Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

**Diploma Thesis**

# Trust Model for Global Peer-To-Peer Intrusion Prevention System

Author: Bc. Lukáš Forst
Supervised by Garcia Sebastian, Assist. Prof., PhD

May 2022

# MASTER‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Forst Lukáš** Personal ID number: **465806**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Cyber Security**

## II. Master’s thesis details

Master’s thesis title in English:

**Trust Model for Global Peer-To-Peer Intrusion Prevention System**

Master’s thesis title in Czech:

**Model důvěry pro globální peer-to-peer IDS/IPS**

Guidelines:

The goal is to design and implement a trust model for distributed multi-agent environments of intrusion prevention systems (IPS). One IPS is the Stratosphere Linux IPS (Slips)[6] which will have a globally distributed peer-to-peer system. With this capability and the fact that peer-to-peer systems are permission-less, Slips determines how much can trust the data from other peers. We aim to solve this challenge and design and implement a trust model as a Slips module. The trust model should be able to evaluate the behavior of other Slips agents (which can also be acting as malicious actors) in a global peer-to-peer data sharing network and compute a trust value. The question that we want to answer is “how much can the local system trust the data coming from the said global peer?”.
The student will analyze different trust models and options to attack them. A new trust model that uses data from Slips will be proposed, and its performance will be evaluated. Finally, the model will be implemented as a module inside Slips and will enable sharing said network data with other nodes running Slips.

Bibliography / sources:

[1] Eleni Koutrouli, Aphrodite Tsalgatidou: Taxonomy of attacks and defense mechanisms in P2P reputation systems - Lessons for reputation system designers, 2010
[2] Jingpei Wang, Jie Liu: The Comparison of Distributed P2P Trust Models Based on Quantitative Parameters in the File Downloading Scenarios, 2016
[3] Tigist Abera, Ferdinand Brasser, Lachlan J. Gunn, David Koisser, Ahmad-Reza Sadegh: SADAN: Scalable Adversary Detection in Autonomous Networks, 2017
[4] Nicolas Falliere: Sality: Story of a Peer-to-Peer Viral Network, 2011
[5] Emmanouil Vasilomanolakis, Jan Helge Wolf, Leon Böck, Shankar Karuppayah, Max Mühlhäuser: I Trust my Zombies: A Trust-enabled Botnet, 2017
[6] SLIPS: Stratosphere labs intrusion prevention system, https://github.com/stratosphereips/StratosphereLinuxIPS/

Name and workplace of master’s thesis supervisor:

**Ing. Sebastián García, Ph.D. Artificial Intelligence Center FEE**

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **22.02.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **19.02.2024**

| Ing. Sebastián García, Ph.D.<br>Supervisor's signature | Head of department's signature | prof. Mgr. Petr Páta, Ph.D.<br>Dean's signature |
|---|---|---|

 

## III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

 

## Declaration

I hereby declare that the presented work has been composed solely by myself and that I have listed all sources of information used within it in accordance with the methodical instructions about ethical principles in the preparation of academic theses.

*Prague, Date* *Signature*

## Acknowledgment

First and foremost, I would like to thank my supervisor Sebastian Garcia for countless consultations and brainstormings and mainly for the significant amount of his time he invested in me and this thesis. Thank you.

I want to say thank you to the whole Stratosphere Research Laboratory, specifically to Veronica Valeros, Maria Rigaki, and Ondřej Lukáš, for providing me with all support I needed, for proofreading the thesis, and for guiding me through the academic world. I'm grateful for doing this work with you and being part of the team in Stratosphere.

Finally, I must express my very profound gratitude to my parents and my whole family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

This accomplishment would not have been possible without you.

Thank you.

Lukáš Forst

## Abstract

Most network defense systems only rely on evidence-based knowledge about past cyberattacks, known as threat intelligence. Firewalls and intrusion prevention systems rely on the shared threat intelligence generated by other systems to prevent attacks before is too late. Such threat intelligence is usually shared via centralized public and private blocklists, where a single centralized authority, hopefully, has complete control over what is published. Such centralized systems have many issues: single point of failure both technically and in trust, lack of flexibility on new data and providers, and manual trust in the providers.

To mitigate these problems, peer-to-peer networks can be used to share threat intelligence. However, because these networks are open to anyone, including malicious actors, peers need to be able to determine who to trust and which data is better to discard.

This thesis introduces Fides. Fides is a generic trust model fine-tuned for sharing security threat intelligence in highly adversarial global peer-to-peer networks of intrusion prevention agents. We design and build Fides taking into account the problems and limitations of previous state-of-the-art trust models, optimizing it for a broad spectrum of peer-to-peer networks where peers can join and leave at any time. Fides evaluates the behavior of peers in the network, including their membership in pre-trusted organizations and uses this knowledge to compute the trust. Fides continually assesses received data from the peers, and by weighting and comparing them with each other as well as with the existing knowledge, Fides is able to determine which peer provides better threat intelligence and which peers are more reliable. The received threat intelligence is always aggregated and weighted and then provided to the underlying intrusion prevention system. Among many results, our experiments show that in *the worst possible scenario*, when 75% of the network is *completely controlled by malicious actors* Fides is still able to provide the correct values of the threat intelligence data under an assumption that the other part of the network, the remaining 25%, are peers that are part of trusted organizations.

The direct contributions of this thesis are the computational model of the trust model Fides, the reference implementation of the model in Python, the simulation framework for modeling peers' behavior in the network including the implementation of the framework and the implementation of the Fides module for reference intrusion prevention system.



***Keywords:*** trust model, threat intelligence sharing, collaborative network defense, intrusion prevention system

**Abstrakt**

Většina systémů síťové obrany se spoléhá pouze na důkazy o předchozích kybernetických útocích, tzv. threat intelligence neboli informace o hrozbách. Firewally a systémy prevence narušení se spoléhají na sdílené informace o hrozbách vytvořené jinými systémy, aby předešly útokům dříve, než bude pozdě. Takové informace o hrozbách jsou obvykle sdíleny prostřednictvím centralizovaných veřejných a soukromých seznamů tzv. blocklistů, kde má jediná centralizovaná autorita úplnou kontrolu nad tím, co zveřejní. Tyto centralizované systémy mají mnoho problémů: centrální bod, jehož selhání zapříčiní kolaps celého systému jak z technického hlediska, tak i z hlediska důvěryhodnosti, nedostatečná flexibilita v oblasti nových dat a poskytovatelů a také důvěra v poskytovatele těchto dat.

K řešení těchto problémů lze ke sdílení informací o hrozbách využít sítě typu peer-to-peer. Protože jsou však tyto sítě otevřené komukoli, včetně podvodných aktérů, musí být jednotliví agenti schopni určit, komu důvěřovat a která data je lepší skartovat.

Tato práce představuje systém Fides. Fides je obecný model důvěry optimalizovaný pro sdílení informací o bezpečnostních hrozbách ve vysoce nepříznivých globálních peer-to-peer sítích agentů prevence narušení. V této práci navrhujeme a implementujeme Fides s ohledem na problémy a omezení předchozích modelů důvěryhodnosti a optimalizujeme jej pro široké spektrum peer-to-peer sítí, kde se členové této sítě mohou kdykoli připojit a odpojit. Fides vyhodnocuje chování těchto agentů, včetně jejich členství v důvěryhodných organizacích, a tyto znalosti využívá ke zjistění jak moc jsou agenti důvěryhodní. Fides průběžně vyhodnocuje přijatá data od agentů v síti a jejich hodnocením a porovnáváním mezi sebou i s existujícími znalostmi je schopen určit, který agent poskytuje lepší informace o hrozbách a který agent je spolehlivější. Výsledné informace o hrozbách jsou poté poskytnuty systému prevence narušení. Z mnoha výsledků našich experimentů vyplývá, že v *nejhorším možném scénáři*, kdy je 75% sítě *úplně ovládáno škodlivými aktéry*. Fides je stále schopen poskytovat správné hodnoty údajů o hrozbách za předpokladu, že druhá část sítě, zbývajících 25%, jsou agenti, kteří jsou součástí důvěryhodných organizací.

Přímým přínosem této práce je výpočetní model modelu důvěry Fides, referenční implementace modelu v jazyce Python, simulační software pro modelování chování agentů v síti včetně jeho implementace a implementace modulu Fides pro referenční systém prevence narušení.

***Klíčová slova:*** model důvěry, sdílení informací o hrozbách, kolaborativní obrana sítě, systém prevence narušení

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

When protecting local networks, the basic firewalls and advanced Intrusion Prevention Systems (IPS) [37] rely on evidence-based knowledge about past cyberattacks, known as threat intelligence [26]. Threat intelligence is primarily generated from the previous attacks locally or received from the remote sources. Such sources can be, for example, public blocklists [2, 8, 9] or centralized collaborative databases such as MISP [33] where the community collectively shares the threat intelligence. However, all of these traditional threat intelligence sources are, in the end, controlled by the centralized authority, which has complete control over the resource and can at anytime restrict access to these valuable assets or censor what is published and what is not. In addition to that, central authorities allow the publishing of threat intelligence only to *verified* entities and not everyone can contribute. Thus, there are many systems that were left out even though they might have precious knowledge about past attacks.

To limit the impact of one organization shutting down the database and many other problems outlined above, we can use a peer-to-peer (P2P) model and remove the central authority and single point of failure. In peer-to-peer networks, each peer has the same privileges and can freely share and receive any data without any central authority. Although P2P models solve the problem of a single central authority, they may introduce new problems. The most important questions to ask are: first, how much can each peer trust the threat intelligence received from other peers, given that their reliability changes? Second, how to deal with peers providing contradictory information when a single aggregated value is needed? Third, how to deal with adversarial peers that lie in many different ways?

In this thesis, we answer those questions and tackle the problem of trust relationships when sharing security threat intelligence data. The algorithms

that tackle trust relationships between multiple entities are called *trust models* [34]. The research field of trust models is not new, and there are many existing trust models [1, 5, 6, 18, 20, 22, 24, 27, 36]. However, these models were primarily designed to support file sharing in peer-to-peer networks and inherently operate with different types of data. Almost no P2P system was designed to fulfill the purposes of the security community. An exemption to them is Dovecot [19], a local P2P trust model designed to operate with threat intelligence data, but with the limitation that operates only in the local networks.

This thesis proposes a new trust model called **Fides**[1]. Fides is a *generic* trust model fine-tuned for sharing threat intelligence in highly adversarial global peer-to-peer networks of intrusion prevention agents. Fides was built by solving most of the problems of previous state-of-the-art trust models [5, 19] and its computational model is based on the existing trust model SORT [5] which we modified and extended to support threat intelligence sharing. Fides was optimized for a broad spectrum of networks; from local networks controlled by a single company, to public global Internet peer-to-peer networks where anybody can join and leave at any time.

Fides works with the concept of peers that belong to specific *organizations* and allows administrators to pre-trust specific peers and organizations. This pre-trusting is a novel feature of Fides, representing the common criteria of security practitioners of trusting some groups or companies more than others. The comprehensive configuration options enable administrators to share data only with particular organizations. This data filtering guarantees that no privacy-sensitive intelligence is shared with peers that should not have access to it.

Fides considers many security requirements for a global P2P trust model: pre-trusted organizations, trust values depending on the service provided by the peer, asking for the reputation to other peers, cold start problem of new peers, aggregation of information received, evaluation of how services were provided, adversarial peers that try to mislead others.

Fides was designed to be as modular and generic as possible, allowing other uses of its computational model for different data than threat intelligence by simply adding new evaluation methods.

Fides does not create or administer the low-level actions of the peer-to-peer network but instead relies on a different system that performs these operations on the network layer. The system is called **Iris**. Iris was designed and developed simultaneously in the thesis by Bc. Martin Řepa [28] and it

[1] Fides was named after the ancient goddess of trust and good faith [35].

facilitates safe and secure communications between Fides instances in the global peer-to-peer network. Communication between the two systems is done via a standard defined interface, which allows replacing the network module Iris if needed.

To interact with a real intrusion prevention system, we chose Slips [16] and implemented a module that allows Slips to use Fides for receiving and sharing threat intelligence over the network. By using this shared global knowledge, Slips can prevent attacks on the local environment even before they happen by acting upon received threat intelligence from other peers in the global network.

While evaluating the trust model performance, we simulate multiple benign and malicious peers. We also consider peers that provide incorrect data since the beginning of the simulation and intelligent malicious actors that try to exploit the trust model by gaining trust at the beginning and then manipulating the trust model to wrong conclusions.

The only way to evaluate our trust model was to simulate myriad complex situations. We proved that Fides could correctly uncover the peers' behavior in the network and make sensible decisions about the threat intelligence. Fides especially excels in the situations where it communicates with peers from trusted organizations. Moreover, in a situation where the Fides talks to at least 25% of pre-trusted peers, it can eventually determine correct threat intelligence no matter how other peers in the network behave and how many of them are adversarial.

## 1.1 Thesis Structure

This thesis explains the required background and describes the current state-of-the-art in Chapter 2. In Chapter 3 we propose a new trust model Fides and outline how it works from the high-level perspective in Section 3.1. After that, we explain problems related to gaining trust in Section 3.2. In the following Sections 3.3 and 3.4 we analyze the taxonomy of attacks and the attack vectors related explicitly to the trust models and discuss how Fides defends against them. Once we explain our design choices, we describe the entire computational model in depth in Section 3.5 and illustrate how Fides can determine trust relationships in the network by evaluating interactions between peers in Section 3.6. The last part of the computational model in Section 3.7 explains how Fides can aggregate threat intelligence from the network. Chapter 4 describes the Fides architecture and how we implemented it as a new Slips module. In the following Chapter 5 we propose simulations that evaluate the performance of the trust model and give a brief overview

of the simulation framework that we developed alongside Fides. Chapter 6 then describes the results that we discovered in the evaluations. Finally, Chapter 7 concludes our results and proposes further areas of improvement for Fides. We also include an appendix with multiple interesting cases of evaluations discovered in Chapter 6.

