TAXII Proxy

An inline visual STIX validator and conversion tool.

Contents

[Introduction 4](#_Toc157031469)

[1.1 Purpose 4](#_Toc157031470)

[1.2 Scope 4](#_Toc157031471)

[1.3 Intended Audience 4](#_Toc157031472)

[1.4 System Components 5](#_Toc157031473)

[1.5 Content Summary 5](#_Toc157031474)

[System Overview 5](#_Toc157031475)

[2.1 Description 5](#_Toc157031476)

[2.2 Features 5](#_Toc157031477)

[System Architecture 6](#_Toc157031478)

[Architecture overview 6](#_Toc157031479)

[Component diagram 8](#_Toc157031480)

[Data Flow Diagrams 8](#_Toc157031481)

[Software requirements 8](#_Toc157031482)

[Detailed Design 8](#_Toc157031483)

[Module Descriptions 8](#_Toc157031484)

[Class Diagrams 8](#_Toc157031485)

[Data Structures 8](#_Toc157031486)

[Data Persistence design 8](#_Toc157031487)

[Schema design 8](#_Toc157031488)

[Data storage and retrieval mechanisms 8](#_Toc157031489)

[User Interface Design 8](#_Toc157031490)

[Description 8](#_Toc157031491)

[Wireframes or mockups. 8](#_Toc157031492)

[Interaction patterns. 8](#_Toc157031493)

[Code Structure 8](#_Toc157031494)

[organization 8](#_Toc157031495)

[File structure 8](#_Toc157031496)

[Coding conventions 8](#_Toc157031497)

[Data Flow 8](#_Toc157031498)

[Description 8](#_Toc157031499)

[Input, processing, and output 8](#_Toc157031500)

[API 8](#_Toc157031501)

[Security Considerations 8](#_Toc157031502)

[Security Measures 8](#_Toc157031503)

[Authentication and authorization mechanisms 8](#_Toc157031504)

[Error Handling 8](#_Toc157031505)

[Testing Strategy 8](#_Toc157031506)

[Plan 8](#_Toc157031507)

[Test cases and scenarios. 8](#_Toc157031508)

[Containerization 8](#_Toc157031509)

[Deployment and Installation 9](#_Toc157031510)

[System requirements 9](#_Toc157031511)

[Installation guide 9](#_Toc157031512)

[Maintenance 9](#_Toc157031513)

[Recommendations for ongoing maintenance 9](#_Toc157031514)

[Potential areas for improvement 9](#_Toc157031515)

[Appendix 9](#_Toc157031516)

[Glossary 9](#_Toc157031517)

[references 9](#_Toc157031518)

**Table of Figures**

[Figure 1: TAXII Proxy data flow from producer 7](file:///C:\Users\corte\Desktop\Practicum\TAXII-Proxy\Documentation\Practicum_SDD.docx#_Toc157031812)

# Introduction

## Purpose

The current threat landscape calls for more advanced and adaptable methods of threat information sharing while still maintaining the ability to share intelligence at machine speeds. More formally, how does the cyber intelligence community improve the mechanisms used to share CTI at machine speeds so they do not break existing tools when standards change and allow for enhanced or extensions of the standard between industry partners? TAXII Proxy looks to mitigate data loss as an inline tool between the TAXII server and the TAXII client. TAXII Proxy looks to mitigate the following:

1. Improve the consumption of enhanced STIX objects by allowing CTI producers to publish their schemas so that subscribers can ingest their enhanced STIX objects as a part of a federated system, utilizing this middleware tool.
2. Improve predictable schema conversions by ingesting CTI schemas that either describe exactly what information a specific cybersecurity tool can ingest or describe what enhancements to the STIX standard the publisher will produce in their CTI feeds.
3. Prevent CTI data loss by verifying that the structure of the data meets the standard a specific tool downstream is expecting. If it does not meet the standard it gives a warning of what information will be lost in a visual way (i.e. visual graph). It then conducts a conversion to meet the standard the tool is expecting.
4. Finally, because the tool can visualize any enhanced STIX standard version based on a given schema, it allows analysts to continue to analyze CTI while industry tools catch up to adapting to the STIX objects being produced.

## Scope

This document intends to be a detailed technical manual for the TAXII Proxy software project. Though some information within this document must cover parts of both the STIX standard and TAXII protocol, this document is not meant to cover STIX or TAXII in detail.

## Intended Audience

The intended audience for this document are researchers and developers who would like to modify or extend the existing codebase. This software will be implemented using the python coding language. Readers should have some understanding of the python coding language.

