product ID.

In the context of the Netbox data model, 'Products' are typically either Devices or Software. Therefore, such a relationship can only be modeled between these two data models using `ProductRelationships`. In the following subsections, the Add, Detail, Table, and Edit-View for `ProductRelationships` are described.

4.5.1 Add, Edit and Detail-View

The Add-View is utilized to create a new `ProductRelationship` via the user interface. The user can access the Add-View through one of the following three options:

- Append the following url: `/plugins/d3c/productrelationship/add/`
- Click on the green plus symbol located right beside the `ProductRelationship`-Item in the navigation bar, illustrated in figure 4.38.
- The `ProductRelationship` Table-View, as described in section 4.5.2, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.39.

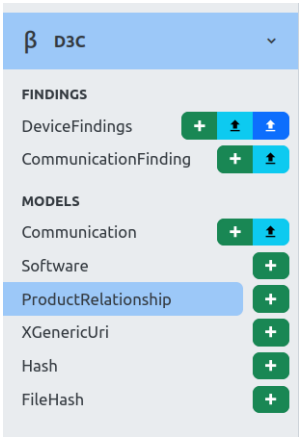

Figure 4.38: Navigation bar including the `ProductRelationship` item, which allows users to navigate to the Table-View, described in section 4.5.2. Right next to the item, the green button with the plus icon enables the user to navigate to the Add-View.

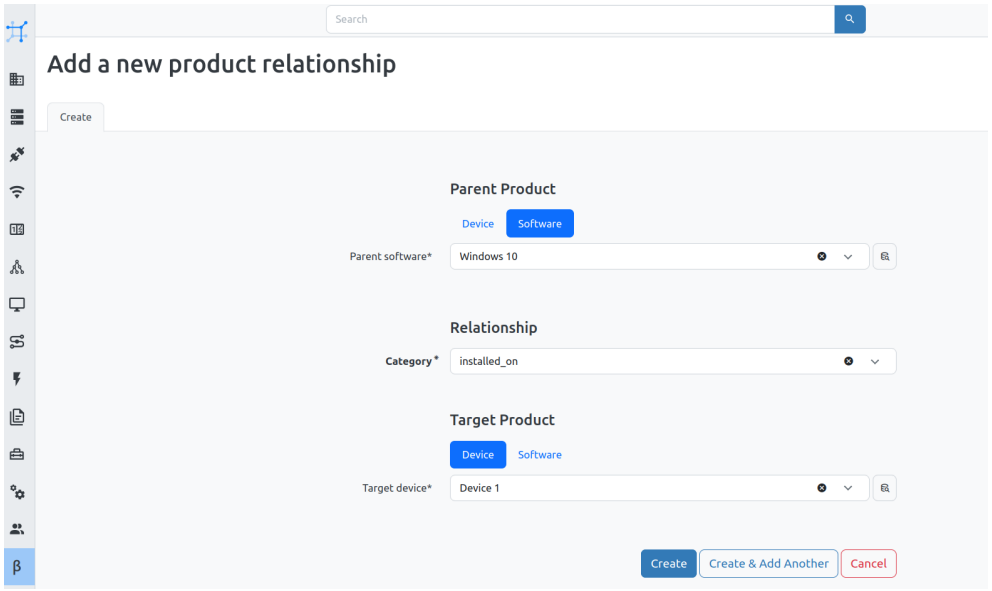

Figure 4.39: Add-View of `ProductRelationship`

The Add-View in figure 4.39 enables the definition of relationships between Device and Software objects that already exist in the Netbox database. The user should first select the parent product, which can be either a device or software object. Afterwards, the category of the relationship can be specified. This is a choice set with the 5 defined product relationship types defined by CSAF:

- `default_component_of`
- `external_component_of`
- `installed_on`
- `installed_with`
- `optional_component_of`

At last, the user can specify the target product, which can be again either a device or software object. All fields in this form are mandatory, so the creation of the relationship will fail if one of the input fields is not set.

Warning: Please note that only one parent product and only one target product can be assigned. If the user specifies, for example, both a parent device and a parent software object, the creation will fail with the message 'A ProductRelationship can only be assigned to a single parent object'.

After specifying the values for the `ProductRelationship`, it can be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.39. The creation can also be canceled using the corresponding button. If the creation is successful, the user will be navigated to the detail view of the `ProductRelationship`, as illustrated in figure 4.40.

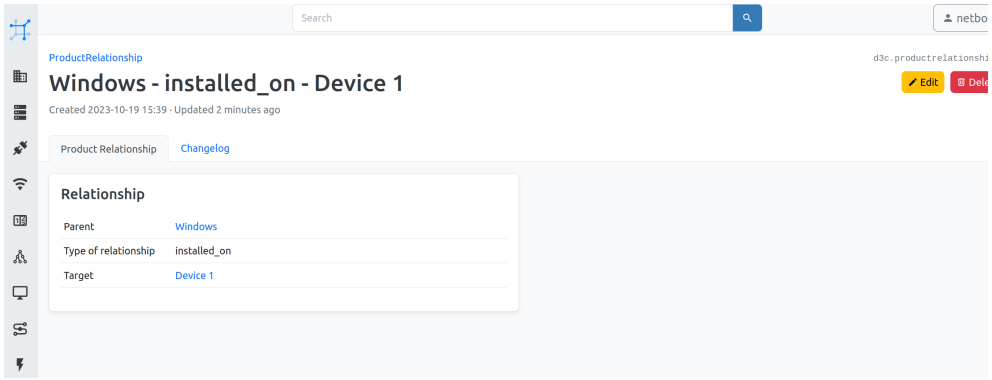

Figure 4.40: Detail-View of `ProductRelationship`

`ProductRelationships` are also displayed within the corresponding Detail-Views of the Device and Software models:

- Figure 4.41 shows the Detail-View of a Software object named 'Windows'. In the 'Relationship as parent' field, it can be seen that this software is installed on a Device named 'Device 1'. Furthermore, a Software named '.NET Framework' is installed on the operating system, as shown in the field 'Relationship as target'.
- Figure 4.41 shows the ProductRelationships-Tab within the Detail-View of a Device object named 'Device 1' displaying all `ProductRelationships` where the corresponding device is either the parent or source of a `ProductRelationship`.

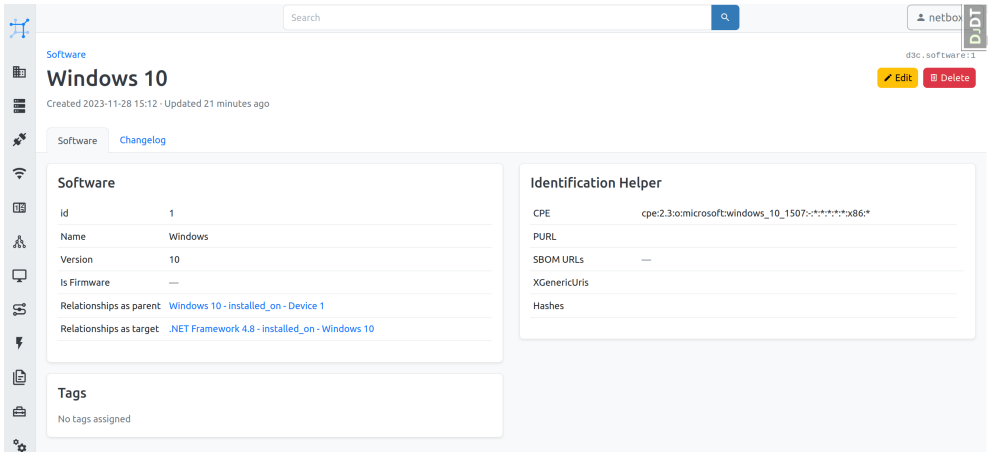

Figure 4.41: Detail-View of Software with `ProductRelationship`

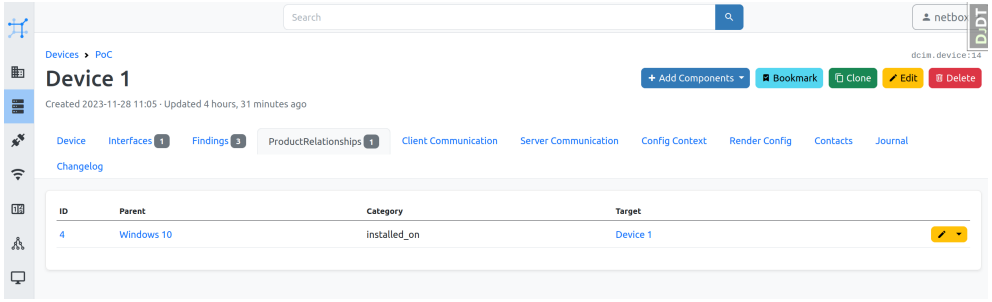

Figure 4.42: ProductRelationships-Tab of a Device listing all `ProductRelationship` for this Device

On the top right corner of the Detail-View in figure 4.40, the following two action buttons are available:

- Edit: A click on this button navigates the user to the edit page, as illustrated in figure 4.43. The Edit-View is identical to the Add-View, with the only difference being the presence of filled-out attributes. Changes can be saved by clicking the 'Save'-button.
- Delete: A click on this button deletes the object after a warning message is displayed to the user.

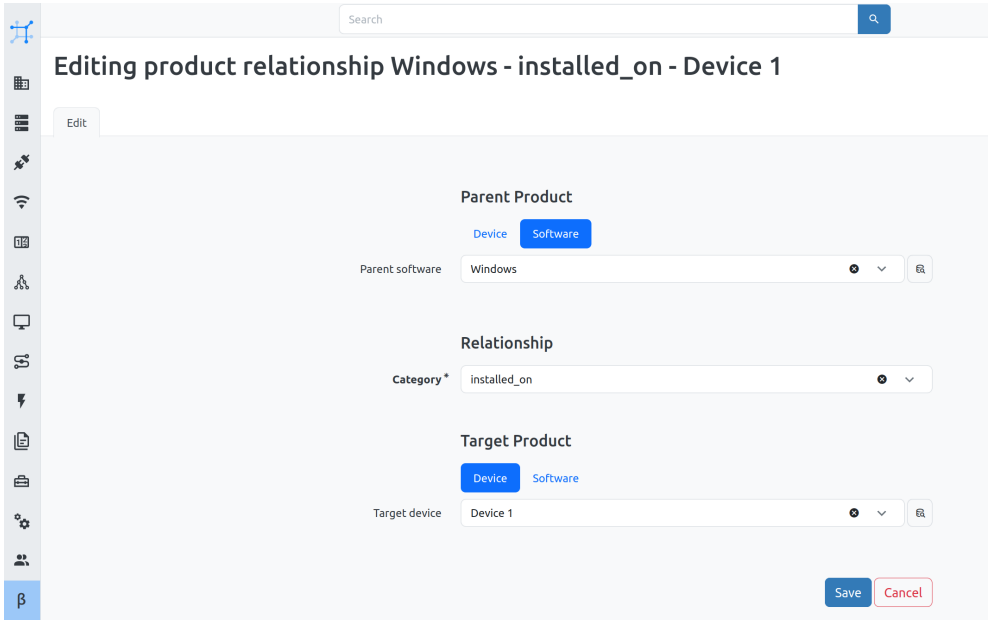

Figure 4.43: Edit-View of `ProductRelationship`

4.5.2 Table-View

The Table-View of the `ProductRelationship` data model is illustrated in figure 4.44. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/productrelationship/`

- Click on the `ProductRelationship`-item inside the navigation bar as illustrated in figure 4.7.