## 1.2 List of Contributions

The contributions of this thesis are:

- Analysis of state-of-the-art trust relationships models in peer-to-peer networks. (2)
- Design of Fides, a generic trust model fine-tuned for sharing security data in global adversarial peer-to-peer networks. (3.1)
- Design and implementation of multiple methods to evaluate interactions between peers that share threat intelligence data. (3.6)
- A method that enables weighting and aggregation of threat intelligence from multiple peers. (3.7)
- A complete working reference implementation of Fides in Python. (4)
- An implementation of a Slips module that allows the use of Fides for global threat intelligence sharing. (4)
- A simulation framework for modeling any environment for Fides evaluation. (5)
- An simulated evaluation of Fides in different environments and analysis of its behavior in unfavorable situations. (6)

# Chapter 2

# Previous Work and Background

This chapter outlines security tools used in the local networks and describes what threat intelligence is. Then we describe the basic concepts of peer-to-peer networks and how the trust relationships are modeled in them. Modeling trust in peer-to-peer networks is not a new concept. Thus we describe previous work in this area and explain why we designed a new trust model for sharing threat intelligence in peer-to-peer networks of Intrusion Prevention Systems.

## 2.1 Intrusion Detection/Prevention System

An Intrusion Detection System (IDS) is a system that continuously monitors a network for malicious activity and reports its findings to the network administrator [4].

An Intrusion Prevention System (IPS) can be seen as an extension of an IDS. It is a network security tool that continuously monitors a network for malicious activity and takes action to prevent it, including reporting and blocking when it does occur [37]. There are plenty of commercial and open-source solutions that offer IDS/IPS capabilities. One of them is Slips.

## 2.2 Slips

Slips is an open-source behavioral Intrusion Detection and Prevention System developed by the Stratosphere Research Laboratory of the Faculty of Electrical Engineering at the Czech Technical University in Prague. Slips

uses machine learning to detect malicious behaviors in the network traffic. It was designed to focus on targeted attacks, detect command and control channels, and provide good visualization for the analyst [16]. Its design allows developers to extend Slips by developing new modules that add new functionality, such as additional machine learning models for network analysis, or enable Slips to communicate over the network with other Slips instances.

As the goal of the thesis is to design and build a trust model that cooperates with the IDS, we chose Slips as our IDS/IPS mainly because it is a highly modular system. Its modularity allowed us to integrate deeply into the system without the need for architectural changes on it.

## 2.3 Threat Intelligence

Threat intelligence is the provision of evidence-based knowledge about existing or potential threats [26]. Such knowledge is produced by, for example, intrusion detection/prevention systems. When the prevention systems receive the threat intelligence, they act according to that. For example, if an endpoint detection system provides threat intelligence that claims that IP $x.y.z.w$ is malicious, a blocking system with access to the firewall will block access from and to this IP address.

This introduces a trust relationship between the system that detects the malicious behavior and the system that can block it. The system taking actions, for example, blocking the IP address, needs to trust the threat intelligence that it received from other systems in the network. This also means that the trust in some threat intelligence is an important aspect when sharing and acting according to the said threat intelligence.

In this thesis, we will operate with the threat intelligence generated by Slips. Slips uses multiple internal detection modules, where each provides its assessment of the traffic network flows. Each module's assessment is then taken into account when Slips aggregates it and produces threat intelligence that consists of score and confidence. The threat intelligence (score and confidence) refers to a target - primarily to an IP, domain, hashes of files, or any other unique identification of the subject that Slips classified with score and confidence.

The score is a threat intelligence indicator and explains how much Slips thinks the target is malicious or benign. Confidence then describes to what extent Slips believe that the score is correct. In this thesis, whenever we refer to threat intelligence, we refer to this score and confidence provided by Slips.

## 2.4 Peer-to-Peer Networks

A peer-to-peer network is a distributed network of computers without a direct hierarchy where nodes (peers) communicate with each other directly, without using a centralized server or any authority [30]. Unlike more traditional client-server networks, where the server provides data and the client consumes them, in peer-to-peer networks, all peers are usually similar in terms of permissions and what services they provide. Peers are free to join and leave the network at any time, making the network highly dynamic, and it does not provide any guarantees about data availability. Nowadays, peer-to-peer networks are mostly used for file sharing, which Napster popularized in 1999 [29].

## 2.5 Trust in Peer-to-Peer Networks

There are many existing approaches to model trust in peer-to-peer networks and many existing trust models. Unfortunately, most of them were explicitly designed with file-sharing in mind, as that is the most common use case for peer-to-peer networks. However, multiple trust models are generic enough, such as SORT described in Section 2.5.3, or designed specifically for sharing threat intelligence, for example, Dovecot described in Section 2.5.2.

### 2.5.1 Problems of Trust

In peer-to-peer networks, where no central authority can enforce rules and assess whether the peers are honest or not, it can be problematic to find out how much the peers can trust each other. Because anybody can freely join and leave, a knowledgeable adversary can misuse the network for their benefit by providing inaccurate data to the rest of the peers in the network. For that reason, the peers need to be able to tell how much they can trust each other.

An algorithm that models such trust relationships and can assign the trust value to each peer is called a *trust model* [34]. An analysis of existing trust models was performed by Shree and Basha in [31] and by Pinyol et al. [27]. We took both analyses into account when researching existing implementations of trust models that might be a good fit for our highly adversarial global peer-to-peer network for sharing threat intelligence.

### 2.5.2 Dovecot

Dovecot is a trust model developed by Dita Hollmannová [19]. This trust model was designed explicitly for Slips to share threat intelligence in *local* peer-to-peer networks. Dovecot counts interactions between the peers, and the more interaction peers have, they have a higher base for trust. This design is based on the Sality botnet [12], where botnet peers were storing the *goodcount* that counted the number of interactions between each other. The idea behind this is simple yet very effective. The more peers talk to each other, the more they are trusted.

The final peer trust is computed using *goodcount* combined with Slips's threat intelligence about the peer. This is possible since, in local networks, Slips knows every IP address and can obtain a complete overview of the device on the network and its behavior.

Unlike other trust models where new peers start with no trust, Dovecot trusts new peers by default. However, the authors mentioned in future work that this property should be explored more in detail, so this may change in future versions of Dovecot.

### 2.5.3 SORT

Self-ORganizing Trust model (SORT) aims to decrease malicious activity in a P2P system by establishing trust relations among peers in their proximity [5].

In SORT, peers are assumed to be untrustworthy until they prove otherwise by providing *good* service to the local peer. For example, in file-sharing networks, this might be providing access to required files or, as in our case, providing threat intelligence about some target.

Unlike other trust models, for example, Eigentrust [22], SORT does not need a priori information about the network or any pre-trusted peers to operate effectively in the network. Peers do not try to collect trust information from all peers. Each peer develops its own local view of trust about the peers who interacted in the past [5].

Even though peers do not collect trust information from other peers, there is a recommendation system in place, where any peer can ask for a recommendation about another peer. Because of the nature of the peer-to-peer network, this allows the trust model to gain faster knowledge about new peers that can join and leave at any time.

Even though SORT's authors evaluated the algorithm on the file-sharing peer-to-peer networks, the algorithm is generic enough to be reused for different environments. It achieves this thanks to its computational model,

which is generic and flexible. As we mentioned previously, the peers are gaining trust by providing services. When they provide the service to the local peer, the trust model evaluates this interaction using the interaction evaluation function. This evaluation is then passed to the computational model, which assigns the trust value to the peer.

Thanks to this design, we can adapt the trust model for different environments by re-implementing the interaction evaluation function. For example, in the file-sharing environment, such an interaction evaluation function can be based on the speed of the upload/download so that peers will prefer nodes with a faster internet connection.

However, we cannot use SORT directly, primarily because of the recommendation algorithm, which can, in some cases, request a recommendation even from peers that are not *trusted enough*. We describe this situation more in detail in Section 3.5.4. Another reason was that SORT does not support any pre-trusted peers or organizations, which is something we believe can significantly improve the model performance. Moreover, threat intelligence is a specific piece of data where we do not request data only from a single peer but rather from multiple peers and then we aggregate it. Thus, we could not use SORT as is, but rather extend it and modify it.

### 2.5.4 Related Trust Models

The field of trust models for peer-to-peer networks is not new, and there are many other interesting trust models that attempt to tackle trust relationships between peers. In this section, we describe only notable trust models relevant to sharing threat intelligence in the peer-to-peer networks but did not base our design on them at the end.

Sadan, proposed in [1], is a trust model that uses committees and computational challenges to identify malicious peers in the network. Sadan requires hardware chips, Trusted Platform Module (TPM), to verify the running software, which allows other peers to verify the trustworthiness of the peer. The requirement of TPM chips means that Sadan cannot be used as the base for our trust model, as we do not want to limit our solution to specific hardware.

Xiong and Liu then proposed PeerTrust [36], which uses a transaction-based feedback system to determine the reputation and trust of the peers. Even though the model is generic and was designed for peer-to-peer networks, it was optimized for *e-commerce communities* that inherently have completely different behavior than the network for sharing threat intelligence, which needs to protect itself from the malicious peers.

Huynh, Jennings, and Shadbolt proposed the FIRE trust and reputation model [20] which was designed for open multi-agent systems. FIRE incorporates multiple different trust metrics that are aggregated and provide a view of the agents' behavior. After analyzing the paper, we do not believe that *witness information* [20] can be applied in settings for sharing threat intelligence because it relies on a fact that the peers are honest. For that reason, we decided not to use FIRE as the base for our further design.

In [18], He et al. proposed gathering peers and modeling their trust relationships as *clouds*. They proposed an *extended-cloud-model-based trust model* (ECMBTM). ECMBTM utilizes cooperation history between peers to compute trust clouds. This trust model attempts to mitigate the cold start problem by directly propagating trust to another peer in the network. We find the application of this method in our setting problematic because it would be easier for the malicious peers to propagate their false information through the network.

In [24], Li et al. proposed a machine learning-based trust model for the collaboration of IDS instances. Even though this model is closely related to our use case, we cannot use it because it uses a central certificate authority to issue a registration for the new peers. In our design, we want to have a permission-less peer-to-peer network where each peer is equal, and there is no central authority that controls the network and thus no single point of failure.

After careful analysis, we decided to create a new trust model, **Fides**, that is based on the SORT algorithm with various modifications and fine-tuning. We describe how Fides work in the following Chapter 3. We chose SORT because it is easily extensible and robust. Moreover, the evaluation provided by the authors in [5] promised interesting results.

# Chapter 3

# Trust Model Design

This thesis aims to design and implement a trust model for sharing threat intelligence in global peer-to-peer networks where the peers are instances of intrusion prevention systems. The previous chapter outlined what threat intelligence is, how it is generated, and why trust relationships are essential when making decisions based on the said threat intelligence. We described the trust models in the context of peer-to-peer networks and analyzed the notable ones. We discovered the promising trust model SORT [5], which we described in Section 2.5.3. After careful analysis, we decided to use SORT's algorithm as a base for our trust model design mainly because of its flexibility and modularity.

In this chapter, we propose a new trust model **Fides**. Fides was named after the ancient goddess of trust and good faith Fides [35]. The trust model utilizes modified SORT's computational model with multiple modifications and extensions that allows it to work in highly adversarial global peer-to-peer networks effectively. Fides is a generic and heavily configurable trust model specializing in sharing threat intelligence. Thanks to its modular architecture, it can operate with any data and it is not limited only to threat intelligence. In Section 3.1 we describe our trust model design and explain how Fides works on a high level, the inputs and outputs, and how it behaves in which situation. After outlining the general overview, we analyze the weaknesses of the trust models and how they apply to Fides. First, we start with *the cold start problem* in the Section 3.2, which describes how peers can gain trust when they are new in the network and how Fides tackles this issue. In the next Section 3.3 we analyze possible attack vectors on our trust model, and then we describe the taxonomy of attacks in the next Section 3.4. Once all trust model requirements are explained, we dive deep into Fides's computational model in the next Section 3.5 and explain how it can uncover

trust relationships in the network. The following Section 3.6 explains how Fides can evaluate interactions between peers and how that affects trust. Because Fides specializes in sharing threat intelligence and integrates with Slips, in the next Section 3.7 we explain how Fides aggregates the weighted threat intelligence from the network.

In the upcoming pages, we use the following terminology to talk about the trust model:

- **Target**: An identification of a resource for which is Slips able to generate threat intelligence. It can be, for example, either an IP address, a domain, or hashes.
- **Local Peer**: The unique local instance of Slips that is connected to the global P2P network and runs Fides. In equations, we use $i$ when referring to the local peer.
- **Remote Peer**. A peer on the Internet is connected to the global Slips P2P network. In equations, we use $j$ when referring to the remote peer.
- **Service Trust**: How much does Fides trust a remote peer that it provides the local peer with good service. In other words, to what extent does Fides trust a specific peer that it provides correct and valuable threat intelligence. We denote it $st$ and discuss it in detail in Section 3.5.1.

## 3.1 General Overview of Fides

In this section we describe how Fides work from a high level perspective. We reference chapters and sections further into the thesis that provide more information and describe particular situations and solutions in more detail.