## System Components

The following is a list of python dependencies required to use this tool:

|  |  |  |
| --- | --- | --- |
| Module Name | Description | Current Version |
| Flask | Web Framework for python Frontend |  |
| networkx | Python Graphing library |  |

## Content Summary

This document is an overview of the TAXII Proxy tool, to include design considerations of the software architecture, components, class hierarchy, data flow, and user interface.

# System Overview

## 2.1 Description

UNDER CONSTRUCTION

## 2.2 Features

The following table is the list of features and their current implementation status in TAXII Proxy:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Feature | Description | Status |
| 1 | Outbound STIX Validation | Validate STIX outbound from CTI producers to TAXII Servers | Not Implemented |
| 2 | Inbound STIX Validation | Validate STIX inbound from TAXII Servers to TAXII clients | Not Implemented |
| 3 | Visualization | Allow for STIX bundles and objects to be visualized | Not Implemented |
| 3.1 | Visualize STIX Bundles |  | Not Implemented |
| 3.2 | Visualize STIX Objects |  | Not Implemented |
| 3.3 | Visualize Schemas |  | Not Implemented |
| 3.4 | Visualize Data Loss | When a STIX object requires does not meet the expected Schema standard, display what data will be lost if sent downstream | Not Implemented |
| 4 | STIX Conversion | Allow for analysis to identify a conversion Schema | Not Implemented |
| 4.1 | Set Schema Profiles | Allow the ability to set an expected schema profile for the downstream STIX consumer | Not Implemented |
| 4.2 | Manual Schema Conversion | Allow for Users to create conversion schemas in a visual way | Not Implemented |
| 5 | Default Schemas | Allow for STIX 2.0, 2.1, and MITRE ATT&CK to be built into TAXII Proxy | Not Implemented |
| 6 | Forward STIX Objects | Allow the Ability to set conditions on when a STIX object can be forwarded downstream | Not Implemented |
| 6.1 | Unaltered STIX forwarding | Allow Users to use TAXII Proxy as a passive tool, forwarding all STIX objects downstream but alerting User of data loss based on default schemas | Not Implemented |
| 7 | User Interface | Have all features available as a part of the User Interface | Not Implemented |
| 7.1 | Schema Set User Interface | Allow for custom Schemas to be built in the User Interface | Not Implemented |
| 7.2 | Conversion UI | Allow for manual conversion setting by the User of inbound objects | Not Implemented |
| 7.3 | STIX Table View | Allow STIX to be visualized as a table | Not implemented |
| 7.4 | STIX Graphical View | Allow STIX to be visualized as a graphic | Not Implemented |
| 8 | API | API used for receiving and sending STIX bundles | Not Implemented |
| 8.1 | API POST |  | Not Implemented |
| 9 | Set Target TAXII Server | Allow User to set a targer TAXII server to either push or pull data to | Not Implemented |
| 10 | Set Behaviors | Allow users to set the behaviors when encountering unexpected STIX standards | Not Implemented |
| d |  |  |  |

# System Architecture

## Architecture overview

As seen in figure 1, TAXII Proxy is meant to sit inline between the TAXII client and TAXII server. From a producer, it can be used as a STIX verification mechanism, insuring outbound objects meet the expected schema before hitting the TAXII server. The User