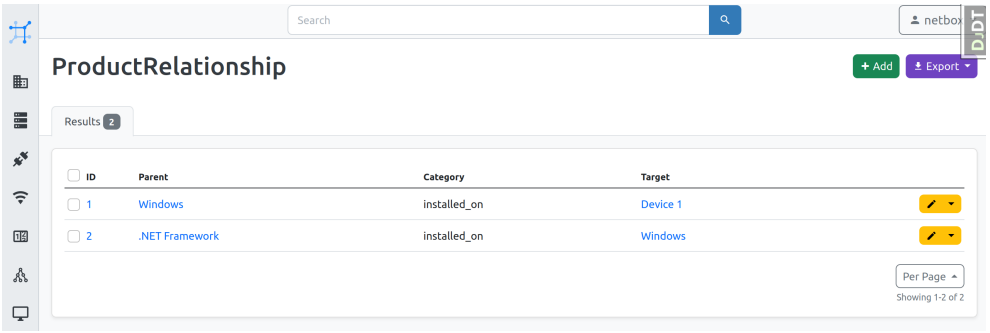

Figure 4.44: Table-View of `ProductRelationships`

In figure 4.44, two `ProductRelationship` objects are displayed. For each object, the corresponding attributes are shown. At the end of each row, the yellow button with the pen icon allows users to navigate to the Edit-View of the `ProductRelationship` object. Clicking the arrow pointing downward right next to the pen icon provides options to either delete or view the 'Changelog' for the item.

Furthermore, in the top right corner, two buttons are available: One button to add a new `ProductRelationship` object through the Add-View, and another button providing standard export functionality for tables in Netbox.

4.6 Generic URIs

A generic URI is an identification helper used in the Common Security Advisory Framework (CSAF), a machine-processable format for security advisories. It consists of the following two attributes, as defined in [23]:

- `namespace`: String with format uri referring to a URL which provides the name and knowledge about the specification used or is the namespace in which these values are valid.
- `uri`: String with format uri containing the identifier itself.

In the context of the data model in Netbox, a generic URI can be assigned to either a Device Type or Software. In the following subsections, the Add, Detail, Table, and Edit-View for `XGenericURIs` are described.

4.6.1 Add, Edit and Detail-View

The Add-View is utilized to create a new `XGenericURI` via the user interface. The user can access the Add-View through one of the following three options:

- Append the following url: `/plugins/d3c/xgenericuri/add/`
- Click on the green plus symbol located right beside the `XGenericURI`-Item in the navigation bar, illustrated in figure 4.45.

- The `XGenericURI` Table-View, as described in section 4.5.2, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.46.

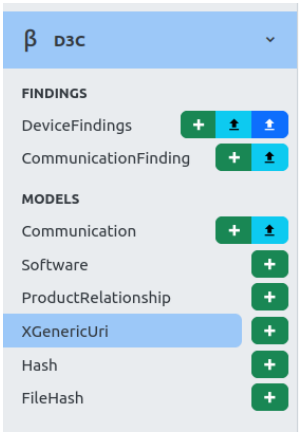

Figure 4.45: Navigation bar including the `XGenericURI` item, which allows users to navigate to the Table-View, described in section 4.6.2. Right next to the item, the green button with the plus icon enables the user to navigate to the Add-View.

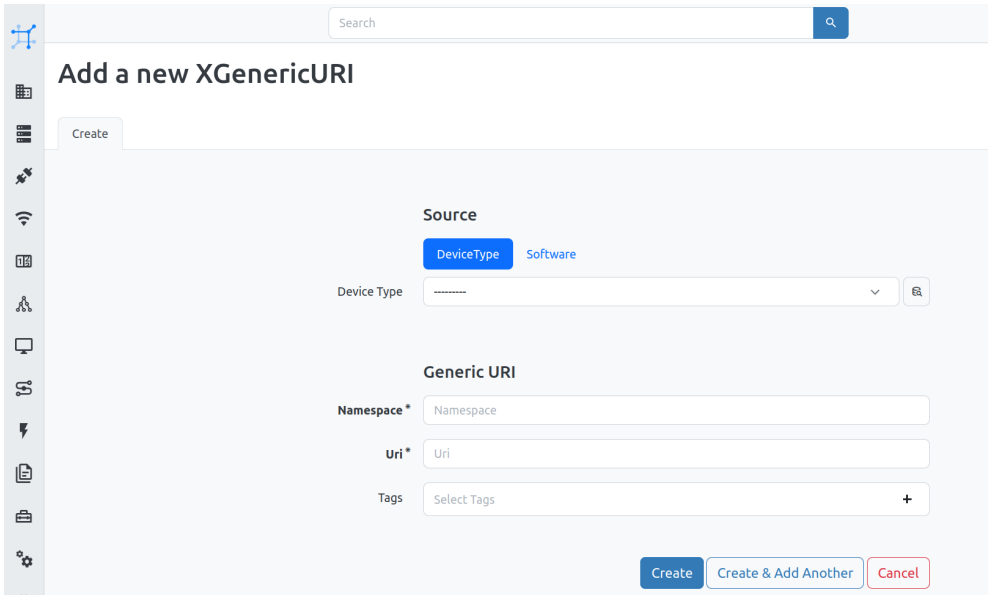

Figure 4.46: Add-View of `XGenericURI`

The Add-View in figure 4.39 allows the user to define a `XGenericURI` for a Device Type or a Software object. First, the user selects the object for which the `XGenericURI` is associated. Then, the user specifies the namespace and URI strings in the text fields of the input form. The form validates if the provided strings are valid URIs. Optionally, the user can assign tags. Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`. All fields in this form, except for the tag field, are mandatory. Therefore, the creation of the URI will fail if any of these input fields is left empty. After specifying the values for the `XGenericURI`, it can

be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.46. The creation can also be canceled using the corresponding button. If the creation is successful, the user will be navigated to the detail view of the `XGenericURI`, as illustrated in figure 4.47.

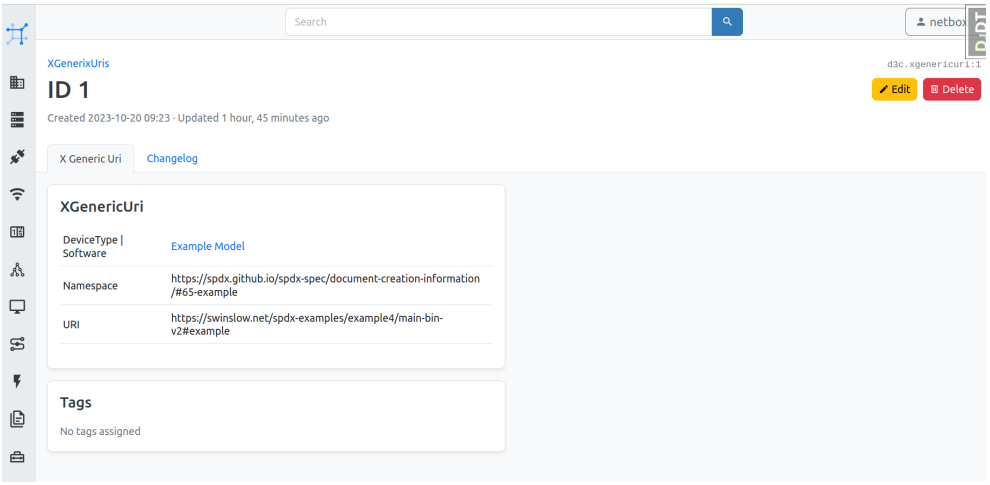

Figure 4.47: Detail-View of `XGenericURI`

`XGenericURIs` are also displayed within the corresponding Detail-Views of the Device Type and Software models:

- Figure 4.48 shows the Detail-View of a Device Type object named 'Example Model'. In the bottom right corner, a new section called 'Identifier' is displayed, listing the associated `XGenericURIs`. In the case of figure 4.47, a single `XGenericURI` with ID 1 is associated with this Device Type.
- The Detail-View of a software is also listing the associated `XGenericURIs` (see figure 4.35).

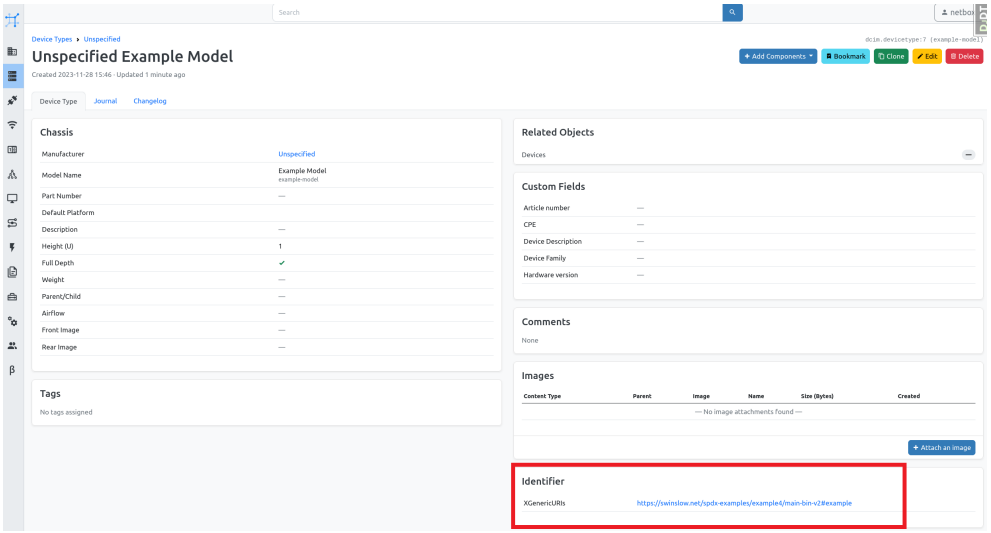

Figure 4.48: Detail-View of a Device Type

On the top right corner of the Detail-View in figure 4.48, the following two action buttons are available:

- Edit: A click on this button navigates the user to the edit page of a `XGenericURI` object, as illustrated in figure 4.49. The Edit-View is identical to the Add-View, with the only difference being the presence of filled-out attributes. Changes can be saved by clicking the 'Save'-button.
- Delete: A click on this button deletes the object after a warning message is displayed to the user.