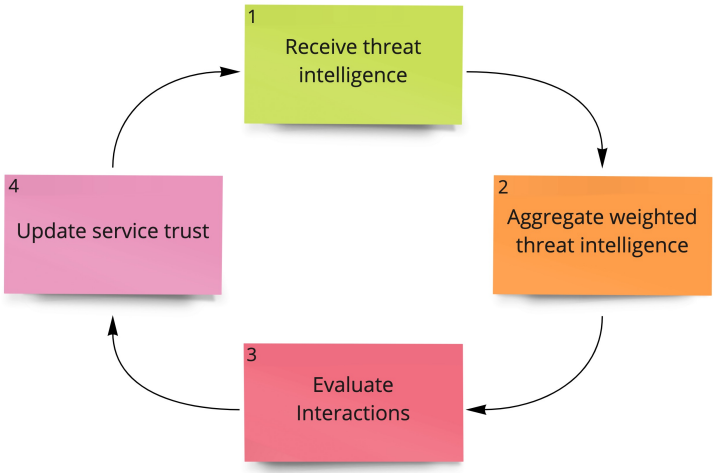


Figure 3.1: Generic Trust Model Life Cycle of Fides

Fides operates in four general phases, which are visualized in Figure 3.1. In the first phase, a local Fides instance receives threat intelligence data from the remote peers in the network. How Fides receives data from the network is described in Chapter 4 where we discuss its architecture.

In the second phase, Fides aggregates the threat intelligence data using the trust data it has for each remote peer. In general, data from highly trusted peers have a higher impact on the final aggregated threat intelligence than the data from peers with low trust. How does Fides do that is described in the Section 3.7. The aggregated threat intelligence is also sent to Slips as an output of the trust model.

In the third phase, Fides evaluates the interactions with each peer. Fides computes how much it was satisfied with the threat intelligence it received from each remote peer. The evaluation does not depend on the *content* of the threat intelligence and therefore is a generic method. This satisfaction metric has then a direct influence on the trust relationship between the local and remote peers because it is used in the next step to compute trust data. The evaluation process and possible interaction evaluation functions are described in detail in Section 3.6.

In the fourth step, Fides updates the trust data for each peer according to the satisfaction that is computed in step number three. Computations that allow Fides to do that are described in detail in Section 3.5.

All operations, including the data flow and the communication with other peers and Slips, can be found on the operational diagram shown in Figure 3.2.

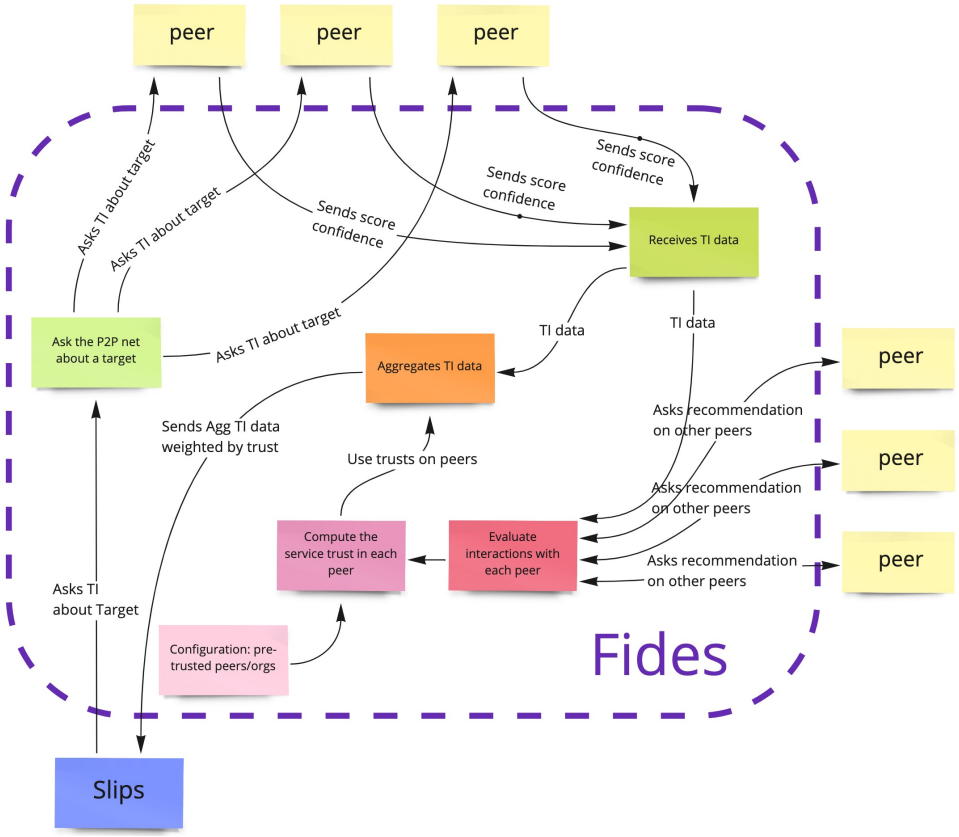


Figure 3.2: Detailed operational diagram of Fides trust model. All the inner parts of Fides are represented, together with the external parts: Slips and the P2P network.

## 3.2 Cold Start Problem

A dynamic and global environment such as a global peer-to-peer network is open to anyone since any peer can freely join and leave. Because of that, the local peer will encounter many other peers that were not seen before. Therefore, the trust model does not have any information about their reliability or how much it can trust them. New benign peers need to be *somehow* trusted by the local peer in order to be a useful part of the network. However, the local peer also needs to be able to discover new malicious peers that are trying to gain its trust.

The problem of how to know something about a new entity in order to quickly work better is called the *Cold Start Problem* [6]. For Fides, it means how to compute a good initial trust for new unknown peers.

We selected several solutions to this issue, which are all implemented in Fides. Fides also combines them according to a provided configuration with the aim to achieve the best result for the cold start problem with adversarial peers.

### 3.2.1 Static Initial Trust

In this approach, whenever the trust model encounters a new peer, it assigns a static value as an initial trust. The value is assigned by pre-choosing some third-party trust models in the configuration.

For example, in the **Dovecot** trust model [19], every peer starts with trust 1 (highest possible), and various interactions can lower the trust in the peer to 0. In other words, the trust model considers new peers honest from the beginning, and only during this time their reputation can be lowered when they perform incorrect interactions or are discovered as a malicious peer.

On the other hand, the **Sality** botnet [12] uses a value called *goodcount* as a counter of good interactions with any other peer, the higher the *goodcount*, the greater trust the local peer has on the remote peer. The goodcount for each new peer starts with 0 in Sality. Meaning, that the botnet does not trust fresh peers at all and they can gain trust only by following the Sality protocol.

The model of *static initial trust* is easy to implement, but it requires assumptions about the network. If the network is considered *mostly benign*, it might be safe to use an initial trust of 1, however, for highly adversarial networks using an initial trust of 1 might be dangerous and it is better to use 0. Using low initial trust and no mechanism to gain more trust fast

means that the benign peers that joined recently do not affect the final decisions of the model, even though they might have useful information about adversaries.

Static initial trust is supported by Fides as a form of fallback when no other cold start technique is used. The administrator provides a configuration that contains the initial reputation for each new peer.

### 3.2.2 Pre-Trusted Peers

This master thesis was done simultaneously to the master thesis on global P2P security TI sharing by Bc. Martin Řepa [28], called Iris, which implements the new idea of pre-trusted peers in organizations for the Slips IPS. Therefore, Fides works with the concept of pre-trusted organizations which have pre-trusted peers. Iris implements the concept of pre-trusted organizations, and Fides uses this knowledge to assign a higher or lower trust to new peers.

The global P2P framework implemented by Řepa supports these type of peers and provides a cryptographically-secure way how to identify a single peer in the network, and its membership in an organization. This allows Fides to *pre-trust* specific peers or all the peers from organizations by assigning them an initial value.

Fides can be configured to use pre-trust in two different ways. First, to assign the pre-trusted peers an initial *reputation*. This means that the peer will have an initial reputation, but it will be required to interact with the local peer and it will slowly change that initial reputation according to its interactions with others. All the interactions will be evaluated and Fides will compute a service trust for the peer, as described in Section 3.6.

Second, Fides can use the initial pre-trusted value read from the configuration as the final service trust. This effectively means that Fides will not evaluate any data received from the pre-trusted peer and this service trust will be kept forever.

This configuration for Fides is called *enforceTrust*. If it is enabled and thus $enforceTrust = False$ is set in the configuration, Fides uses the first variant where the trust for the peer will move during the interactions. If the administrator uses $enforceTrust = True$, Fides uses the second option and fixates the service trust for the peer to a set *pre-trust*.

Both options help solve the cold start problem for specific peers and organizations, as they will start with a high reputation or fixed service trust. Which organization or peer to trust is completely left to the administrator of Slips. However, as the administrator needs to know the identity of the

peers or organization, it does not solve the cold start problem globally for all peers.

### 3.2.3 Recommendations

As the local peer might have multiple remote peers that it trusts enough, Fides uses these relationships to ask the remote peers about how much they trust a new peer. Fides only asks for recommendations once: when the local peer finds a new peer for the first time.

Using a recommendation system introduces new attack vectors that can be exploited by adversaries, either by getting trust for the malicious peer or by lowering trust in honest peers that might have some threat intelligence about the malicious actor. These attacks are called *bad-mouthing* and *unfair praises* and we need to consider them and implement countermeasures.

Because of the possible attacks, the local peer should not solely rely on the network recommendations when computing the final service trust for the fresh peer. In case when the recommending peers are malicious, it might skew the decisions of the local peer for the time being. In order to solve this, when computing the final service trust for the remote peer, the local peer should take into account its own interaction with the peer as well as the received recommendations.

Moreover, the local peer should request recommendations only if it has *enough* trusted remote peers, otherwise, it can expose itself to *bad-mouthing* and *unfair praises* attacks more easily.

Fides employs recommendation systems based on SORT [5] but with more strict rules when it is actually used and combines it with the pre-trusted peers (3.2.2) as well as with the static initial trust (3.2.1) as a fallback when no other option is available due to constraints such as having not enough trusted peers. The algorithm used for the recommendation system is explained in detail in Section 3.5.

## 3.3 Attack Vectors

Since Fides is a trust model that computes how much to trust peers, it is potentially open to attacks from adversarial peers. Adversarial peers are peers that know how to talk the protocol and manipulate the recommendations, or threat intelligence data in order to influence the final decisions.

Adversarial peers can try to (i) send bad threat intelligence data; (ii) lie about a peer that is benign, and (iii) lie about a peer that is malicious.

### 3.3.1 Influencing Aggregated Score & Confidence

The main output of Fides is the aggregated score and confidence of a group of reports on a target. The sequence of actions typically are (i) Slips wants to know what the P2P network thinks about target $T$; (ii) it then asks some peers; and (iii) uses Fides to aggregate the scores and confidences sent by all the peers. The aggregated score and confidences are used for computing the service trust of peers and also to weight the aggregated score and confidence of the data sent by the peers.

Any attacker wants to ultimately influence these aggregated values either to make malicious IP/domain seem to be benign or another way around. However, for that to happen, the attacker needs to gain sufficient service trust. For more information about the aggregated threat intelligence see Section 3.7.

### 3.3.2 Influencing Service Trust

Fides always computes service trust for the peers locally and does not take over the service trust computed by any other peer. How does Fides computes the service trust is described in detail in Section 3.5.

Malicious peers can influence the service trust value for some other remote peer in the network in the eyes of the local Fides in two situations.

Firstly, the peers can influence the service trust in a peer by manipulating their recommendation responses when the Fides encounters the peer for the first time and asks the network for the recommendations on it. For that reason, the recommendation protocol is engaged only when the network is *trusted enough* and only for the first time when the new remote peer encountered. We describe this more in detail in Section 3.3.3 below.

The second case when the malicious peer can indirectly influence the service trust for any other remote peer is a situation when Fides uses one of the interaction evaluation strategies that utilizes the aggregated threat

intelligence (3.6.2, 3.6.3, 3.6.5). Because in that case, even data from malicious peers are taken in account when computing final satisfaction with the interaction for each peer so by submitting incorrect data the group of malicious peers can influence the interaction evaluation result which will lower the service trust in benign peers.

This is expected as this interaction evaluation strategy (3.6.2, 3.6.3, 3.6.5) uses aggregated network opinion to evaluate the interactions. Thus if the *wrong* opinion is in majority, and while considering the service trust of each peer, it is taken into the account even though it is *wrong*.

For that reason, the intermediate goal of an attacker is to gain the service trust of the local peer in order to have *any* influence over the decisions the Fides makes. We explore this behavior in more detail in the experiments in Section 5.3, when we let malicious peers gain the service trust at the beginning of the simulation. In addition to that, we describe how resilient Fides is to these types of attacks as part of the experiment results analysis in Section 6.2.

### 3.3.3 Influencing Peers Reputation

When a new peer joins the network, Fides in some cases requests recommendations from other peers in the network. We go into detail of this process further in Section 3.5.4.

Because Fides asks for the recommendation, it is possible that one or more of the peers providing the recommendations is malicious and it provides *incorrect* recommendations with the goal either to silence a benign peer or to support another malicious peer.

Even though the reputation of a peer can be skewed by the attacker, it is still able to gain *correct* service trust by following the protocol and providing useful data. The service trust Equation 3.1 suggests that the more experience a local peer has with a remote peer, the more it ignores the initially received recommendations. This means that the service trust will tend to converge to *correct* values that do not necessarily depend on the initial recommendations and eventually will lose that information completely. In other words, if the peer's initial reputation was *incorrect* (from the ground truth point of view), it will only take the peer longer to gain *correct* service trust, but eventually, it will end up with the same value as with the correct reputation value. We talk more about the service trust and how does it behave further in Section 3.5.1.

## 3.4 Taxonomy of Attacks

We were inspired by the thorough threat model analysis in Dovecot [19] and based our own analysis on the same original paper from Koutrouli and Tsalgatidou [23] which describes the taxonomy of different attack methods on reputation systems in peer-to-peer networks. They classify the reputation attacks in the following categories.