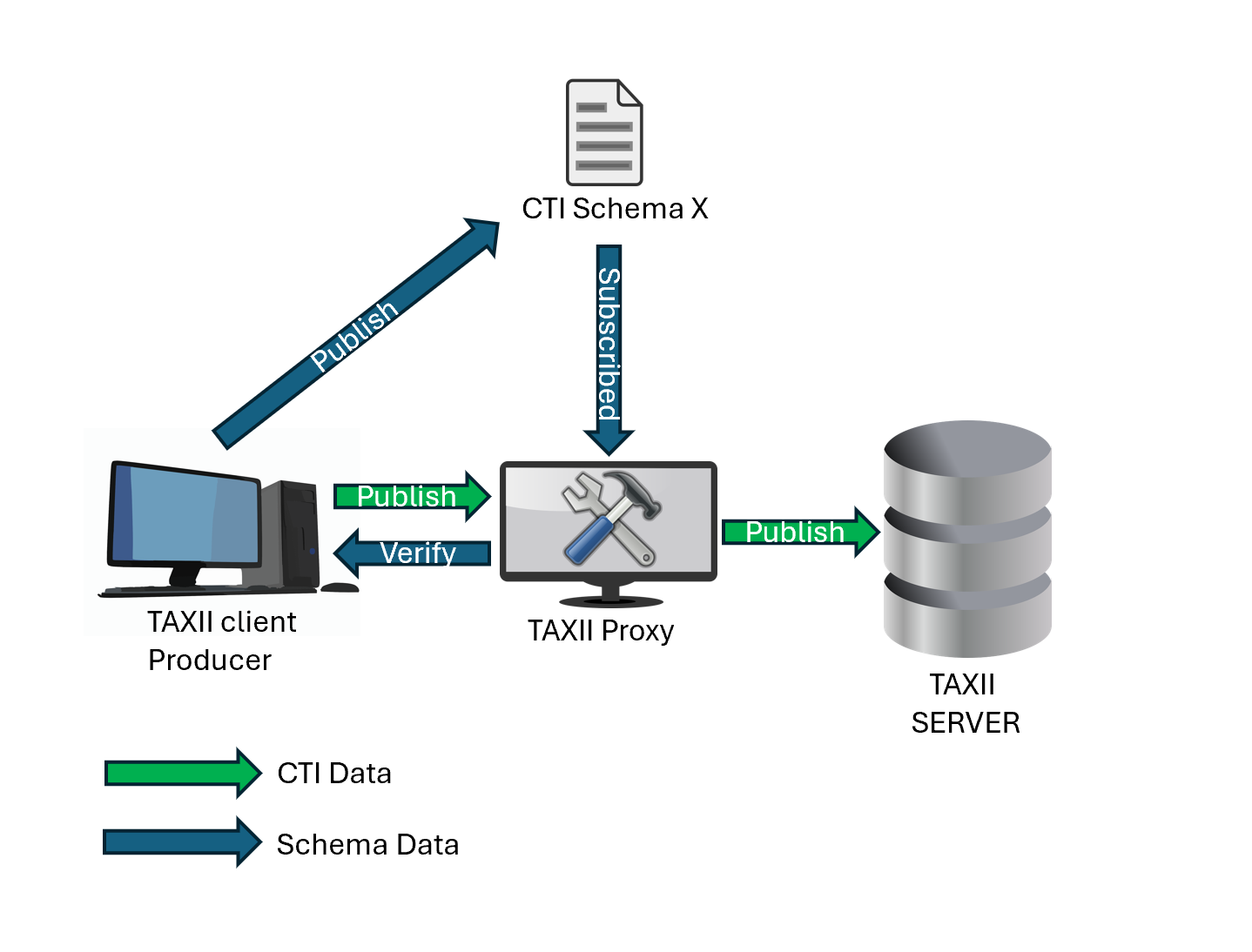


Figure 1: TAXII Proxy data flow from producer

## Component diagram

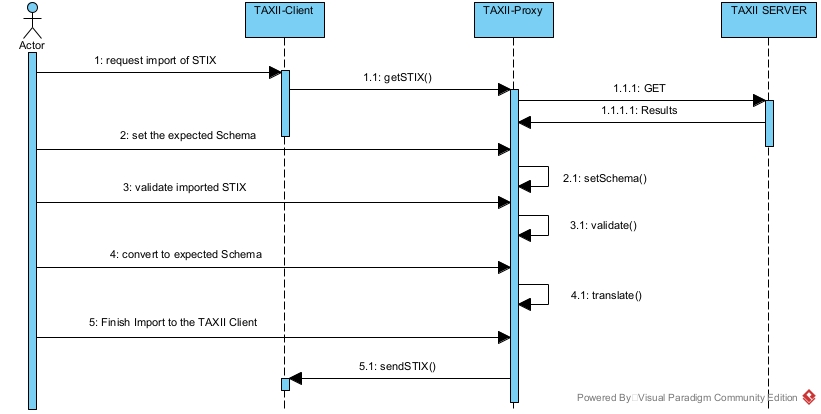
Explanation

A diagram of a computer network

Description automatically generated

## Data Flow Diagrams

Call flow when user wants to manipulate the imported STIX to a specific schema.



## Software requirements

# Detailed Design

## Module Descriptions

## Class Diagrams

## Data Structures

# Data Persistence design

## Schema design

## Data storage and retrieval mechanisms

# User Interface Design

## Description

## Wireframes or mockups.

## Interaction patterns.

# Code Structure

## organization

## File structure

## Coding conventions

# Data Flow

## Description

## Input, processing, and output

## API

# Security Considerations

## Security Measures

## Authentication and authorization mechanisms

## Error Handling

# Testing Strategy

## Plan

## Test cases and scenarios.

## Containerization

# Deployment and Installation

## System requirements

## Installation guide

# Maintenance

## Recommendations for ongoing maintenance

## Potential areas for improvement

# Appendix

## Glossary

## references