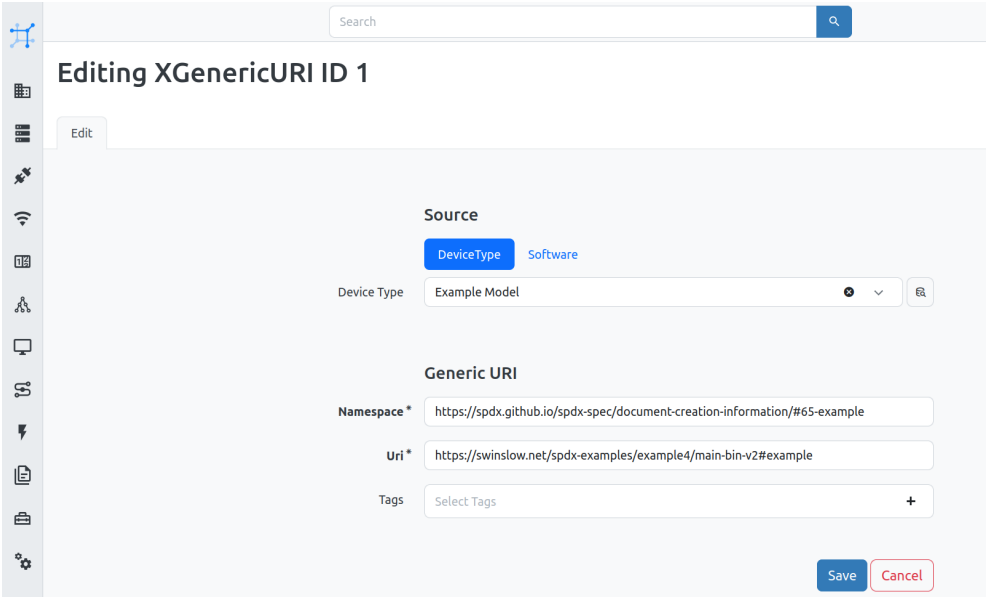

Figure 4.49: Edit-View of `XGenericURI`

4.6.2 Table-View

The Table-View of the `XGenericURI` data model is illustrated in figure 4.50. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/xgenericuri/`
- Click on the `XGenericURI`-item inside the navigation bar as illustrated in figure 4.45.

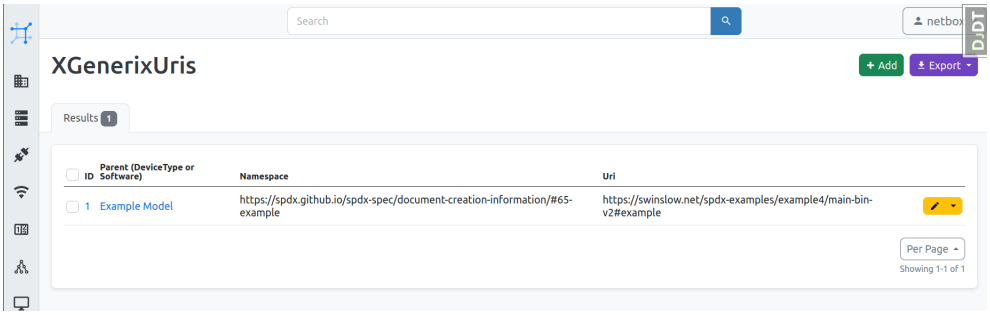

Figure 4.50: Table-View of `XGenericURIs`

In figure 4.44, a single `XGenericURI` objects is displayed. For each object, the corresponding attributes are shown. At the end of each row, the yellow button with the pen icon allows users to navigate to the Edit-View of the `XGenericURI` object. Clicking the arrow pointing downward right next to the pen icon provides options to either delete or view the 'Changelog' for the item. Furthermore, in the top right corner, two buttons are available: One button to add a new `XGenericURI` object through the Add-View, and another button providing standard export functionality for tables in Netbox.

4.7 Hash

A Hash is an identification helper used in the Common Security Advisory Framework (CSAF), a machine-processable format for security advisories. It consists of the following two attributes, as defined in [23]:

- `filename`: String with one or more characters containing the name of the file which is identified by the hash values
- `file_hashes`: List of hashes holding at least one item containing a list of cryptographic hashes usable to identify files.

To model this identification helper in Netbox, the DDDC-Plugin introduces the two data models: `Hash` and `FileHash`. `FileHashes` are described in section 4.8. Each `FileHash` must be assigned to one `Hash`, allowing the addition of multiple `FileHashes` to a single `Hash`. In the following subsections, the Add, Detail, Table, and Edit-View for `Hashes` are described.

4.7.1 Add, Edit and Detail-View

The Add-View is utilized to create a new `Hash` via the user interface. The user can access the Add-View through one of the following three options:

- Append the following url: `/plugins/d3c/hash/add/`
- Click on the green plus symbol located right beside the `Hash`-Item in the navigation bar, illustrated in figure 4.51.
- The `Hash` Table-View, as described in section 4.7.2, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.52.

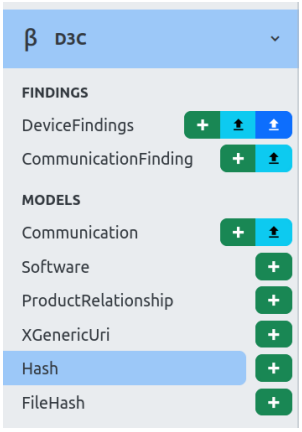

Figure 4.51: Navigation bar including the `Hash` item, which allows users to navigate to the Table-View, described in section 4.7.2. Right next to the item, the green button with the plus icon enables the user to navigate to the Add-View.

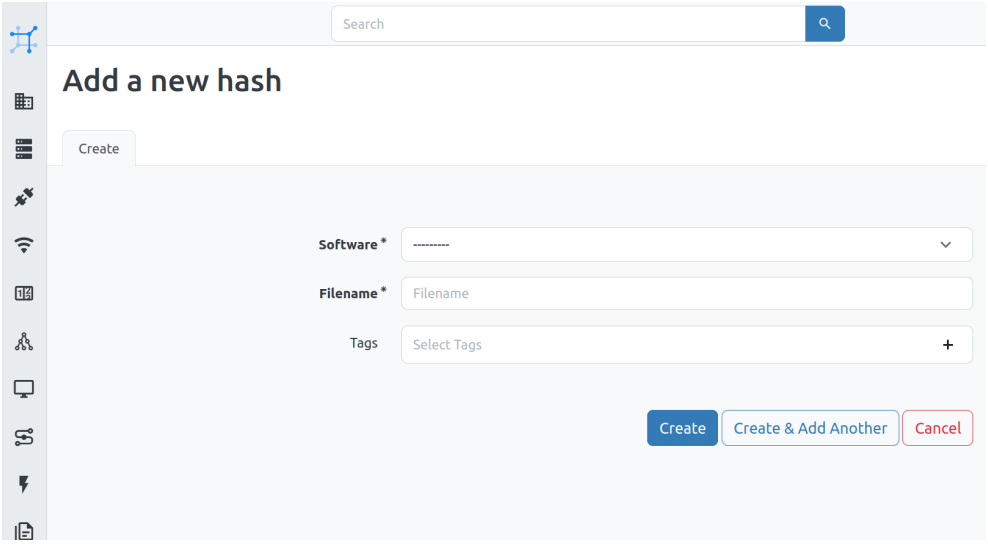

Figure 4.52: Add-View of `Hash`

The Add-View in figure 4.52 allows the user to define a `Hash`. First, the user selects the Software object for which the `Hash` belongs. Then, the user specifies the filename in the text field of the input form. Optionally, the user can assign tags. Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`. All fields in this form, except for the tag field, are mandatory. Therefore, the creation of the `Hash` will fail if any of these input fields is left empty. After specifying the values for the `Hash`, it can be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.52. The creation can also be canceled using the corresponding button. If the creation is successful, the user will be navigated to the Detail-View of the `Hash`, as illustrated in figure 4.53. In this figure, a software called 'Word' is displayed with 'WINWORD.EXE' as the filename. No file hashes are specified yet. To do this, the user has to add a `FileHash` via the Add-View for `FileHashes`, as described in section 4.8. `Hashes` are also displayed within

the corresponding Detail-Views of Software model. In figure 4.54, the `Hash` can be identified in the 'Hashes' field within the 'Identification Helper' card.

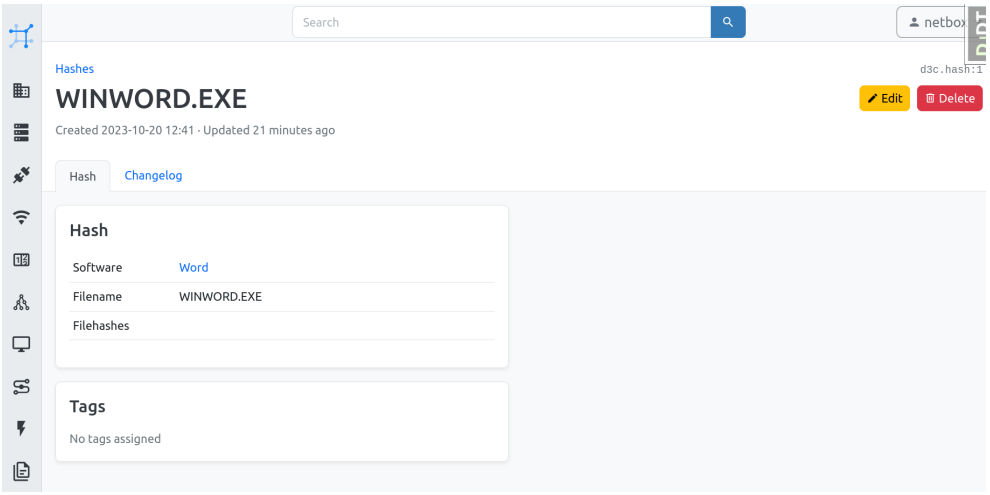

Figure 4.53: Detail-View of `Hash`

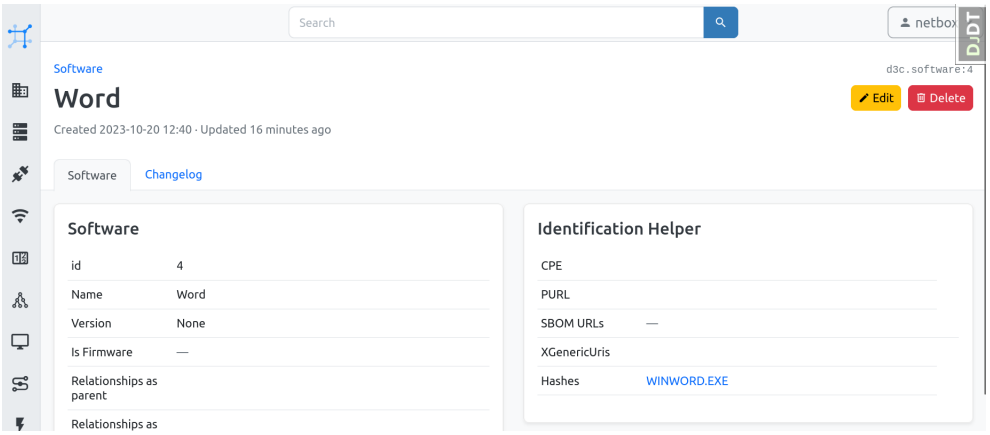

Figure 4.54: Detail-View of a `Hash`

On the top right corner of the Detail-View in figure 4.53, the following two action buttons are available:

- Edit: A click on this button navigates the user to the edit page of a `Hash` object, as illustrated in figure 4.55. The Edit-View is identical to the Add-View, with the only difference being the presence of filled-out attributes. Changes can be saved by clicking the 'Save'-button.
- Delete: A click on this button deletes the object after a warning message is displayed to the user.