### 3.4.1 Unfair Recommendations

This category describes the behavior when a peer provides incorrect data. The peer does not need to be necessarily malicious in order to do that, it can also have not enough data to make correct decisions or maybe it is missing some important information. In the case of Fides, these types of attacks also influence service trust as well as the reputation system, because the service trust depends on the initial reputation. Moreover, the malicious peers can collude to amplify the effect on the final computations of the trust model.

The intent of the malicious peers, in this case, is to lower someone's service trust/reputation (*badmouthing attack*) or to make someone's service trust/reputation higher (*unfair praises*). In a case of service trust, this is not possible directly, but rather by colluding with multiple high trusted peers as described in detail in Section 3.3.2. In the case of reputation, this is possible if the malicious peer is selected as a recommender. Fides mitigates both of these problems by asking numerous peers for their opinion (in case of service trust) and by asking only pre-selected and highly trusted peers in case of recommendations. Of course, it is not possible to eliminate the possibility of a malicious peer being asked for the recommendations, that is why, in experiments, we simulate malicious peers as *Malicious Peer* (5.2.4) behavioral patterns. In simulations we then evaluate what network topology is needed in order for Fides not to be easily manipulated into believing the malicious peers.

*Inaccurate recommendations* are a type of *unfair recommendation* when an honest peer provides wrong data due to a lack of complete information. This can happen, for example, because they were not attacked by the adversary (yet), and they consider them to be benign because they have no reason to see it otherwise. Another example can be a peer that does not have the latest threat intelligence data from the black lists or other remote resources. These peers are included in the experiments as well, we call that *Confident Incorrect* (5.2.3) behavioral pattern.

Koutrouli and Tsalgatidou [23] also mention *Random opinions* where the peer is essentially providing random data. We simulate this in our experiments as well, because there will be peers, in the network, that simply do not have enough information to make a good and confident decision about some target. This is the *Uncertain Peer* (5.2.2) behavioral pattern.

Because of the nature of Fides, which aggregates all network opinions it receives, the worst-case scenario is the situation where the attackers collude together because it amplifies their effect on the final aggregated score & confidence. However, our trust model uses service trust during computing the final score with confidence so, in order for attackers to influence this decision, their total service trust must be higher than the service trust of the benign peers. This makes it harder for the adversary to overturn the decisions in their favor because it forces them to gain the service trust of all their peers. In simulations, we have malicious peers that collude (and lie about the same targets) as well as peers that do not collude and lie about different targets.

### 3.4.2 Inconsistent Behavior

In the aforementioned Section 3.3.1 any malicious peer needs to gain *some* service trust in order to have the ability to meaningfully influence the trust model's decisions. This leads to malicious peers that will have different behavior when they try to gain the service trust and when they provide misleading data to achieve their goals. This behavior is equivalent to the *Traitors* from [23]. Fides tries to mitigate this problem with some of the interaction evaluation strategies that compare individual threat intelligence data from a single peer to aggregated network opinion (such as 3.6.2). Thanks to these strategies, even peers that gained service trust at the beginning can be eventually identified as malicious and their service trust will be lowered whenever they provide threat intelligence data that are different from the aggregated ones.

However, even the honest peers can have inconsistent behavior, mainly when they do not have enough information about IP/domains. In experiments, we simulate this behavior for honest peers with *Uncertain Peer* (5.2.2) behavioral patterns. For malicious peers, we have a period during which they provide correct and consistent data, allowing them to gain the service trust.

### 3.4.3 Identity Management Related Attacks

The service trust and reputation are tied to the peer's identity. In our case, Fides uses a peer's identity that was provided by the Network Layer

[28]. From the technical point of view, the identity is, in fact, a public key, and any data the peer provides is signed with the peer's private key. Thus, we can verify that the data were provided by the owner of the private key to said public key (identity). Moreover, any peer in our network can belong to one or more organizations that are, again, represented by their public key. Peers prove their membership to the organization by providing their own public key signed by the organization's private key. The identity, as well as the organization's membership, is cryptographically verified by the network layer [28] and Fides does not perform any additional verification.

#### Impersonation

Thanks to the data signatures and identities tied to private/public key pairs, the *Impersonation* based attacks are then possible solely in cases when the attacker gained access to the private key of the peer. Unfortunately, this type of attack is not possible to prevent completely. However, when the attacker gains access and starts submitting incorrect data, Fides will start lowering the service trust associated with that identity, thus eventually limiting the attacker's influence.

#### Man-in-the-middle attack

*Man-in-the-middle attack*s are attacks when a third party is able to either intercept or manipulate the transmitted data. From Fides point of view, the data manipulation is not possible, as the data are signed by the sender and the network layer [28] ensures that the signatures are verified. On the other hand, the network layer is designed in a way that peers pass messages to each other through the network [28], so any malicious peer can choose not to pass down the message. How this affects the propagation of messages is part of the experiments in said paper [28].

### 3.4.4 Whitewashing & Sybil Attack

Due to the nature of the global peer-to-peer network, where many devices run behind NAT[1] or even NAPT[2] and have the same IP address, the identity is not associated to an IP address. However, this also means that any device can have multiple identities and can essentially generate new ones as time

[1] Network address translation - a router mapping multiple IP addresses from the private network to a single public IP address.

[2] Network address and port translation - similar to NAT, but on the private network even the ports are used during the translation process.

goes by. This opens Fides to other types of attacks such as *Whitewashing*, where the malicious peer drops an identity that was discovered as malicious and its service trust dropped in 0, and then it generates a new, fresh identity. However, this behavior does not benefit the attacker as much as in Dovecot [19], because Fides assigns the initial service trust 0, instead of 1. In other words, Fides distrust new peers by default, so whenever a peer drops its identity and creates a new one, it starts with a service trust of 0.

As it is not expensive to generate a new identity, it is not costly for the attacker to perform a *Sybil attack*. Sybil attack refers to a situation where a single malicious peer creates multiple identities and uses them in concert to defeat the system [11]. In our case, attackers can maliciously flood the network with wrong data thus making some of the interaction evaluation strategies from Section 3.6 perform poorly. Moreover, if the attacker is able to gain *some* service trust for its malicious peers, it can effectively overtake the network and influence most of the decisions that Fides makes. The defense against this attack is to make it *computationally hard* to join the peer-to-peer network, for example, by making it hard to compute peer IDs or by letting peers solve some other type of computational puzzle.

However, we did not introduce any of these measures to our system, and we leave that as part of future work in Section 7.1.3.

## 3.5 Computational Model of Fides

This section describes how Fides determines to whom and how much it can trust other remote peers. Our trust model expresses trust in a specif peer with metrics called *service trust*. Service trust is a value that describes how much the local peer can trust a *specific* remote peer.

In the following pages, we describe the process top-down starting with the most important parts - service trust - and then breaking it down into bits. Note that there are two main ideas behind most of the equations.

The first one, is that we want to robustly capture the average behavior of the peers. In order to do that, we will be computing the average behavior of the peers and then approximating the deviations from said behavior.

The second part compares and weights first-hand experience with the remote experience. First-hand experience is what happened between local and remote peers during the time they interacted. This can be, for example, threat intelligence sharing, file-sharing, or the results of the recommendation protocol. Remote experience is what happened between one remote peer and another remote peer. In other words, first-hand experience for peer $j$ are actions between $j$ and $z$. Whenever $j$ shares information about these actions with peer $i$, for $i$ it is a remote experience.

Table 3.1 describes the most important notation we use in the following sections.

| | |
|---|---|
| $i$ | local peer, instance of Fides |
| $j$ | remote peer somewhere on the internet |
| $st_{i,j}$ | service trust - how much $i$ trusts $j$ that it provides good service |
| $r_{i,j}$ | $i$'s reputation value about $j$ |
| $rt_{i,j}$ | $i$'s recommendation trust about $j$ |
| $sh_{i,j}$ | size of $i$'s service history with $j$ |
| $s^k_{i,j}$ | $i$'s satisfaction value with interaction with peer $j$ in window $k$ |
| $w^k_{i,j}$ | weight of $i$'s interaction with $j$ in $k$ |
| $f^k_{i,j}$ | fading effect of $i$'s interaction with $j$ in $k$ |

Table 3.1: Fides Computational Model Notation

### 3.5.1 Service Trust

As outlined previously, service trust $st_{i,j}$ is a value that describes how much peer $i$ trusts that remote peer $j$ will provide a *good service* [5]. We compute the $st_{i,j}$ in the Equation 3.1 by weighing local experience with peer's $j$ service, with the reputation $j$ got from the network when it was first seen by $i$. The used weight is the size of the service interaction history $sh_{i,j}$ to global maximal history size $sh_{max}$.

$$st_{i,j} = \frac{sh_{i,j}}{sh_{max}} \cdot \left( cb_{i,j} - \frac{1}{2} ib_{i,j} \right) + \left( 1 - \frac{sh_{i,j}}{sh_{max}} \right) \cdot r_{i,j} \tag{3.1}$$

Equation 3.1 implies that the more interaction there was between peers $i$ and $j$, the bigger impact on $st_{i,j}$ it has. In other words, the more $i$ and $j$ interact the less $i$ relies on the reputation that $i$ computed from the values provided by the network, at the time when $j$ was seen for the first time by the peer $i$.

### 3.5.2 Local Experience for Service Trust

The first part of the Equation 3.1 contains *competence belief* - $cb_{i,j}$, and *integrity belief* - $ib_{i,j}$. Both values are based solely on the history of the interactions that the peer $i$ experienced with the peer $j$.

#### Competence Belief

*Competence belief* represents how much did peer $j$ satisfied local peer $i$ with the past interactions. We measure it as an average of interactions from the past [5].

$$\begin{aligned} cb_{i,j} &= \frac{1}{\beta_{cb}} \sum_{k=1}^{sh_{i,j}} s_{i,j}^k \cdot w_{i,j}^k \cdot f_{i,j}^k \\ \beta_{cb} &= \sum_{k=1}^{sh_{i,j}} s_{i,j}^k \cdot w_{i,j}^k \end{aligned} \tag{3.2}$$

It holds that $0 \leq cb_{i,j} \leq 1$ and where $s_{i,j}^k$ is the evaluation of the interaction in window $k$, $w_{i,j}^k$ is the weight of the interaction (how important it was) and $f_{i,j}^k$ is the fading effect of that interaction. We describe $s_{i,j}^k$, $w_{i,j}^k$ and $f_{i,j}^k$ in Section 3.5.3. $\beta_{cb}$ is the normalization coefficient that ensures that $cb_{i,j}$ stays within the interval of $0 \leq cb_{i,j} \leq 1$.

#### Integrity Belief

*Integrity belief* $ib_{i,j}$ is a level of confidence in the predictability of future interactions [5]. It is measured as a deviation from the average behavior $cb_{i,j}$. Therefore, $ib_{i,j}$ is calculated as an approximation to the standard deviation of interaction parameters [5].

$$
\begin{aligned}
ib_{i,j} &= \sqrt{\frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} \left( s_{i,j}^{k} \cdot w_{i,j}^{\mu} \cdot f_{i,j}^{\mu} - cb_{i,j} \right)^2} \\
f_{i,j}^{\mu} &= \frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} f_{i,j}^{k} \\
w_{i,j}^{\mu} &= \frac{1}{sh_{i,j}} \sum_{k=1}^{sh_{i,j}} w_{i,j}^{k}
\end{aligned}
\tag{3.3}
$$

It holds that $0 \leq ib_{i,j} \leq 1$. The more consistent behavior peer $j$ has, the lower the $ib_{i,j}$ is. Consistency is a highly desired property as the local peer then has more precise estimates about the future behavior of the remote peer.

### 3.5.3 Interaction Satisfaction

$s_{i,j}^{k}$ is $i$'s satisfaction value with interaction with peer $j$ in window $k$ [5]. We outlined before, that each interaction between two peers is evaluated, $s_{i,j}^{k}$ is a result of this evaluation of a single interaction between peers $i$ and $j$. Because our trust model is generic, the evaluation function can be implemented differently for different data. How did we design it and implemented it for threat intelligence is described in the Section 3.6.

However, even with the computed interaction satisfaction value, not all interactions are the same. Some interactions are more important than others. Moreover, because peers can change their behavior, most recent interactions should be more important than the interactions that happened a long time ago. That is why we include the weight of the interaction and the fading effect.

#### Weight of the Interaction

Because each interaction is different and its importance is different, we have $w_{i,j}^{k}$ that measures the importance [5]. The weight belongs to interval $0 \leq w_{i,j}^{k} \leq 1$ and Fides implements it as a discrete function of interaction

type. For example, the weight of interaction when a remote peer shares the threat intelligence is higher than when the remote peer requests threat intelligence.

#### Fading Effect

Fading effect $f_{i,j}^k$ determines *"how much does the algorithm forget"* as the algorithm prefers most recent interactions over past interactions and thus $f_{i,j}^k$ reduces the weight of the past interactions [5]. $f_{i,j}^k$ is a *non-increasing function* of interaction and time or an index of said interaction in history.

The actual implementation of the fading effect depends on the data the trust model is processing. For example, SORT implements it as a decreasing linear function $f_{i,j}^k = \frac{k}{sh_{i,j}}, 1 \leq k \leq sh_{i,j}$ [5]. However, in our case and after multiple iterations, we decided not to forget the interactions that the model remembers and rather have all interactions with the same impact.

$$f_{i,j}^k = 1 \tag{3.4}$$

The way Fides computes $f_{i,j}^k$ might be changed in the future and implemented as a function of time, we discuss this in more detail as a part of the future work in Section 7.1.