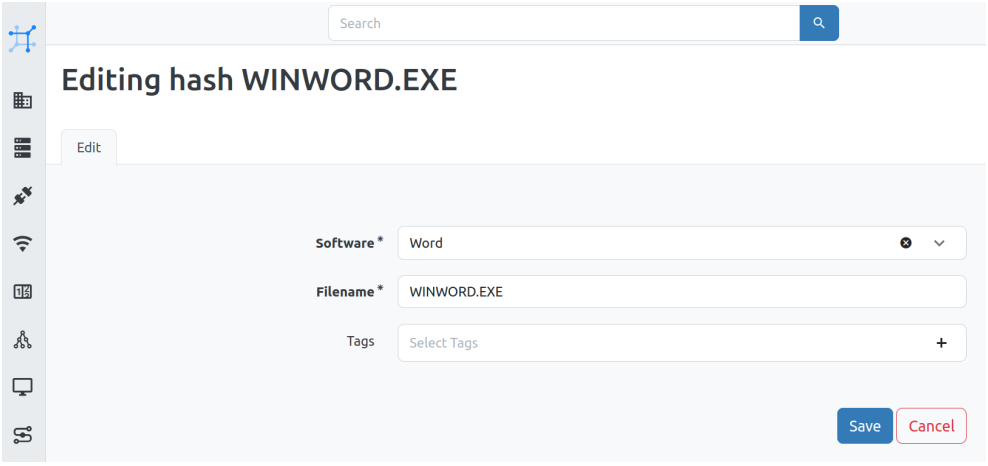

Figure 4.55: Edit-View of `Hash`

4.7.2 Table-View

The Table-View of the `Hash` data model is illustrated in figure 4.56. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/hash/`
- Click on the `Hash`-item inside the navigation bar as illustrated in figure 4.51.

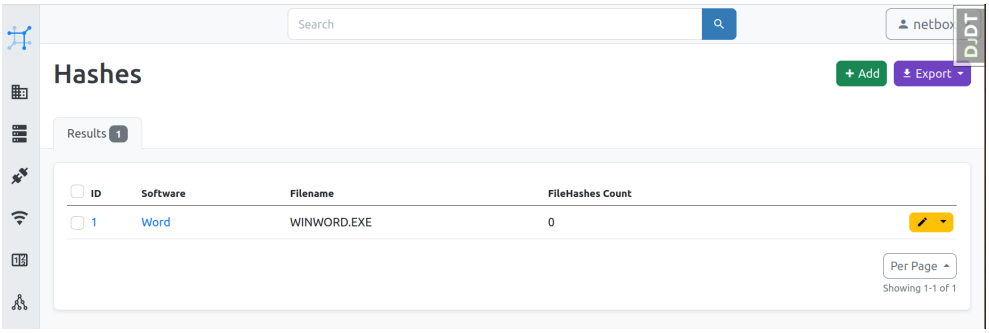

Figure 4.56: Table-View of `Hashes`

In figure 4.56, a single `Hash` objects is displayed. For each object, the corresponding attributes are shown. A counter for all associated `FileHashes` is also provided. At the end of each row, the yellow button with the pen icon allows users to navigate to the Edit-View of the `Hash` object. Clicking the arrow pointing downward right next to the pen icon provides options to either delete or view the 'Changelog' for the item. Furthermore, in the top right corner, two buttons are available: One button to add a new `Hash` object through the Add-View, and another button providing standard export functionality for tables in Netbox.

4.8 FileHash

`FileHashes` can be assigned to one `Hash`. As described in the previous section, a Hash is an identification helper used in the Common Security Advisory Framework

(CSAF) consisting of the following two attributes, as defined in [23]:

- `filename`: String with one or more characters containing the name of the file which is identified by the hash values
- `file_hashes`: List of hashes holding at least one item containing a list of cryptographic hashes usable to identify files.

To model this identification helper in Netbox, the DDDC-Plugin introduces the two data models: `Hash` and `FileHash`. `Hashes` are described in previous section 4.7. Each `FileHash` must be assigned to one `Hash`. In the following subsections, the Add, Detail, Table, and Edit-View for `FileHashes` are described.

4.8.1 Add, Edit and Detail-View

The Add-View is utilized to create a new `FileHash` via the user interface. The user can access the Add-View through one of the following three options:

- Append the following url: `/plugins/d3c/filehash/add/`
- Click on the green plus symbol located right beside the `FileHash`-Item in the navigation bar, illustrated in figure 4.57.
- The `FileHash` Table-View, as described in section 4.8.2, features a green action button labeled 'Add' that allows the user to navigate to the view depicted in figure 4.58.

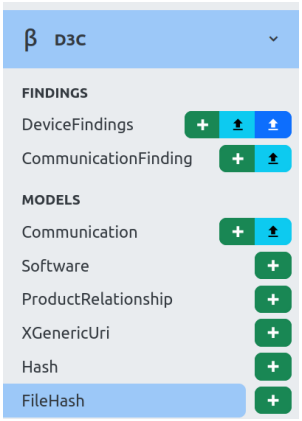

Figure 4.57: Navigation bar including the `FileHash` item, which allows users to navigate to the Table-View, described in section 4.8.2. Right next to the item, the green button with the plus icon enables the user to navigate to the Add-View.

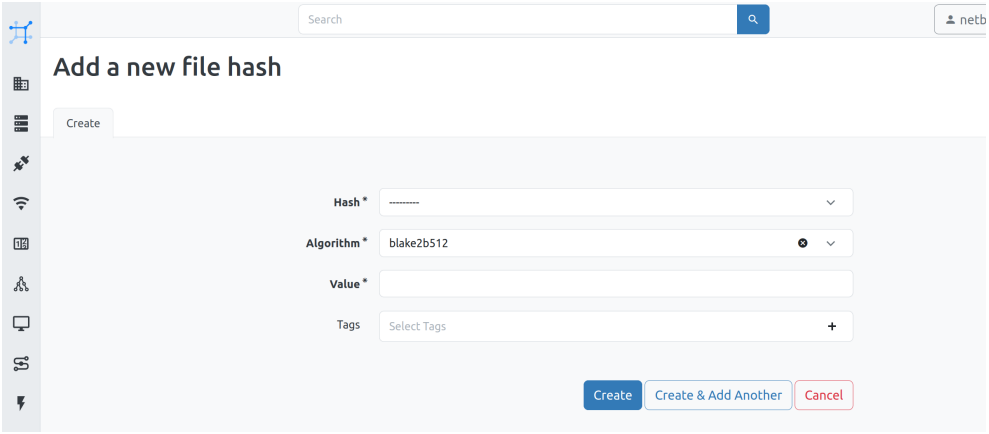

Figure 4.58: Add-View of `FileHash`

The Add-View in figure 4.58 allows the user to define a `FileHash`. First, the user selects the Hash object for which the `FileHash` belongs. Then, the user specifies the algorithm and value in the text fields of the input form. The algorithm has to be specified from a predefined set and the value has to be a hexadecimal value. The predefined set of algorithms can be customized by the user through editing the "d3c_filehash_algo" choice set which is accessible through the the navigation bar under "Customization" and "Custom Field Choices", as illustrated in figure 4.59. Optionally, the user can assign tags. Tags are user-defined labels as defined by Netbox in [17] and can be accessed via the url: `/extras/tags`. All fields in this form, except for the tag field, are mandatory. Therefore, the creation of the `FileHash` will fail if any of these input fields is left empty. After specifying the values for the `FileHash`, it can be created using one of the two buttons, either *Create* or *Create & Add Another*, as illustrated in figure 4.58. The creation can also be canceled using the corresponding button. If the creation is successful, the user will be directed to the Detail View of the `FileHash`, as illustrated in figure 4.60. In this figure, a file hash for the filename 'WINWORD.EXE' is shown, with a specified sha256 hash.

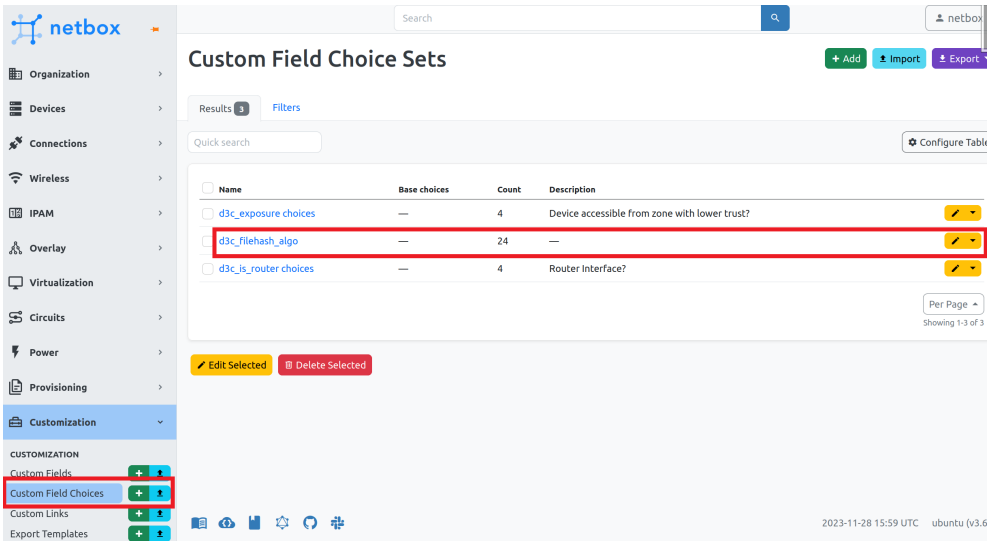

Figure 4.59: Predefined algorithms for `FileHash`

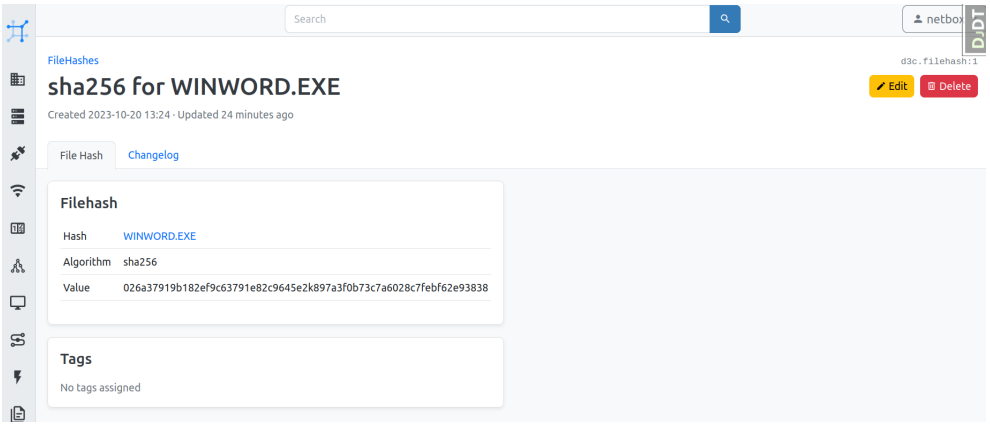

Figure 4.60: Detail-View of `FileHash`

`FileHashes` are also displayed within the corresponding Detail-Views of the `Hash` model. In figure 4.61, the `FileHash` can be identified in the 'Filehashes' field.