### 3.5.4 Reputation and Recommendations

In order to mitigate the cold start problem outlined in Section 3.2 and in the cases when there are no or few interactions between $i$ and $j$, the algorithm relies on $r_{i,j}$ - *reputation value* [5]. $r_{i,j}$ is the second part of the service trust Equation 3.1 that introduces *remote experience* to the service trust.

The *reputation* value is computed from the *recommendations* received from the remote peers. This value represents what remote peers think about another remote peer. However, this value is calculated by the local peer with respect, to how much it trusts each peer, that provided the recommendation. When the local peer $i$ encounters remote peer $j$ for the first time and it does not have any data about its trustworthiness, $i$ can request recommendations on peer $j$ from $i$'s most trusted peers. We denote a set of remote peers, that provided the recommendations as $T_i$.

#### Requesting a Recommendation

The recommendation system built into Fides cannot be used in every scenario. Because of the sensitive nature of the environment, the trust

model was designed for, there are cases when it is dangerous to ask for recommendations. This is mainly the case when there are *not enough* peers that are *trusted enough*.

SORT requests recommendations every time it encounters a new peer. The set of recommending peers is created by taking all known peers and selecting those that have higher than average service trust. However, those can also be peers with trust as low as 0.001. In a sensitive environment, which the peer-to-peer network of IPS definitely is, we do not want to get recommendations from peers, that have low trust at all. Moreover, given the nature of Slips, we decided to combine a recommendation system based on SORT with static initial trust (3.2.1) and with pre-trusted peers (3.2.2). This approach provides a more robust basis for a trust-sensitive environment and it helps us to mitigate the cold start problem (3.2).

If the peer is part of a pre-trusted organization or it is pre-trusted itself, it inherits the configured reputation - $r_{i,j}$ from the configuration. In this case, Fides does not engage the recommendation protocol at all, because the peer already has reputation $r_{i,j}$ assigned from the configuration and it was *recommended* by the administrator. Moreover, the administrator can choose if this value is *frozen*, or not. *Frozen Service Trust* configuration means, that the peer $j$ has in eyes of $i$ *static service trust* $st_{i,j}$ - it will never change and whatever data peer $j$ sends to $i$ will not influence the $st_{i,j}$. On the other hand, when this configuration is not selected, the peer's service trust is going to change during the time when it communicates with the local instance according to the data and interactions it provides.

In the case where the peer is not pre-trusted, Fides evaluates if it has *enough* well-trusted peers that can be trusted to provide the correct recommendation. This value as well as a number of maximal peers used for recommendation is configurable. In addition, the administrator can enforce that for the recommendation protocol, only the pre-trusted peers or the peers from pre-trusted organizations are used.

#### Recommendation Response

A single recommendation response from peer $z \in T_i$ about giving the recommendation to peer $i$ about peer $j$ contains the following data.

- $cb_{z,j}$, $ib_{z,j}$ - summary of $z$'s interactions with $j$, competence belief and integrity belief

- $sh_{z,j}$ - service history size, number of interactions between $z$ and $j$ - the more interactions they had, then the $z$'s recommendation has more credibility
- $r_{z,j}$ - summary of recommendations that $z$ received on $j$
- $\eta_{z,j}$ - number of peers that provided recommendations for $j$ when $j$ was new to $z$ and their recommendation was used to compute $r_{z,j}$

$cb_{z,j}$, $ib_{z,j}$ are included in the recommendation in order to provide a view on what does $z$ think about $j$. $sh_{z,j}$ and $\eta_{z,j}$ are included to indicate how much experience with $j$ does $z$ actually have. To determine to which extent is the $z$ sure about correctness of $cb_{z,j}$, $ib_{z,j}$, $r_{z,j}$ in the recommendation. And also to protect the $z$'s recommendation trust in $i$'s eyes, if $cb_{z,j}$, $ib_{z,j}$, $r_{z,j}$ values are wrong, because $i$ inspects $sh_{z,j}$ and $\eta_{z,j}$ and does not penalize $z$ that much, if the history size or the number of original recommender are low.

#### Computing Reputation

When the local peer receives all recommendations, it computes the reputation value $r_{i,j}$ as a weighed expected local experience ($ecb_{i,j}$, $eib_{i,j}$ - estimates about competence and integrity) from the remote peers with their remote experience ($er_{i,j}$ - estimate about reputation of said peer).

$$r_{i,j} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \cdot \left( ecb_{i,j} - \frac{1}{2} eib_{i,j} \right) + \left( 1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \right) \cdot er_{i,j} \tag{3.5}$$

The weight, used in the Equation 3.5, is the average of history sizes in all recommendations to $sh_{max}$, maximum interactions history size. We calculate $\mu_{sh}$ as follows.

$$\mu_{sh} = \frac{1}{|T_i|} \sum_{z \in T_i} sh_{z,j} \tag{3.6}$$

Again, we are weighing local experience to remote experience. However, in this case, it is local for the remote peers that provided the recommendations.

### 3.5.5 Remote Local Experience

Similarly, when we compute the service trust in Equation 3.1, we need to get competence and integrity belief. However, while creating reputation value in 3.5 where the values are coming from the remote peers, we are trying to estimate those values received from the network. For that reason, we call

them *estimated competence belief* - $ecb_{i,j}$ and *estimated integrity belief* - $eib_{i,j}$.

#### Estimated Competence Belief

$ecb_{i,j}$ is estimation about competence belief made by $i$ about $j$. This value is computed from the received recommendations in combination with $rt_{i,z}$ - a *recommendation trust* that $i$ has about $z$. Similarly, as for service trust, we have a normalization coefficient $\beta_{ecb}$ that moves the resulting data to the correct interval. It holds that $0 \leq ecb_{i,j} \leq 1$.

$$\begin{aligned} ecb_{i,j} &= \frac{1}{\beta_{ecb}} \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j} \cdot cb_{z,j}) \\ \beta_{ecb} &= \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j}) \end{aligned} \tag{3.7}$$

Recommendation trust $rt_{i,z}$ is described in detail in Section 3.5.7.

#### Estimated Integrity Belief

Following the $ecb_{i,j}$, $eib_{i,j}$ is estimation about the integrity belief made by $i$ about $j$. Equation 3.8 is almost similar, but we use $ib_{z,j}$ instead of $eb_{z,j}$. This means that normalization coefficient $\beta_{eib} = \beta_{ecb}$.

$$\begin{aligned} eib_{i,j} &= \frac{1}{\beta_{eib}} \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j} \cdot ib_{z,j}) \\ \beta_{eib} &= \sum_{z \in T_i} (rt_{i,z} \cdot sh_{z,j}) \end{aligned} \tag{3.8}$$

### 3.5.6 Remote Remote Experience

Going back to Equation 3.5 from Section 3.5.4, we use *estimated reputation value* - $er_{i,j}$. This value represents information that was created by the peers that are remote even for remote peer $j$. In other words, this information came from the *second ring of trust* - from acquaintances of an acquaintance.

$$\begin{aligned} er_{i,j} &= \frac{1}{\beta_{er}} \sum_{z \in T_i} (rt_{i,z} \cdot \eta_{z,j} \cdot r_{z,j}) \\ \beta_{er} &= \sum_{z \in T_i} (rt_{i,z} \cdot \eta z, j) \end{aligned} \tag{3.9}$$

### 3.5.7 Recommendation Trust Metric

Recommendation trust - $rt_{i,z}$ - is another metric that a peer calculates and stores. It expresses how much does $i$ trust that $z$ provides *good recommendations*. Even though one could theoretically use service trust $st_{i,z}$ for this, we have another trust metric because there are peers that can provide very good data (service), but they are surrounded by bad peers or the other way around. This also gives us the ability to have specialized nodes in the network that serves as a peers registry for organizations - a single node that only provides recommendations on peers.

We calculate the recommendation trust in a similar way as the service trust and reputation, but we use recommendation competence belief $rcb_{i,z}$, recommendation integrity belief $rib_{i,z}$ and reputation $r_{i,z}$. This time, we use the weight $rh_{i,z}$, which is the size of the history of the recommendations provided by $z$ to $i$, and $rh_{max}$, the maximal size of said history.

$$rt_{i,z} = \frac{rh_{i,z}}{rh_{max}} \left( rcb_{i,z} - \frac{1}{2} rib_{i,z} \right) + \left( 1 - \frac{rh_{i,z}}{rh_{max}} \right) r_{i,z} \tag{3.10}$$

#### Recommendation Competence and Integrity Belief

Similarly for interactions, we use three different parameters for calculating the $rcb_{i,z}$ and $rib_{i,z}$. We use satisfaction $rs^x_{i,z}$, weight $rw^x_{i,z}$ and the fading effect $rf^x_{i,z}$. The parameters have the same background as described in Section 3.5.3, but in this case, they are connected to recommendations instead of service. We calculate $rcb_{i,z}$ as follows:

$$\begin{aligned} rcb_{i,z} &= \frac{1}{\beta_{rcb}} \sum_{x=1}^{rh_{i,z}} \left( rs^x_{i,z} \cdot rw^x_{i,z} \cdot rf^x_{i,z} \right) \\ \beta_{rcb} &= \sum_{x=1}^{rh_{i,z}} \left( rw^x_{i,z} \cdot rf^x_{i,z} \right) \end{aligned} \tag{3.11}$$

And for recommendation integrity we compute $rib_{i,z}$ as:

$$rib_{i,z} = \sqrt{\frac{1}{rh_{i,z}} \sum_{x=1}^{rh_{i,z}} \left( rs^x_{i,z} \cdot rw^{\mu}_{i,z} \cdot rf^{\mu}_{i,z} - rcb_{i,z} \right)^2} \tag{3.12}$$

One more time, the computational model is trying to approximate average behavior in recommendations - $rcb_{i,z}$ - and then the deviation from such behavior - $rib_{i,z}$.

Fading effect $rf^x_{i,z}$ has similar properties as the fading effect for service trust described in Section 3.5.3. It is a non-increasing function of a number of recommendations or a time. For the recommendations, Fides implements it exactly the same as for the service interactions.

$$rf^x_{i,z} = 1 \tag{3.13}$$

**Evaluating Received Recommendation**

As outlined in Section 3.5.7, in order to evaluate a particular recommendation from remote peer $z$, we have satisfaction, weight, and the fading effect. We calculate the recommendation satisfaction $rs^x_{i,z}$ by comparing values from $z$'s recommendation $r_{z,j}$, $cb_{z,j}$, $ib_{z,j}$, with values that are the results of the the recommendation algorithm. In other words, we compare each recommendation, with the aggregated values - $er_{i,j}$, $ecb_{i,j}$ and $eib_{i,j}$. This gives us an estimate of how off was the peer $z$'s recommendation from the final result of the recommendation algorithm.

$$\begin{aligned} rs^x_{i,z} = \frac{1}{3}\Bigg[&\left(1 - \frac{|r_{z,j} - er_{i,j}|}{er_{i,j}}\right) + \\ &\left(1 - \frac{|cb_{z,j} - ecb_{i,j}|}{ecb_{i,j}}\right) + \\ &\left(1 - \frac{|ib_{z,j} - eib_{i,j}|}{eib_{i,j}}\right)\Bigg] \end{aligned} \tag{3.14}$$

We calculate the weight of recommendation $rw^x_{i,z}$ as a weighed sum of the proportion of the size of the service history between $z$ and $j$ with maximal service history size. And a number of peers that provided the initial reputations $\eta_{z,j}$ divided by a maximal number of possible recommending peers.

$$rw^x_{i,z} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \cdot \frac{sh_{z,j}}{sh_{max}} + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}}\right) \cdot \frac{\eta_{z,j}}{\eta_{max}} \tag{3.15}$$

## 3.6 Interaction Evaluation Strategies

In order to determine which remote peers are providing valuable data and which peers are not, the local peer needs to be able to evaluate each interaction it had with the remote peer. In general, there are two options for how to approach this, (i) by designing an evaluation that is protocol-aware (i.e. it understands the protocol and the data that the two peers shared ); or (ii) by having an evaluation function that does not need to understand the protocol and can be used for any type of data.

We choose to implement both approaches and they are described in the following sections. In order to evaluate which strategy is better in what scenarios, we designed and run many simulations - their results are described in Chapter 5. We will use the notation from the table (3.2) when referring to peers and their interactions.

| | |
|---|---|
| $i$ | local peer |
| $j$ | remote peer |
| $T$ | target of network intelligence, domain or IP address |
| $k$ | evaluation window |
| $s^k_{i,j}$ | $i$'s satisfaction value with interaction with peer $j$ in window $k$ |
| $S^k_{j,T}$ | score computed by the peer $j$ about target $T$ in window $k$ |
| $C^k_{j,T}$ | confidence, how much is the score correct |
| $S^k_T$ | aggregated score from all threat intelligence reports in window $w$ for target $T$ |
| $C^k_T$ | aggregated confidence |

Table 3.2: Interactions Symbols

### 3.6.1 EvenTIEvaluation

This strategy does not need to understand the underlying data, its semantics, or its structure. It is a naive approach when the trust model uses the same satisfaction value for all data it received. It does not check if the data make sense and assigns all peers the same satisfaction value $s^k_{i,j}$. The value itself is loaded from the configuration provided by the administrator. We denote it as $CSS$. The idea behind this algorithm is that when the

peers are interacting for a long time or have more interactions, they are more trustworthy.

$$s_{i,j}^{k} = CSS \tag{3.16}$$

This approach is used, for example, by the botnet **Sality** or by the **Dovecot** trust model. Fides implements it as an *EvenTIEvaluation* strategy with configurable satisfaction value and the administrator can use this strategy if they see it as the most optimal.

The disadvantage of this approach is that we do not penalize remote peers when they provide wrong data, because the evaluation method does not care nor understand the underlying data. Because of that and in a case when the adversary gains the service trust of the model by following the protocol for a longer time, it may significantly influence the aggregated score as the adversary has higher trust than other remote peers. If this happens, there is no way to automatically downgrade the adversary's service trust.

### 3.6.2 DistanceBasedTIEvaluation

Because Fides is designed for sharing and aggregating threat intelligence, and understands the protocol that is being used, we can use this and penalize the peers that are providing the local peer with incorrect data. The interaction evaluation is performed at the end of the threat intelligence sharing process, where at that point, Fides already aggregated the data and calculated the aggregated network score and confidence. Thus, we can use the aggregated values as a baseline. Then we compare them against each remote peer's threat intelligence we received. This evaluation strategy is implemented in the Fides as a *DistanceBasedTIEvaluation*.

Suppose, that remote peer $j$ provided data about target $T$ to local peer $i$ in window $k$. The provided data consist of score and confidence ($S_{j,T}^{k}$, $C_{j,T}^{k}$). Where score, $-1 \leq S_{j,T}^{k} \leq 1$, indicates if the target is malicious ($-1$) or begin (1). The confidence $0 \leq C_{j,T}^{k} \leq 1$ on the other hand indicates, how certain is the peer about its assessment of $S_{j,T}^{k}$.

In order to evaluate the interaction between the local peer $i$ and the remote peer $j$ we need to compute the satisfaction value $s_{i,j}^{k}$. It holds that $0 \leq s_{i,j}^{k} \leq 1$ where 1 means peer $i$ was satisfied with the interaction.

$$s_{i,j}^{k} = \left(1 - \frac{|S_{T}^{k} - S_{j,T}^{k}|}{2} \cdot C_{j,T}^{k}\right) \cdot C_{T}^{k} \tag{3.17}$$

Where $S_{T}^{k}$ is the final score aggregated across the reports from the peers, $C_{T}^{k}$ is aggregated confidence.

The problem with this evaluation algorithm is the situations where the aggregated confidence $C_T^k$ is close to 0. In this case, the algorithm will penalize all peers for providing any threat intelligence as the final $s_{i,j}^k$ is close to 0. Another issue with this approach is that when a single honest peer has unique information about an IP address or domain, which is significantly different than what other peers have, it is automatically penalized for not sharing the same opinion as the other peers. However, if the peer is trusted enough, it has a higher impact on the aggregated value and it is not penalized too much.

### 3.6.3 ThresholdTIEvaluation

In order to compensate for the low confidence, $C_T^k$ and in order not to penalize all peers in the algorithm explained in Section 3.6.2, this evaluation strategy considers $C_T^k$ value and employs the $DistanceBasedTIEvaluation$ only when $C_T^k$ is *"high enough"*. In this case *"high enough"* means higher than a value $CT$, configured by the Slips administrator. In a case when $C_T^k < CT$, the algorithm fallbacks to using $EvenTIEvaluation$, because it is not possible to distinguish between *"good"* and *"bad"* network intelligence due to low confidence in the decision. This strategy is implemented in Fides under the name $ThresholdTIEvaluation$. What should be the correct value for $CT$ from the configuration is subject to evaluation in the simulations in Chapter 5.

**Algorithm 1** $ThresholdTIEvaluation$

```
1: CT ← configuration      ▷ configuration provided by the administrator
2: if C_T^k < CT then
3:     s_{i,j}^k ← EvenTIEvaluation()
4: else
5:     s_{i,j}^k ← DistanceBasedTIEvaluation()
6: end if
```

### 3.6.4 LocalCompareTIEvaluation

This approach uses a similar equation for computing the satisfaction value outlined in Section 3.6.2. However, the input is different. Instead of comparing the remote peer's ($j$ ) threat intelligence ($S_{j,T}^k$, $C_{j,T}^k$) to the aggregated intelligence ($S_T^k$, $C_T^k$), we compare it to the threat intelligence of the local ($i$) Slips instance - ($S_{i,T}^k$, $C_{i,T}^k$). Thus the evaluation is the following:

$$s_{i,j}^k = \left(1 - \frac{|S_{i,T}^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_{i,T}^k \tag{3.18}$$

This approach is useful when the local peer has enough information about the target but wants to verify the behavior of the remote peers. To determine whether they are sending data that are somewhat correct. This strategy is implemented in Fides with name *LocalCompareTIEvaluation*.

### 3.6.5 WeighedDistanceToLocalTIEvaluation

Another implemented strategy combines Sections 3.6.2 and 3.6.4 and mixes them using a weight $w$, provided in the configuration. This is a good approach when Slips or the network has a lot of data on the target. It evaluates interactions with what the local instance thinks and what the network opinion is. What the correct balance is, is subject to simulations and configuration by the administrator.

$$\begin{aligned} s_{i,j}^k = w \cdot \left(1 - \frac{|S_{i,T}^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_{i,T}^k + \\ (1-w) \cdot \left(1 - \frac{|S_T^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k\right) \cdot C_T^k \end{aligned} \tag{3.19}$$

In Fides this is implemented as the $WeighedDistanceToLocalTIEvaluation$.

### 3.6.6 MaxConfidenceTIEvaluation

As pointed out in Section 3.6.2, $DistanceBasedTIEvaluation$ strategy performs poorly if the confidence of aggregated data is low. The strategy $ThresholdTIEvaluation$ is solving the problem by introducing a threshold that has to be configured by the administrator. This is not optimal and we wanted to have a strategy that does not require configuration for its own behavior, thus we introduce a new strategy implemented under the name $MaxConfidenceTIEvaluation$.

The goal of this strategy is to evaluate the received data with as much confidence as possible while having a fully automatic process without the administrator's configuration. In order to do that, we combine all previous strategies into one, where we utilize all available information into a single $s_{i,j}^k$ value.

We introduce new variables here - $w_0, w_1, w_2$ - which are essentially weights of the particular strategies. These weights are based on the confidence the strategy has in its own decision. Note, that there is a hierarchy,

and the order matters. In our case we decided to prefer decisions coming from strategy $DistanceBasedTIEvaluation$ (Section 3.6.2), then we add data from the $LocalCompareTIEvaluation$ (Section 3.6.4) and if the final decision still does not have the confidence of 1, we add the static value configured by the administrator (noted as $CSS$). The last part - $CSS$ - simulates the static strategy described in strategy $EvenTIEvaluation$ (Section 3.6.1) and is set by the Slips administrator.

$$\begin{aligned} w_0 &= C_T^k \\ w_1 &= min(1 - C_T^k, C_{i,T}^k) \\ w_2 &= 1 - w_0 - w_1 \end{aligned} \tag{3.20}$$

The weights $w_0, w_1, w_2$ in the Equation 3.20, are designed to gather as much confidence as possible. $w_0$ is the confidence of the aggregated network data, essentially saying how much is the network sure about the given score. $w_1$ is the confidence coming from the local IPS and the $w_2$ is the remaining confidence to 1.

We use the calculations from Sections 3.6.2, 3.6.4 and 3.6.1 multiplied by the weights $w_0$, $w_1$ and $w_2$ respectively.

$$\begin{aligned} s_{i,j}^k = \\ & w_0 \cdot \left[ \left( 1 - \frac{|S_T^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k \right) \cdot C_T^k \right] + \\ & w_1 \cdot \left[ \left( 1 - \frac{|S_{i,T}^k - S_{j,T}^k|}{2} \cdot C_{j,T}^k \right) \cdot C_{i,T}^k \right] + \\ & w_2 \cdot CSS \end{aligned} \tag{3.21}$$

$MaxConfidenceTIEvaluation$ is the implementation name of this strategy in Fides.

All strategies and their short description are part of the diagram 3.3 that we include for clarity.

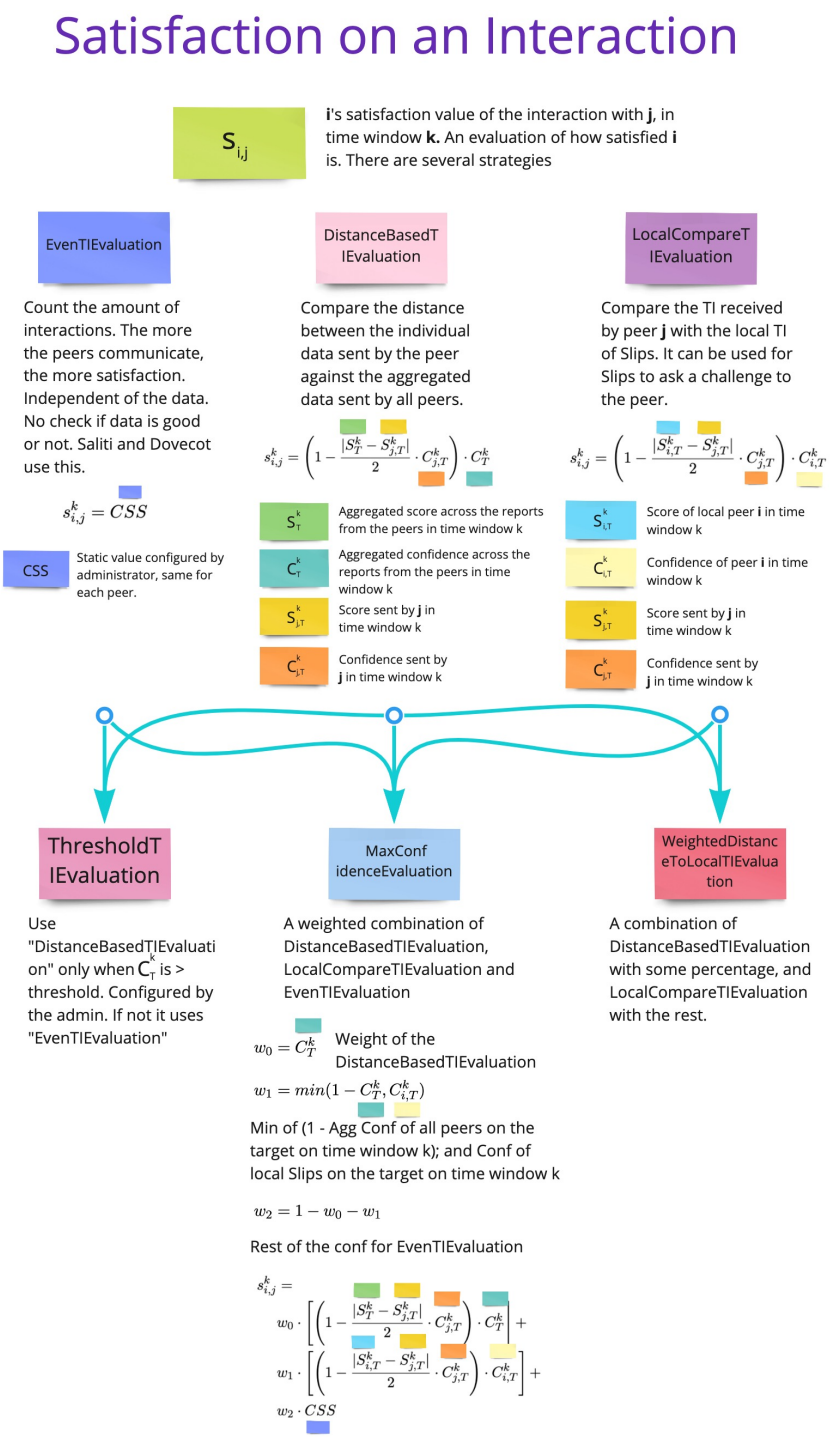


Figure 3.3: Overview of the interaction evaluation methods with their description and used values.

## 3.7 Network Intelligence Aggregation

Fides is a trust model designed for global peer-to-peer networks of Slips instances. It is designed to support Slips in detecting malicious actors on the network and enables threat intelligence sharing between peers of Slips instances. Because Slips was designed to be as modular as possible, Fides is effectively running as a module that provides aggregated threat intelligence to Slips. In other words, Fides provides a view of what the network thinks about some threat intelligence target. This is necessary so Slips can have a unique *view* of the network on a specific Threat Intelligence. Fides needs to aggregate elements of threat intelligence from remote peers into a single value that is then presented to Slips.

Fides needs to say that some reports are better than others, based on the service trust the local peer has in the remote peer (previously computed as $st_{i,j}^k$). Thus Fides needs to weigh every report based on this trust and come up with an aggregated score $S_T^k$. Apart from the aggregated score, Fides needs to compute the aggregated confidence $C_T^k$ that expresses how confident $i$ is about the aggregated score $S_T^k$ that was computed in the previous step.

Once aggregated, the computed score and confidence ($S_T^k$, $C_T^k$) are sent to Slips to report data on target $T$. Apart from sending to Slips, these same values can be also used to evaluate the interaction of the remote peers, depending on the selected interaction evaluation strategy. We describe this more in depth in Section 3.6.

We designed and implemented two different functions for aggregating threat intelligence and computing $S_T^k$ alongside with $C_T^k$. Both of them are implemented in Fides under their respective names and which method performs better under what circumstances is a subject of the experiments in Chapter 5.

### 3.7.1 AverageConfidenceTIAggregation

In this method, the aggregated score $S_T^k$ is the sum of $S_{j,T}^k$, which is the score sent by each peer $j$ about target $T$ in time window $k$; weighed with the normalized service trust that $i$ computed for peer $j$, denoted $wst_{i,j}^k$. The sum is done over the set of remote peers that provided a report to $i$ for $T$ in time window $k$, denoted $R_{i,T}^k$. We calculate it in Equation 3.22.

$$S_T^k = \sum_{j \in R_{i,T}^k} wst_{i,j}^k \cdot S_{j,T}^k \tag{3.22}$$

The normalized service trust $wst_{i,j}^k$ used as weight is computed as:

$$wst_{i,j}^{k} = \frac{1}{\sum_{j \in R_{i,T}^{k}} st_{i,j}^{k}} \cdot st_{i,j}^{k} \tag{3.23}$$

Equation 3.23 estimates the percentage that the service trust on $j$ $st_{i,j}^{k}$ has relative to the total sum of service trust received by $i$ for all peers, for this target $T$, in time window $k$.