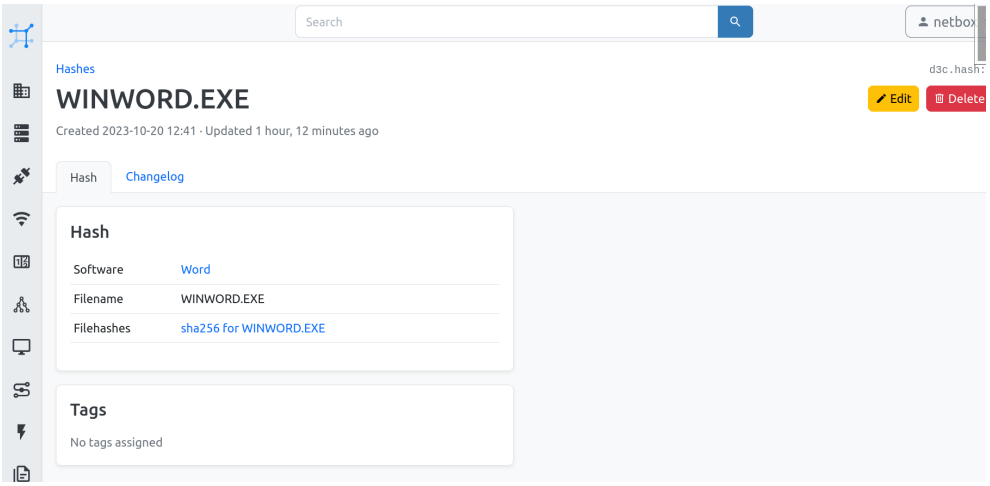

Figure 4.61: Detail-View of a `FileHash`

On the top right corner of the Detail-View in figure 4.53, the following two action buttons are available:

- Edit: A click on this button navigates the user to the edit page of a `FileHash` object, as illustrated in figure 4.62.The Edit-View is identical to the Add-View, with the only difference being the presence of filled-out attributes. Changes can be saved by clicking the 'Save'-button.
- Delete: A click on this button deletes the object after a warning message is displayed to the user.

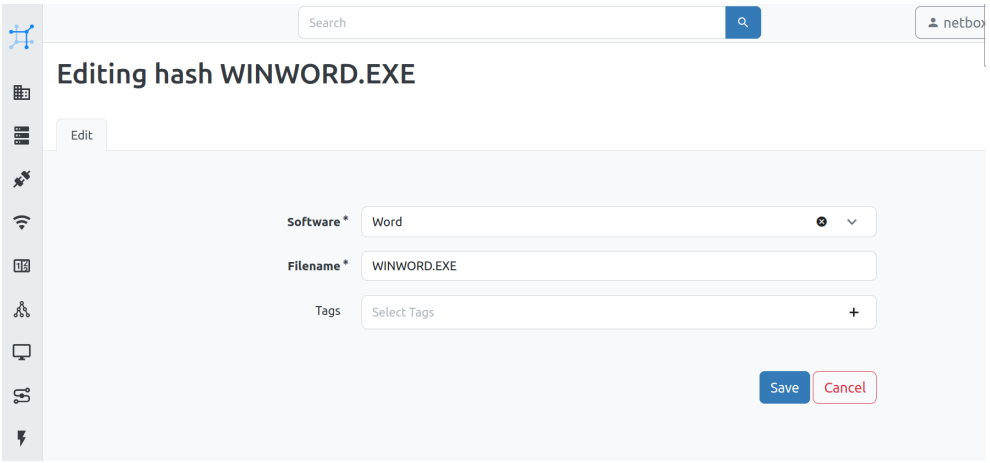

Figure 4.62: Edit-View of `FileHash`

4.8.2 Table-View

The Table-View of the `FileHash` data model is illustrated in figure 4.63. The user can access the Table-View through one of the following two options:

- Append the following url: `plugins/d3c/filehash/`
- Click on the `FileHash` item inside the navigation bar as illustrated in figure 4.57.

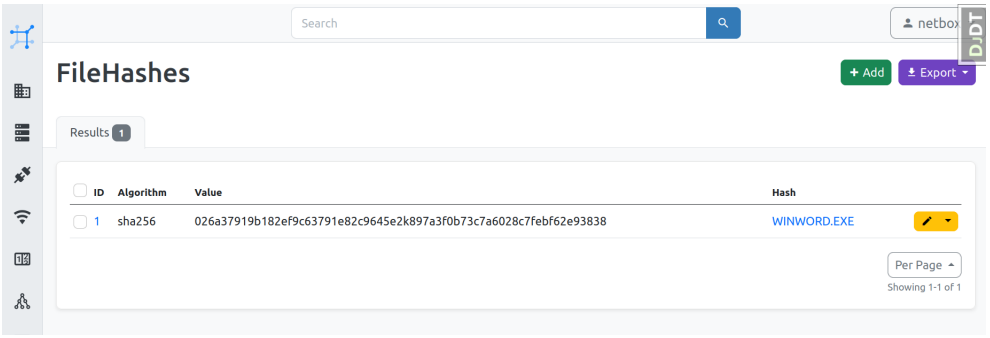

Figure 4.63: Table-View of `Hashes`

In figure 4.63, a single `FileHash` objects is displayed. For each object, the corresponding attributes are shown. At the end of each row, the yellow button with the pen icon allows users to navigate to the Edit-View of the `Hash` object. Clicking the arrow pointing downward right next to the pen icon provides options to either delete or view the 'Changelog' for the item. Furthermore, in the top right corner, two buttons are available: One button to add a new `FileHash` object through the Add-View, and another button providing standard export functionality for tables in Netbox.

4.9 Apply DeviceFindings on a Device

This section describes how `DeviceFindings` can be used to modify properties associated with a device. In general, the user has two options:

1. The user can inspect a single finding and compare this finding to the current device properties.
2. The user can inspect multiple findings at once and compare the findings to the current device properties.

In the next subsection 4.9.1, the first approach will be outlined, and in the following subsection 4.9.2, the approach for multiple findings will be described. Understanding how the DDDC plugin works with a single finding helps with the process of considering multiple findings. For both approaches, the Findings-Tab within the Detail-View of a device serves as the starting point, as illustrated in figure 4.64.

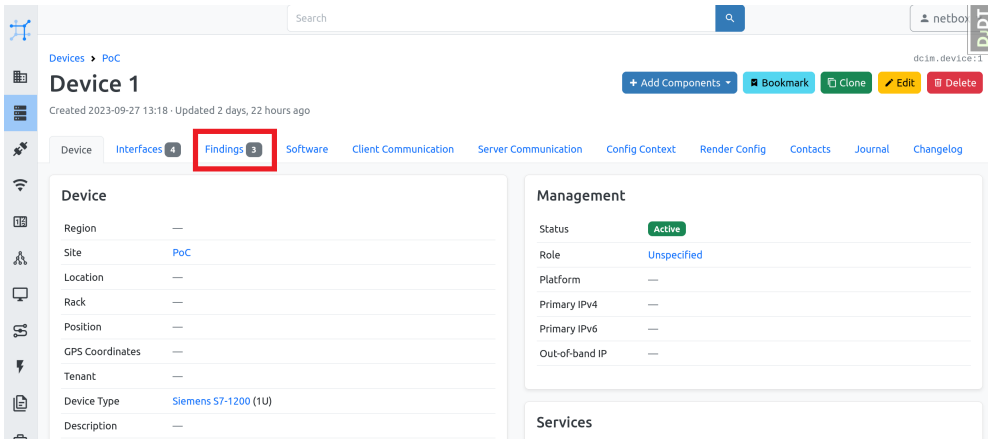

Figure 4.64: Detail-View of a Device.In the rectangular box, the Findings Tab, which is described in this section, is highlighted.

4.9.1 Apply Findings - Single

As a starting point, please refer to figure 4.65. This figure displays the Findings-Tab for a Device object named 'Device 1'. Currently, three different `DeviceFindings` are mapped to this Device, with the IDs 4, 7, and 8. At the end of each row, a blue button with an arrow is provided. Clicking on such a button navigates the user to an input form that enables the comparison of the single `DeviceFinding` with the current set of device properties.

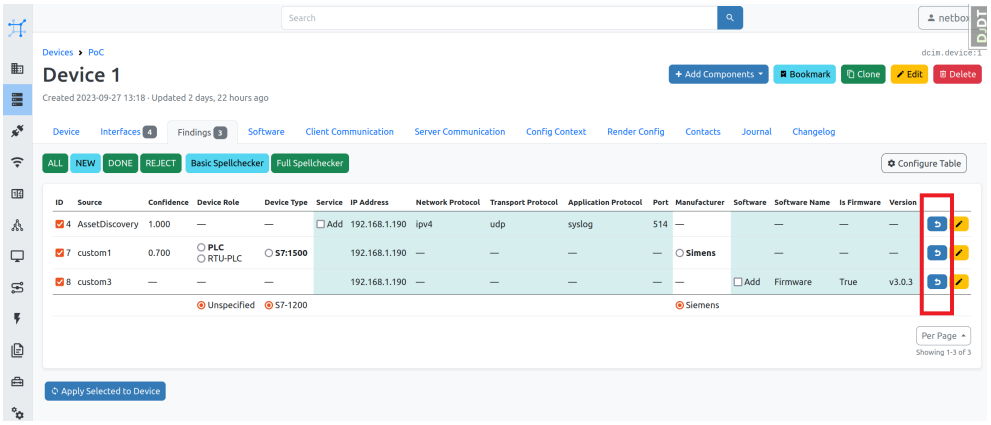

Figure 4.65: Findings Tab in the Detail-View of a device. Three `DeviceFindings` are displayed.