We compute the aggregated confidence $C_{T}^{k}$ for this strategy as:

$$C_{T}^{k} = \frac{1}{|R_{i,T}^{k}|} \cdot \sum_{j \in R_{i,T}^{k}} st_{i,j}^{k} \cdot C_{j,T}^{k} \tag{3.24}$$

Which is an average over all the peers that sent to $i$ a report on $T$ in time window $k$, of the weighted confidence sent by peer $j$ on target $T$ on time window $k$. The weight is done by the service trust that $i$ has on $j$ on time window $k$.

### 3.7.2 WeightedAverageConfidenceTIAggregation

This strategy uses Equation 3.22 to compute the aggregated score $S_{T}^{k}$ similarly to the $AverageConfidenceTIAggregation$ in Section 3.7.1. However, the way how this strategy calculates $C_{T}^{k}$ is different. Instead of using the service trust $st_{i,j}^{k}$ to determine the correct trust in the confidence $C_{j,T}^{k}$ submitted by peer $j$ and then diving it by the number of peers, it uses the normalized service trust $wst_{i,j}^{k}$ computed in Equation 3.23 that already contains the weight of the peers in the final decision.

$$C_{T}^{k} = \cdot \sum_{j \in R_{i,T}^{k}} wst_{i,j}^{k} \cdot C_{j,T}^{k} \tag{3.25}$$

# Chapter 4

# Architecture

Slips is a modular software. Each module is designed to perform a specific detection in the network traffic [16]. The modules can also be used to extend Slips with any additional functionality directly. Fides was designed for seamless interoperability with Slips, and in addition to the generic trust model, we developed the Fides module for Slips. In this chapter, we describe the architecture of Fides and how it interacts with the network and with Slips.

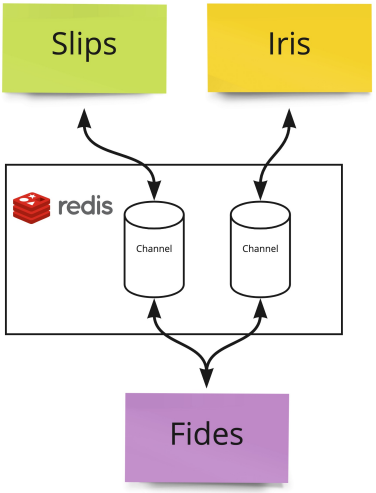


Figure 4.1: Fides high-level architecture that visualizes communication between Fides, Iris and Slips.

From the high-level perspective (see Figures 4.1 and 4.2), the trust model Fides communicates with two different systems - Slips [16] and the network layer Iris [28]. Fides manages the trust relationships in the network, aggregates threat intelligence data, and communicates with Slips. The commu-

nication with the remote peers in the network is facilitated by Iris. Slips then produces and consumes the threat intelligence and defends the network against intruders.

Fides exposes and consumes an API [3] built using the Redis channels for both parts (Figure 4.1). The messages and API calls are consumed using the JSON [21] data format.

Redis is an in-memory data structure store that supports asynchronous channels and a publish-subscribe model [25]. Moreover, it can also persist data on disk if required. We chose to employ Redis channels as the medium that allows communication between the Iris and Fides and allows them to use their respective APIs because Slips already uses Redis for its internal communication between modules. It brings no additional overhead to run Fides with its network layer.

## 4.1 Fides & Network Access

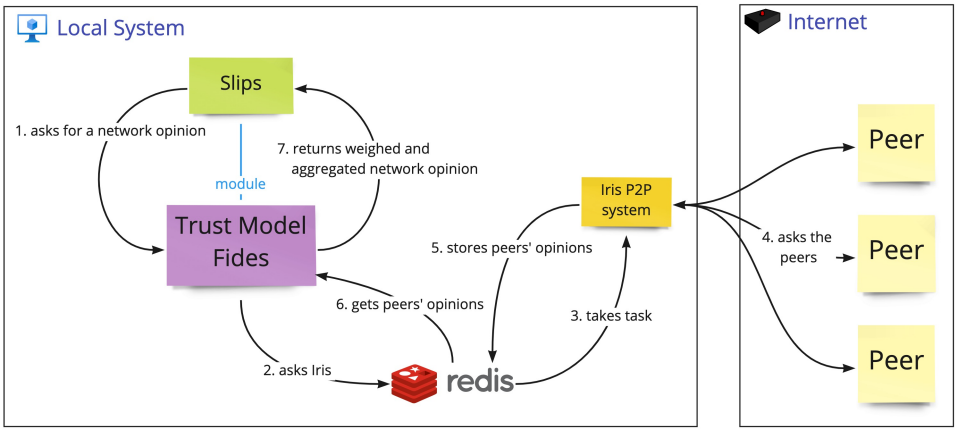


Figure 4.2: High-level overview of communication between Fides, Iris and Slips including examples of messages that sends to each other.

Fides itself is a trust model and it does not interact with the network directly, but rather it exposes an API that can be used either to receive the information from the network or for sending the requests back to the network. Thanks to this design, where all business logic is separated from the network layer, Fides is highly modular and does not depend on the network layer implementation. The network layer Iris then performs all data transfers and facilitates all communications with the remote peers. It also facilitates finding new peers and ensuring that all requests from Fides are dispatched to the correct recipients. In the eyes of Fides, the network layer is a *black box* and it does not need to know how the network layer is implemented. See Figure 4.2 for a high-level overview of the communication.

The network layer, **Iris**, was developed by Bc. Martin Řepa in [28] where Řepa describes how Iris works in detail and what protocols are used to safely deliver necessary information and messages between the instances of Fides.

## 4.2 Implementation

Because Fides was designed to integrate with Slips, we were constrained by the Slips implementation [16] and for that reason we implemented Fides in Python [14] version 3.8.

The code is versioned by Git [7] and published on GitHub [17] in the repository *github.com/stratosphereips/fides* [13]. We chose to use Conda [10] for managing the dependencies and Python versions. Fides package structure is then following:

```
fides
├── evaluation
│   ├── recommendation
│   └── service
├── messaging
├── model
├── persistence
├── protocols
└── utils
```

The implementation is split into multiple Python packages. The package *evaluation* contains all necessary algorithms for any data evaluation. All interaction evaluation strategies (Section 3.6) can be found in the file *ti_evaluation.py*, all methods for aggregating threat intelligence (Section 3.7) in the file *ti_aggregation.py*. The *evaluation.recommendation* package contains all computations that are related to the recommendations and reputation as described in Section 3.5.7. Similarly, the package *evaluation.service* stores algorithm for computing the service trust described in Section 3.5.1. The package *messaging* is a connector that allows Fides to send and receive data from the Redis and thus from the Iris. There is a *network_bridge.py* which allows Fides to send a message to Iris and Slips via message queue and *message_handler.py* which on the other hand is designed for receiving, message syntax validation and parsing of messages. The package *model* stores all data classes that represent the data Fides operate with. The next package *persistence* then contains the implementation of the Fides's persistence layer which we describe more in Section 4.2.2. The next package *protocols* contains all data processing and Fides's decision process. The last package is *utils* where we store our implementation of logging.

In the end, we wrote 6159 lines of code in total where the trust model Fides itself accounts for 2691 lines. The rest of the implementation is related to the unit tests, integration tests, simulation framework, simulations and

the Slips module. We also set up a simple continuous integration pipeline for executing the unit and integration tests in GitHub.

### 4.2.1 Configuration

Our trust model contains many different configuration options either related to the computational model itself or to the data persistence or identity of the local peer.

Computational model settings are for example the threat intelligence aggregation methods described in Section 3.7 or the interaction evaluation strategy from Section 3.6. As the trust model needs only a single method, the administrator needs to define which one of these functions should be used.

The configuration itself is in a single YAML [32] file that is in the repository root in *fides.conf.yml* [13]. This file is loaded and validated during the trust model startup and is used to provide all possible configuration options for the trust model.

### 4.2.2 Persistence

Fides stores trust-related data such as past interactions, cached network opinions, service trust, recommendations, etc. inside the database. The database layer was implemented as an abstract part and can be easily replaced in the future. As of now, we have two different implementations. An in-memory database and a database that stores data in Redis. However, thanks to its modularity, different persistence solutions can be easily implemented.

### 4.2.3 Data Filtering

Part of the configuration is the section about data confidentiality and sharing of the threat intelligence with other peers. Fides allows operators to choose what threat intelligence will be shared, when, and to whom.

For example, if the threat intelligence received from the local Slips instance contains a *confidentiality level*, the operator can enforce that only peers with *high* service trust will receive this threat intelligence when they ask for it.

The confidentiality level, $cl$, $0 \leq cl \leq 1$, defines how sensitive or confidential the threat intelligence is where $cl = 0$ means public information that can be shared with anybody and $cl = 1$ secret information that should not be shared at all.

The Fides administrator can then specify what service trust $st$ is required for what confidentiality level, $cl$, in the configuration (Section 4.2.1) . If this configuration is in place, whenever a remote peer ($j$) asks for threat intelligence and the local ($i$) Slips has the requested threat intelligence, Fides verifies that $st_{i,j} \geq cl$ before providing the intelligence to the remote peer.

This mechanism ensures that Slips does not leak information that is private or somehow more sensitive than the others.

# Chapter 5

# Experiments

We designed a single and comprehensive experiment that simulates a real-world usage of Fides. This chapter describes how we set up an environment that allows us to run experiments and simulate real-world situations in the peer-to-peer network where the peers communicate and share threat intelligence.

In Section 5.1 we describe how we sample threat intelligence shared by the peers. In the following Section 5.2, we list different types of peers in the network, what is their goal, and how they behave. Section 5.3 then describes how we designed the environment and what are the inputs for the simulation itself. The last Section 5.4 presents how we evaluate each scenario and explains the vital simulation indicators.

## 5.1 Sampling Threat Intelligence

Threat intelligence, which is being shared on the peer-to-peer network and is aggregated by Fides, is generated inside Slips by various modules. Each module provides a score on its own and Slips aggregates these evaluations into a single value. This means that threat intelligence is computed as a sum of independent random variables and that tends to follow the normal distribution. For that reason, we sample threat intelligence values from the normal distribution.

As peers have different behavior, we will sample the threat intelligence provided by them every time when they are asked for it. We will characterize the peer's behavior by the threat intelligence it provides, with respect to the baseline, and the ground truth of the target being benign or malicious.

As described in the previous chapters, threat intelligence consists of a *score* and the *confidence* in that score. We use the notation $\mu_s$ for the mean

threat intelligence score and $\sigma_s$ for the standard deviation of the score. Similarly, we use $\mu_c$ for mean confidence and $\sigma_c$ for the standard deviation of the confidence.

Fides also in some cases employs a recommendation protocol, so in the simulations, the peers might be asked to provide recommendations about other peers. They will follow their behavioral strategy when providing the data. Recall the recommendation description from Section 3.2.3. A single recommendation response contains $cb_{k,j}$, $ib_{k,j}$, $sh_{k,j}$, $r_{k,j}$ and $\eta_{k,j}$. We will be sampling those from the normal distribution as well with the corresponding pairs of $(\mu_{cb}, \sigma_{cb})$, $(\mu_{ib}, \sigma_{ib})$, $(\mu_{sh}, \sigma_{sh})$, $(\mu_r, \sigma_r)$ and $(\mu_\eta, \sigma_\eta)$. Every peer will provide recommendations based on his behavioral strategy with respect to the ground truth.

## 5.2 Peer's Behavioral Patterns

For the sake of experiments, we chose the behavior of each peer in the simulated network. We identified multiple different behavioral patterns for the benign as well as for the malicious peers. Every behavior is different and is defined by the $(\mu, \sigma)$ for every data we sample and by the intent the peer has in the network. Most of the behavior depends on the baseline, which is the ground truth for any target in the system, if it is benign or malicious. We note the baseline score as $S_B \in \{-1, 1\}$, where $S_B = -1$ means that the target is malicious and $S_B = 1$ means that the target is benign.

### 5.2.1 Confident Correct Peer

This behavior corresponds to an honest peer that provides correct data according to the baseline. Meaning, that if the target (domain/IP address that we have threat intelligence for) is benign, the peer with *confident correct* behavior will provide threat intelligence that says that the target is benign. Moreover, the peer will provide the data with high confidence.