For example, figure 4.66 illustrates the input form that appears when clicking on the blue button for the `DeviceFinding` with ID 7. This form simply allows the user to change the current properties specified for the device to match the values specified in the device findings. Referring to figure 4.65, the device is currently categorized as 'S7-1200' with no assigned device role (it is set to 'unspecified'). In contrast, the device finding suggests that the device is of type 'S7-1500' and should have the role of 'RTU-PLC.' The confidence of the finding is also displayed to the user in the tab. In this figure, for example, the confidence value is set at 0.7. If the user is confident that one of the suggested values by the finding is correct, they can simply select the 'S7-1500' and/or 'RTU-PLC' buttons and click 'Apply'. **Warning:** Clicking on the 'Apply' button will set the `DeviceFinding` to the status `DONE`. This finding will no longer be seen in the Findings-Tab of the Device under the category `NEW`.

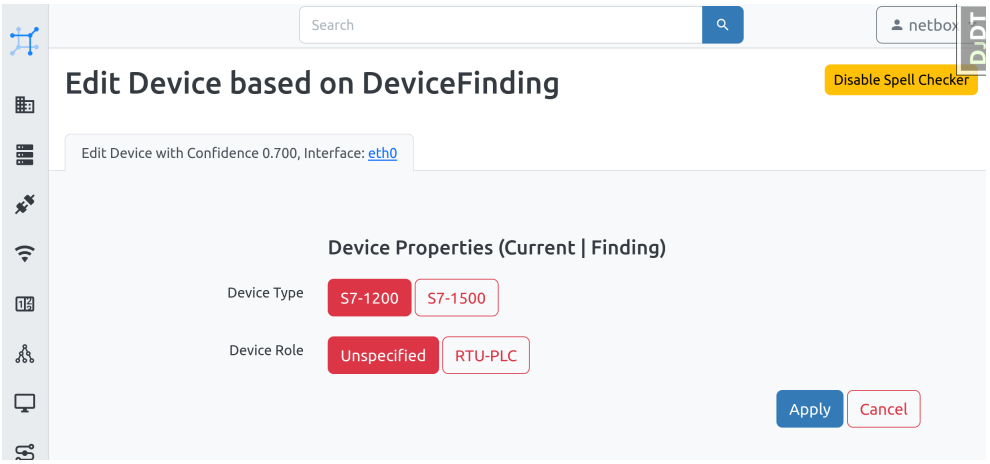

Figure 4.66: Input Form that compares the properties specified by a single `DeviceFinding` with the current values specified by the device (spell checker enabled).

4.9.1.1 Spell checker and normalization

The input form in figure 4.66, used to compare the properties of a finding to the current device values, has a string checker and normalization module activated by default. The user can deactivate this module by clicking on the yellow button labeled 'Disable Spell Checker' in the top right corner of figure 4.66. Clicking on this button will lead to the view illustrated in figure 4.67.

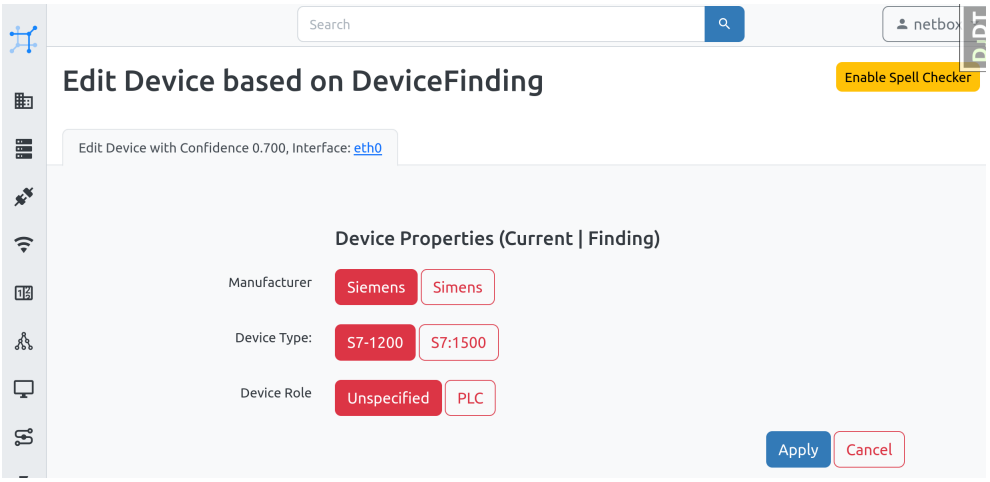

Figure 4.67: Input Form that compares the properties specified by a single `DeviceFinding` with the current values specified by the device (spell checker disabled).

When comparing figure 4.66, and figure 4.67, the following differences can be noticed, corresponding to specific features provided by the spell checker and normalization module:

1. Manufacturer: Figure 4.67 now displays the Manufacturer attribute, which is also specified by this `DeviceFinding` (see figure 4.65). This finding defines 'Simens' as the manufacturer attribute. The spell checker module is capable of correcting this likely misspelling of the manufacturer 'Siemens'. Since 'Siemens' is already the value for the manufacturer of the device, as indicated by the finding, this form simply hides this redundant information from the user.
2. Device Type: The spell checker corrected the likely misspelling of 'S7:1500' to 'S7-1500'.
3. Device Role: The spell checker module normalized the value 'PLC' to the predefined Device Role 'PLC-RTU'.

By design, this form does not allow free-text input. If the spell checker does not provide the correct values, it can be customized by adapting the program code, including the corpus used for providing these corrected values. Alternatively, users can simply use the standard input forms provided by Netbox. Furthermore, the spell checker and normalization module will only be applied to the following `DeviceFinding` attributes, due to the available corpus data: `Manufacturer`, `Device Type`, `Device Role`, `Device Family`, `Article Number` and `Part Number`.

4.9.1.2 Service and Software

Certain attributes defined by the `DeviceFinding` model logically belong together and will be considered as a group when adding this information to a device, specifically, all information related to either service or software:

- For example, consider the `DeviceFinding` with ID 4 in figure 4.68. Clicking on the blue button at the end of the row navigates the user to the view illustrated in figure 4.68. This `DeviceFinding` is a typical finding provided by the Asset Manager, as described in section 3.1. It suggests that a syslog service is running under the IP Address 192.168.1.190, which is used by this Device on Interface 'eth0'. If the user agrees that this service indeed belongs to the device, they can select the checkbox next to 'Add service' and click the 'Apply' button. The service will then be created if it does not already exist. If the user does not believe this service belongs to the device, they can leave the checkbox unselected and then click 'Apply' to change the status of the `DeviceFinding` to 'DONE'.
- For example, consider the `DeviceFinding` with ID 8 in figure 4.68. Clicking on the blue button at the end of the row navigates the user to the view illustrated in figure 4.70. This finding provides information w.r.t. to the firmware installed on this device, called 'Firmware' with version '3.0.3'. If the user agrees that this software indeed belongs to the device, they can select the checkbox next to 'Add software' and click the 'Apply' button. The software will then be created if it does not already exist. If the user does not believe this service belongs to the device, they can leave the checkbox unselected and then click 'Apply' to change the status of the `DeviceFinding` to 'DONE'.

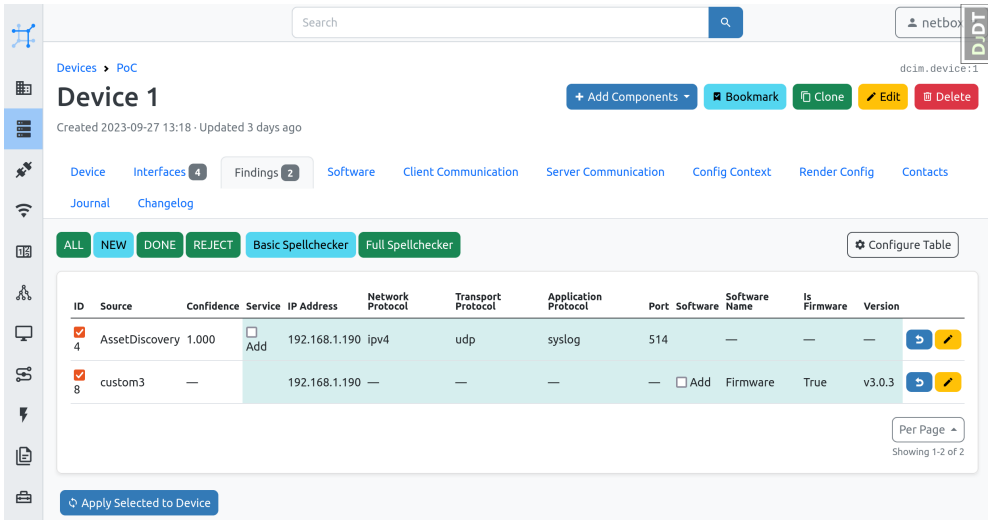

Figure 4.68: Findings Tab in the Detail-View of a device. Two `DeviceFindings` are displayed.

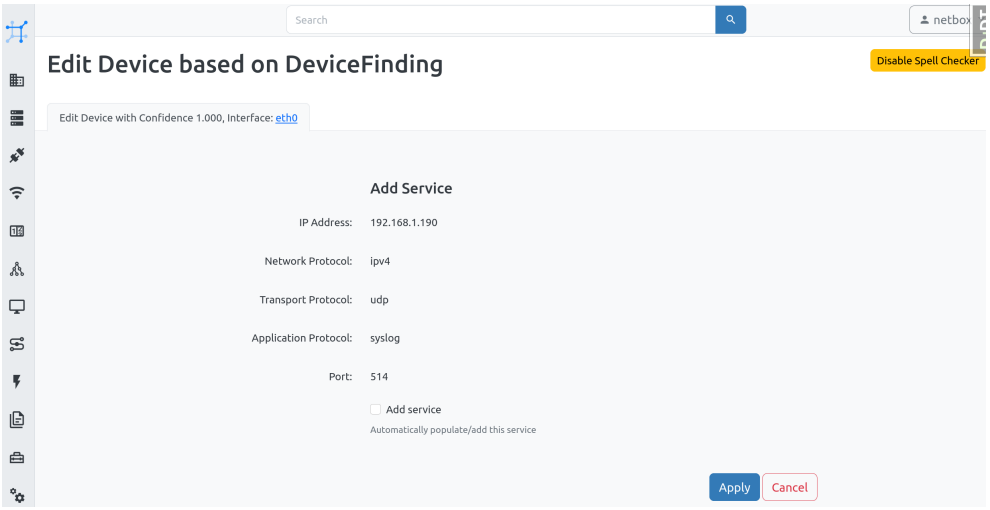

Figure 4.69: Input Form providing properties specified by a single `DeviceFinding` containing information w.r.t. a Service.

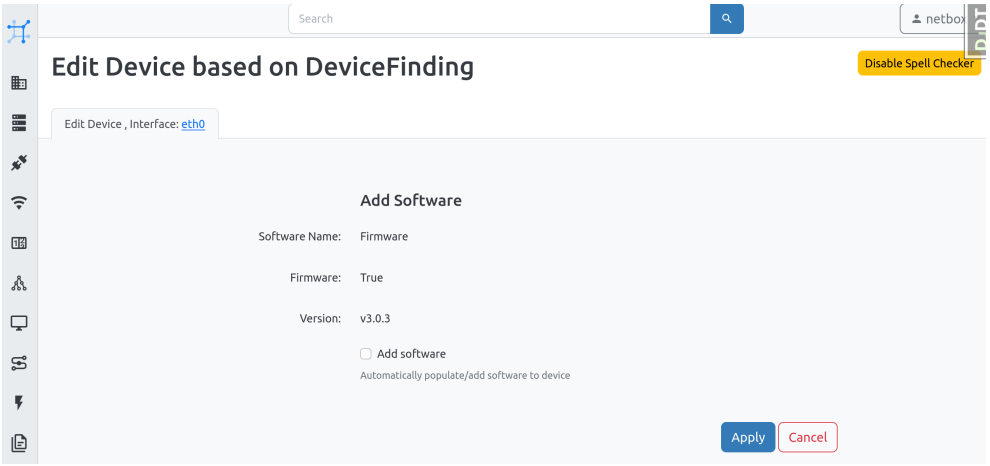

Figure 4.70: Input Form providing properties specified by a single `DeviceFinding` containing information w.r.t. a Software.

4.9.2 Apply Findings - All

The Findings Tab in the Detail-View of a device (figure 4.65) can also be used directly to apply `DeviceFinding` results to a device. The table contains the same data as the form for applying a single `DeviceFinding` (figure 4.66), but the form is displayed horizontally for each `DeviceFinding`, and for all `DeviceFindings` of a device at the same time. All actions normally done one finding at a time can thus be applied in one go.

For most fields, only one value in a column can be selected, and thus the selectable items in these columns have radio buttons to make this selection. For these columns, the current value assigned to the Device is listed in the footer of the column, with a radio button that is selected by default. If the spell checker has alternative suggestions for a value, these alternatives are listed below the original value of the `DeviceFinding`.

The original value of the `DeviceFinding` is displayed in bold. With the buttons *Basic Spellchecker* and *Full Spellchecker* the desired spell checker can be selected. When clicking the *Apply Selected to Device* button, the current values of the Device are replaced with the selected value in each column.

For the columns related to services or software, no radio buttons are used, since a `DeviceFinding` can have multiple services and/or software items. These columns are grouped and highlighted using a different background colour, and a column is added in front containing an *Add* checkbox. When clicking the *Apply Selected to Device* button, each Service or Sofware item that has its *Add* checkbox checked, is added to the Device. By default, all checkboxes in the *ID* column are checked, and thus all `DeviceFindings` are set to *Done*.

When first opening the view, only *NEW* `DeviceFindings` are shown. With the first four buttons above the table the selection can be changed to:

- *ALL* - Show all `DeviceFindings`
- *NEW* - Show only new `DeviceFindings`
- *DONE* - Show only `DeviceFindings` that are done;

Finally, each `DeviceFinding` has a checkbox in the *ID* column. These checkboxes determine which `DeviceFindings` are set to *Done* when the *Apply Selected to Device* button is pressed.

4.10 Custom Fields and default Device Roles

The DDDC-Plugin adds the additional object attributes to the models Device [6], Device Type [6] and Interface [7]. To achieve this, it leverages the Custom Fields provided by Netbox [20]. The following custom fields have been added, as illustrated in figure 4.71, in the custom fields view:

- `Article Number`: Text fields added to the Device Type object. Specifies the stock keeping unit (SKU). It can be the same as model number (NetBox: part_number), especially when seller is the vendor itself.
- `CPE`: Text fields added to the Device Type object. Specifies the Common Platform Enumeration (CPE) string of the device type.
- `Device Description`: Text fields added to the Device Type object. Intended as an additional reminder alongside the device type name (e.g. CPU 414-3 PN/DP Zentralbaugruppe mit: Arbeitsspeicher 4 MB ...).
- `Device Family`: Text fields added to the Device Type object. Specifies the family of a device type (e.g. `SIMATIC, SCALANCE`).
- `Exposure`: Selection added to the Device Type object. Specifies the grade of exposure to other networks. Valid values are (as illustrated in figure 4.72):
  - `Small`: The asset is in a highly isolated and controlled zone. There are no connections from this cyber asset's zone to or from a zone with lower trust.

  - `Indirect`: The asset has no direct access to a zone with lower trust, but other cyber assets in this cyber asset's zone are accessible to or from a zone with lower trust.
  - `Direct`: The asset is directly accessible to or from a zone with lower trust.
  - `Unknown`: Value if category for exposure is unknown.
- `Hardware Version`: Text fields added to the Device Type object. Specifies the hardware version of the device type
- `Is Router`: Selection added to the Interface object. Specifies whether one of the device's interfaces is a router interface or not. Valid values are (as illustrated in figure 4.73):
  - `Yes`: Yes, if it is a router interface.
  - `No`: No, if it is not a router interface.
  - `Maybe`: Maybe, if it might be a router interface.
  - `Unknown`: Value if category is unknown.

  specifies whether one of the device's interfaces is a router interface or not.
- `Safety`: Boolean field added to the Device object. Specifies, if the device is used/provides safety functionality.
- `Secondary Roles`: Multiple objects field added to the Device object. It should be possible to assign multiple Device Roles to a device. Therefore, this custom field enables the user to designate multiple Device Roles for a device using this feature. Additionally, the existing 'Role' attribute of a device should be understood as the primary role of the device.

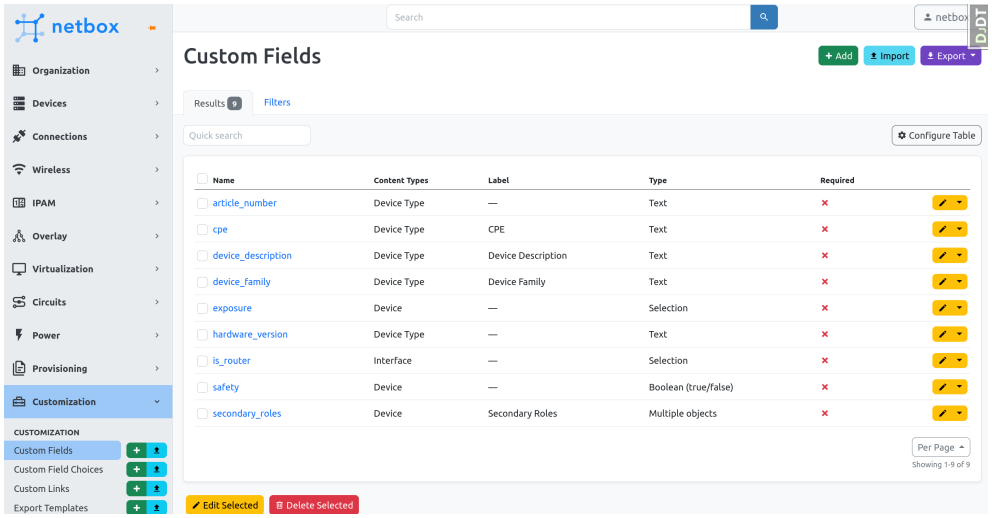

Figure 4.71: Custom Fields added by the D3C-Plugin.

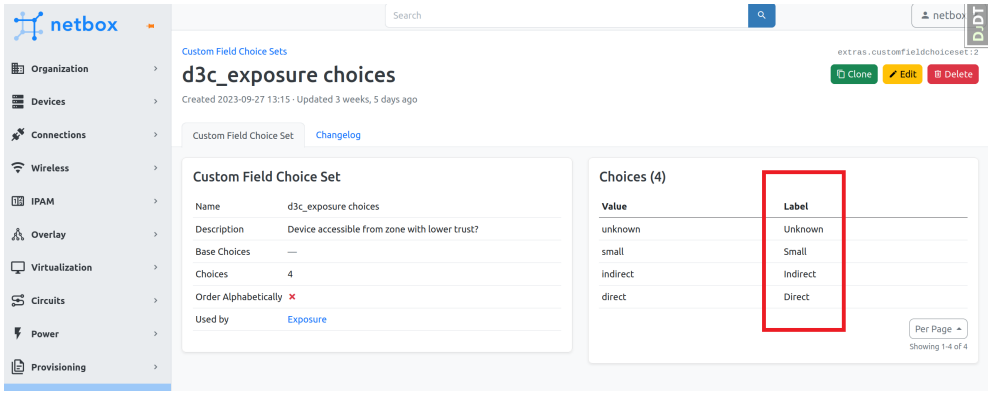

Figure 4.72: Custom Field Choice Set for the Custom Field `Exposure`

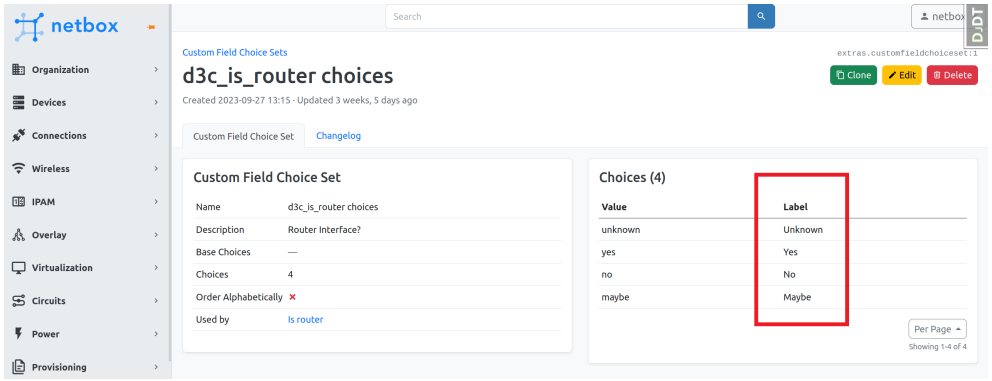

Figure 4.73: Custom Field Choice Set for the Custom Field `Is Router`

With respect to the previously described custom field `Secondary Roles`, multiple Device Roles are introduced by the installation of the DDDC-plugin. The following roles have been defined and are also illustrated in figure 4.74:

- `Active Directory`: A Domain Controller is a server used for central authentication of computers and users in a computer network. Redundant Domain Controllers can also be used to remain operational in case of a failure. In a network with a Domain Controller, multiple computers can be grouped together into a domain. The Domain Controller allows you to specify which users are allowed to log in, to which user groups they belong, and how password rules are enforced. Changes can apply to all computers or users in the domain, or only to subgroups that are members of the domain [5].
- `Actuator`: Actuators are devices that can influence the physical process through standardized signals. Actuators can include pumps, compressors, or control valves, among others [5].
- `EWS`: Engineering workstations are specialized computer systems tailored for programming and managing Programmable Logic Controllers (PLCs).
- `Firewall`: A firewall is a network security device or software that acts as a barrier between a trusted internal network and an untrusted external network, such as the internet. Its primary purpose is to monitor and control incoming

and outgoing network traffic, allowing or blocking data packets based on a set of predetermined security rules or policies.

- `Gateway Coupler`: A communication gateway enables two devices with dissimiliar protocols or transport to communicate. [2]
- `Historian`: Storage for process values. Process values are typically simple physical quantities from sensors, such as temperature and pressure, as time series data, but also control values can be included.
- `HMI Field`: Operator interface in the field, used for operating, monitoring, and, if necessary, for manual intervention in the process.
- `IED`: Intelligent Electronic Device is one or more processors with the capability to receive or send data/control from or to an external source. IED can be polled by an automation process (controller) in the control center or by RTU. [2]
- `MES`: Manufacturing Execution System - Software used in the operations level of the automation pyramid. It serves, among other things, for production planning, resource planning, and maintenance. [5]
- `Router`: Layer 3 specific. An integrated firewall typically does not have the same functionality as a standalone device (primarily packet filtering). Therefore, no distinction is made here.
- `RTU-PLC`: Either a Programmable Logic Controller (PLC) or a Remote Terminal Unit (RTU).
- `SCADA`: A SCADA system may supervise one or more Distributed Control Systems (DCS), or Process Control Systems (PCS) at distant geographic locations [2]. DCS - encompasses both control and operator and monitoring functions. Unlike a PLC (Programmable Logic Controller), visualization is already an integral part of the system here [5].
- `Scada Client`: Distinction from Role `HMI Field`
- `Sensor`: Sensors are devices used to capture physical quantities. They convert these into standardized signals. Sensors can include, among others, pressure measurements, level measurements, or flow measurements [5].
- `Server`: When no category really fits.
- `Switch L2`: Layer 2 Switch.
- `Switch L3`: Layer 3 Switch.
- `Unspecified`: Value if role is unknown.

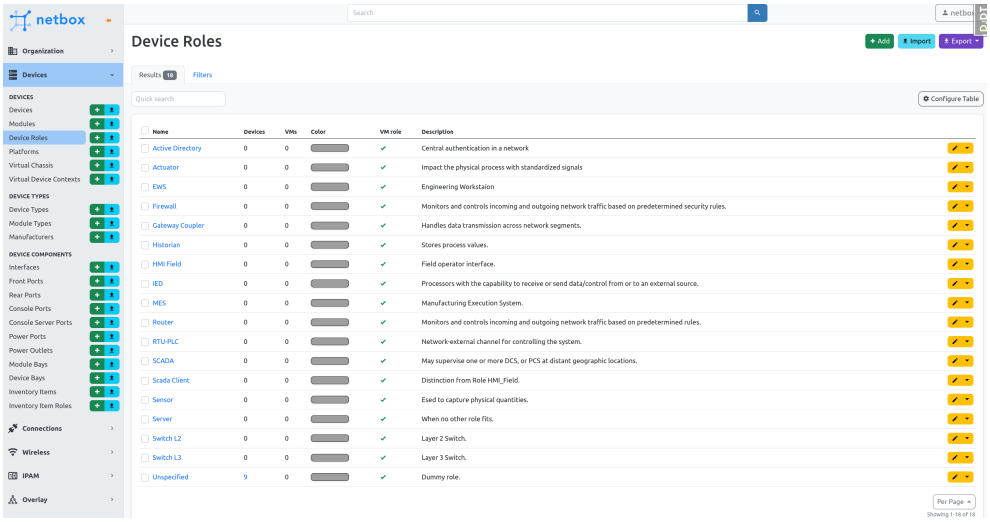

Figure 4.74: Device Roles created via the D3C-Plugin.

4.10.1 DeviceFinding Attribute Association to Devices in NetBox and the Cloning operation

When importing data from various sources, they are initially encapsulated in a DeviceFinding object. The description of the attributes of a `DeviceFinding` can be found in section 4.1.1. After mapping a `DeviceFinding` to a specific device, the user can then map the values of individual attributes of the `DeviceFinding` to attributes of a device, as described in section 4.9. Thus, there is a one-to-one mapping between an attribute of the `DeviceFinding` and an attribute or object in the Netbox database[10]. Table 4.1 illustrates this relationship, whereby relationships between objects are marked with a double colon, and attributes of objects are referenced with a dot. For example, the `Description` attribute of `DeviceFinding` is mapped to the description attribute of the Device Type data model, which is associated with the Device.

Switching a device, the clone operation in Netbox should be mensioned. However, it doesn't perform a deep clone (source: `https://github.com/netbox-community/netbox/issues/33`). When cloning a device, for instance, it's important to note the numerous dependencies it has on other data models, as illustrated in table 4.1. Additionally, during object cloning in Netbox, the values for specified custom fields are not included. As described in section 4.10, this plugin introduces certain custom fields, and users need to be mindful of this absence of included values.

[10]Some attributes, such as `Source` or `Confidence` are not listed in this table because these attributes are not mapped to specific device attributes in Netbox.

DeviceFinding Attribut	Attribute/Object in Netbox
Device Role	Device:Device_Role
Device Name	Device.name
Serial Number	Device.serial
Device Type	Device:Device_Type
Description	Device:Device_Type.description
Device Family	Device:Device_Type.device_family
Manufacturer	Device:Device_Type:Manufacturer
Is Safety Critical	Device.safety
Transport Protocol	Device:Service.protocol
Application Protocol	Device:Service.name
Port	Device:Service.ports
Is Router	Device:Interface.is_router
Status	Device.status
Site	Device:Site
Rack	Device:Rack
Location	Device:Location
Article Number	Device:Interface.mac_address
IP Address	Device:Interface:IP_Addresses
MAC Address	Device:Interface.mac_address
Part Number	Device.part_number
Hardware Version	Device:Device_Type:hardware_version
Hardware CPE	Device:Device_Type.cpe
Software Name	Device:ProductRelationship:Software.name
Is Firmware	Device:ProductRelationship:Software.is_firmware
Version	Device:ProductRelationship:Software.version
Exposure	Device.exposure

Table 4.1: Mapping of DeviceFinding attributes to Netbox data model.

Chapter 5

Summary

This documentation describes the Netbox DDDC plugin from a user's point of view. The plugin enables the building and maintenance of a device database in an OT environment. Such database may serve as the source of truth concerning the devices in the environment and may be helpful in support for various IT security related activities such as network monitoring, patch and vulnerability management, intrusion and threat detection.

The DDDC plugin takes raw input data from several source, including

Device Discovery and Characterization
: The Device Manager implementation (from phase 1 of the project) can be configured to report its findings to the DDC plugin

CSV Data
: Tabular data in CSV format can be imported and mapped to the internal data structure of the finding object

NMAP Data
: XML formatted data from the output of NMAP scripts can be imported and mapped to the internal data structure of the finding object

While the initial processing step involves importing and mapping raw data into so called findings, a subsequent step facilitates associating these findings either with newly created device objects or with pre-existing ones.

Finally, the user reviews device-related findings, determining which to incorporate into the source of truth. This decision-making process is assisted by string processing tools, ensuring consistency in wording.

Bibliography

[1] A. Borcherding, M. Karch, J. Kippe, and F. Specht. AssetDatabase Documentation. https://github.com/DINA-community/ot-assetdatabase, 2023.

[2] Edward Colbert and Alexander Kott, editors. *Cyber-security of SCADA and Other Industrial Control Systems*. Springer Cham, 2016. ISBN 978-3-319-32123-3.

[3] NetBox Community. Populating data. https://docs.netbox.dev/en/stable/getting-started/populating-data/, 2023. Accessed: 2023-11-25.

[4] Django. Models. https://docs.djangoproject.com/en/4.2/topics/db/models/, 2023. Accessed: 2023-10-17.

[5] NAMUR Arbeitskreises 4.18. IT-Grundschutz-Profil „Chemie“. https://www.bsi.bund.de/dok/it-grundschutz-profile, 2023.

[6] Netbox. Devices in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/, 2023. Accessed: 2023-10-17.

[7] Netbox. Interfaces in netbox. https://docs.netbox.dev/en/stable/models/dcim/devicetype/, 2023. Accessed: 2023-10-17.

[8] Netbox. Location in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#location, 2023. Accessed: 2023-10-17.

[9] Netbox. Device name in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#name, 2023. Accessed: 2023-10-17.

[10] Netbox. Part number in netbox. https://docs.netbox.dev/en/stable/models/dcim/devicetype/#part-number, 2023. Accessed: 2023-10-17.

[11] Netbox. Rack in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#rack, 2023. Accessed: 2023-10-17.

[12] Netbox. Device role in netbox. https://docs.netbox.dev/en/stable/models/dcim/devicetype/#device-role, 2023. Accessed: 2023-10-17.

[13] Netbox. Serial number in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#serial-number, 2023. Accessed: 2023-10-17.

[14] Netbox. Service model in netbox. https://docs.netbox.dev/en/stable/models/ipam/service/, 2023. Accessed: 2023-10-17.

[15] Netbox. Site in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#site, 2023. Accessed: 2023-10-17.

[16] Netbox. Status of a device in netbox. https://docs.netbox.dev/en/stable/models/dcim/device/#status, 2023. Accessed: 2023-10-17.

[17] Netbox. Tags in netbox. https://docs.netbox.dev/en/stable/models/extras/tag/, 2023. Accessed: 2023-10-17.

[18] Netbox. Manufacturer in netbox. https://docs.netbox.dev/en/stable/models/dcim/devicetype/#manufacturer, 2023. Accessed: 2023-10-17.

[19] Netbox. Devcetype model in netbox. https://docs.netbox.dev/en/stable/models/dcim/devicetype/#model, 2023. Accessed: 2023-10-17.

[20] Netbox. Custom fields in netbox. https://docs.netbox.dev/en/stable/customization/custom-fields/, 2023. Accessed: 2023-10-17.

[21] Python. Regular expression match objects. https://docs.python.org/3/library/re.html#match-objects, 2023. Accessed: 2023-10-20.

[22] Python. Regular expression syntax. https://docs.python.org/3/library/re.html#regular-expression-syntax, 2023. Accessed: 2023-10-20.

[23] Langley Rock, Stefan Hagen, and Thomas Schmidt. Common security advisory framework version 2.0. https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html, 2023. Accessed: 2023-10-17.

[24] Wikipedia. Backtick. https://en.wikipedia.org/wiki/Backtick, 2023. Accessed: 2023-10-20.