The very same thing applies to the situation when this peer is asked to provide a recommendation for any other peer. The provided recommendation will reflect the real behavior of said peer and it will indicate high confidence in the recommendation. This peer has the *ideal* behavior as its data are useful and correct. Table 5.1 describes the data used for sampling the threat intelligence this peer provides.

| type | notation | $\mu$ | $\sigma$ |
|---|---|---|---|
| score | $\mu_s^{cc}$ | $S_B \cdot 0.9$ | 0.1 |
| confidence | $\mu_c^{cc}$ | 0.9 | 0.1 |

Table 5.1: Confident Correct Behavior

### 5.2.2 Uncertain Peer

This behavior simulates peers that do not have enough information to provide reasonably good data, but they are benign and honest with their behavior. The peer can provide essentially any score but with very low confidence in said score. That is why the $\mu_s^{up}$ is quite high whereas the mean for this behavior is 0.

| type | notation | $\mu$ | $\sigma$ |
|---|---|---|---|
| score | $\mu_s^{up}$ | 0.0 | 0.8 |
| confidence | $\mu_c^{up}$ | 0.3 | 0.2 |

Table 5.2: Uncertain Peer Behavior

### 5.2.3 Confident Incorrect

The peer with this behavior is confident about their data and the threat intelligence, but their threat intelligence is wrong. However, this peer is still benign and is making honest mistakes. This strategy simulates peers that were not attacked by a malicious device and they consider it benign because they do not have any information indicating malicious intent. Thus, whenever the peer is asked to provide threat intelligence, it responds with a score that is opposite of the baseline and with a high confidence value.

| type | notation | $\mu$ | $\sigma$ |
|---|---|---|---|
| score | $\mu_s^{ci}$ | $-S_B \cdot 0.8$ | 0.2 |
| confidence | $\mu_c^{ci}$ | 0.8 | 0.2 |

Table 5.3: Confident Incorrect Behavior

### 5.2.4 Malicious Peer

The malicious peer is going to provide wrong threat intelligence intentionally to achieve their goal of influencing the trust decisions of the local peer. The sampling data are the same as for the *confident incorrect* (Section 5.3) behavior, but the difference is that the malicious peer is providing

misleading data intentionally. Moreover, intelligent malicious peer knows, that it their incentive is to gain the service trust at the beginning in order to more impact the decisions of the trust model in the later stages. We simulate this by introducing a grace period when the malicious peer does not lie, but rather behaves like any other normal peer. This period then allows the malicious peers to gain the initial trust. After that period, they start to lie and thanks to the initial trust, they can influence the Fides's decisions with a larger impact.

As stated before, this behavior simulates knowledgeable adversaries that are able to follow the Fides's protocol and their goal is to influence decisions of the local trust model. The adversaries can be either trying to *bad-mouth* or provide *unfair praises*. In our case, it does not matter why they do that, but rather the fact, that they do that intentionally and that they are providing the opposite of the baseline score with the high confidence.

We decide to design an attacker, that is trying to hide in the data and it is not providing score $\{-1, 1\}$ with the confidence of 1 all the time, but rather uses a distribution that is close to these values. The reason is that if the model sees that there is a peer that provides $\{-1, 1\}$ with high confidence all the time, it would be very easy to detect and penalize this behavior.

| type | notation | $\mu$ | $\sigma$ |
|---|---|---|---|
| score | $\mu_s^m$ | $-S_B \cdot 0.9$ | 0.1 |
| confidence | $\mu_c^m$ | 0.9 | 0.1 |

Table 5.4: Malicious Behavior

## 5.3 Environment Simulation

It is important in the simulations to also simulate time. This is because the trust model depends on when peers join the network and when they decided to lie or not. It is also because new peers are subject to recommendation requests, but only when they are new.

Time in the simulations is measured in *clicks*. The local instance of Fides performs a single action and receives responses from other peers in the network in exactly *one click*. For example, this is the series of events that happen in a *single click*. Fides asks the network for threat intelligence, receives the responses, aggregates network opinion, and evaluates the interactions with peers. Another series of events happening in a *single click* is the actions of recommendation protocol: a new peer joins the network, Fides asks for the recommendation for a new peer, collects the responses, com-

putes the reputation, and evaluates the received recommendations. What is the relation between real time and the *clicks* depends solely on the network layer, mostly on the speed of messages convergence described in-depth in [28].

In order to simulate the environment, we have multiple parameters that correspond to the expectations of how does the peer-to-peer network looks like. We start with the **number of peers in network** that simulates the size of the network and how many different peers can appear during the whole simulation.

The network anatomy is another parameter for the simulation, where we define what **percentage of peers** are using **what strategy** that was described in the Section 5.2. In other words, how many peers in the network are adversarial and how many of them are benign.

Another parameter is the **number of targets** (IP addresses and DNS domains) that will be used when Fides will be requesting the network threat intelligence. For each target, we know the label (malicious and benign) and we will be sampling threat intelligence that came from the local Slips instance. The local threat intelligence will be sampled from the parameters of one of the strategies described in Section 5.2 - confident correct, uncertain, or confident incorrect - which is yet another parameter that describes how the local Slips instances behave.

For each remote peer, we select one of the behaviors from Section 5.2 and **the number of peers for each behavior** is determined by the configuration of the simulation. We also determine if the peer is pre-trusted or if it is a member of the pre-trusted organization. The **percentage of pre-trusted peers** is again configurable. Next, we determine the **time** (in *clicks*), when the peer is going to **join the network**. This allows us to evaluate the recommendation part of Fides, because if the peer joins late, Fides requests recommendations from the other peers which can lead to further problems if the recommending peers are adversarial.

If the strategy selected in the previous step is malicious (with its behavior as described in Section 5.2.4), we determine for **how many targets** is the peer going to **lie** about. This allows us to also simulate a highly advanced attacker that lies only selectively for the targets that they control. It is not rational for the attacker to lie about targets that are not known to them as they do not gain any advantage from that. On the contrary, if they do not lie, they gain more trust which they can use to further influence the local decisions.

The last simulation parameter is how many *clicks* are left at the beginning, for the pears to gain the initial trust. This means that in that initial

time period, malicious peers will behave like confident ones (Section 5.2.1), in order to gain initial trust, and after that, they will switch to their own malicious behavior (Section 5.2.4). This allows us to evaluate how fast is the trust model able to determine that the peer, with the existing service trust, is malicious.

## 5.4 Experiments Evaluation

An important part of the experiments is how to evaluate what Fides setup *(interaction evaluations, threat intelligence aggregation)* is better in which scenario. We will be measuring two performance metrics that are relevant for each situation.

### 5.4.1 Target Detection Performance Metric

This first metric, *tdp*, measures performance of the target detection. We compute *tdp* in Equation 5.1 as an average distance between the ground truth for the target and the final detection made by Fides at the end of the simulation. We use the following notation: $\tau$ is the set of targets in the simulation, $GS_T$ is the ground truth score of the target, $S_T^{k_{max}}$ is then the aggregated score (Section 3.7) for the given target computed by Fides at the end of the simulation.

$$tdp = \frac{1}{|\tau|} \sum_{T \in \tau} \left| GS_T - S_T^{k_{max}} \right| \tag{5.1}$$

This metric provides information on how good Fides was in computing the score (malicious / benign) for some target. It holds that $0 \leq tdp \leq 2$ where 0 is the best detection and 2 is the worst detection. Moreover, if $tdp \leq 1$ the Fides was on average able identify all targets correctly.

### 5.4.2 Peer's Behavior Detection Performance Metric

The peer's behavior detection performance metric *pbdp* measures how close was the trust model's service trust value for the remote peer to the peer's real behavior in the simulation. We measure it in 5.2 as an average distance between computed service trust and the ground truth behavior of the peer in the simulation.

$$pbdp = \frac{1}{|P|} \sum_{j \in P} \left| \bar{b_j} - st_{i,j}^{k_{max}} \right| \tag{5.2}$$

$P$ is the set of remote peers in the simulation, $st_{i,j}^{k_{max}}$ is the service trust that the local trust model $(i)$ had for the remote peer $(j)$ at the end of the simulation. $\bar{b_j}$ is then the ground truth behavior of the remote peer and we compute it in the Equation 5.3.

$$\bar{b_j} = \frac{1 + shift \cdot \mu_s^b}{2} \tag{5.3}$$

Recall the description of the peers' behaviors from the Section 5.2, where each peer's behavior $b$ had $\mu_s^b$ that was used during threat intelligence sampling. Because the sampled score is $[-1, 1]$ and service trust $[0, 1]$, we can not use the $\mu_s^b$ directly, but we need to scale it to the correct interval. Moreover, as malicious and incorrect peers do have $\mu_s^b$ on the opposite scale that the ground truth is, we need to shift it before normalizing it. For that reason, $shift = -1$ for malicious and incorrect peers and $shift = 1$ for confident correct, and uncertain behaviors and thus the Equation 5.3.

### 5.4.3 Environment Hardness

In order to be able measure how *hard* it is for Fides to operate in some environment, we designed the environment hardness variable $eh$. It holds that $0 \leq eh \leq 10$ and the higher the value is, the easier is for Fides to operate in such environment as there are more confident correct peers that provide correct threat intelligence and recommendations. On the contrary, the lower the $eh$ is, the harder it is for Fides to operate as there are more byzantine peers.

$$eh = 10 \cdot \frac{|P_{CC}|}{|P|} + \frac{|P_{UP}|}{|P|} \tag{5.4}$$

Where $P_{CC}$ is a set of peers in simulation that behave like a confident correct (Section 5.2.1) peer and $P_{UP}$ that behave like an uncertain peer (Section 5.2.2).

## 5.5 Simulation Execution

The simulations and experiments were designed to evaluate the trust model in multiple ways and environments. In order to run arbitrary scenarios, we developed a framework, that allows us to simulate virtually any environment with various combinations of Fides configuration.

Unfortunately, it is not possible to run and evaluate all possible scenarios, as there are 14 different sets of parameters that can have many different values. This leads to a combinatorial explosion and therefore we were unable to cover all possible existing scenarios. However, alongside the Fides implementation, we published the simulation framework as well, so anybody can simulate their preferred scenarios.

In the next Chapter 6 we describe how we evaluated the experiments and what we learned about the trust model behavior in various environments with focus on the evaluation of Fides's resilience.

# Chapter 6

# Results

This chapter presents and evaluates the results of the simulations that were designed in the previous Chapter 5. Since there are too many different scenarios to evaluate each setup thoroughly, we mainly focus on evaluating Fides under specific conditions that verify its resilience. These conditions are the more important for the administrator, such as situations with many byzantine peers.

The evaluation focus on finding a scenario where there are as many adversarial peers as possible, and Fides is still able to guarantee that it can come up with the correct target score. This is worst case scenario that every trust model should be evaluated under, since there is no point in evaluating a situation only with good and trusted peers.

However, since the reader may be interested in trying different scenarios, we developed and published a simulation framework [13] where anyone can verify and simulate any scenario they are interested in.

Note that all figures in this chapter can be replicated by re-running the simulation Python code in $simulations/cases/figures$ [13]. The graphs may differ slightly because the threat intelligence and recommendations are sampled from a probability distribution as described in Section 5.1, but the overall results should be the same.

## 6.1 General Overview of a Single Simulation

To understand the results of our simulation we first need describe how does the outcome of a simulation looks like, such as the example shown in Figure 6.1. The simulation framework provides this graph for each possible simulation.

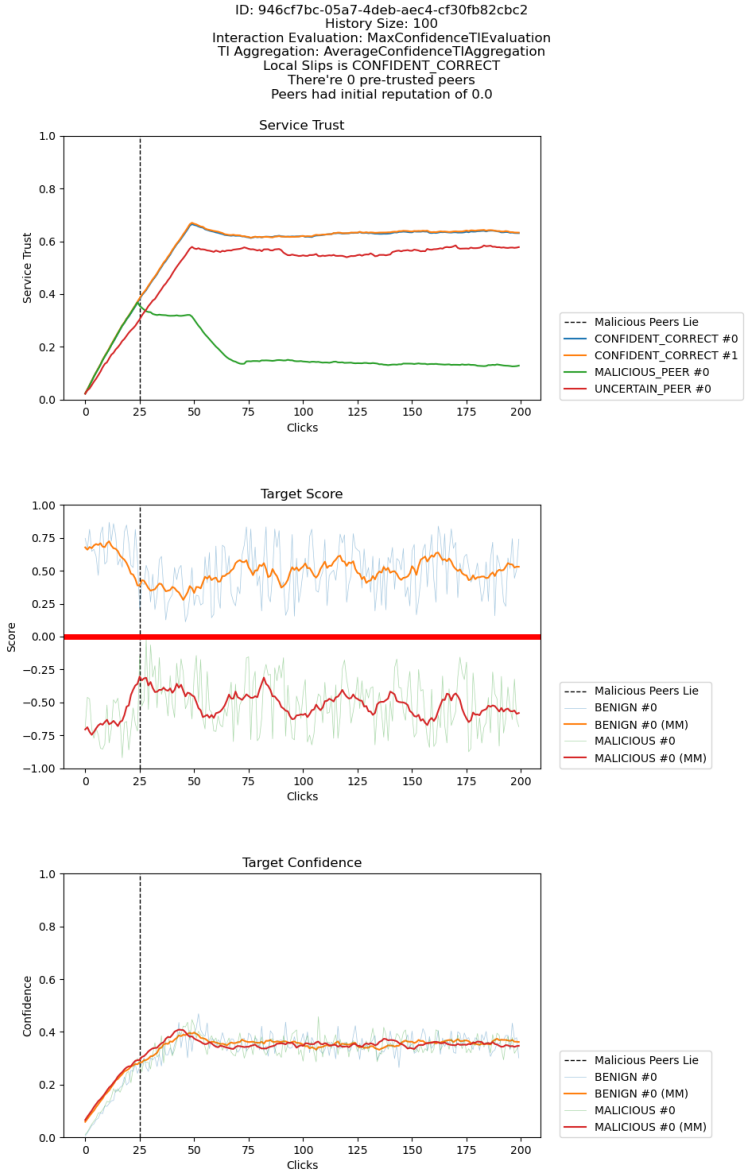


Figure 6.1: An example outcome from a single simulation. The graph on top shows how *service trust* changes as time goes by. In this example there are four peers, two confident correct, one uncertain and one malicious. The graph in the middle shows the score for the targets as computed by Fides based on what the peers said. There are two targets (imagine *google.com* and *evil.com*) and Fides computes the score for each of them: 1 means benign, -1 means malicious. The lower graph shows the aggregated confidence for the same targets. That means how confident is Fides about the score in the middle graph.

The graph's headline explains which setup parameters were used for the trust model. In the case of Figure 6.1 Fides used the interaction evaluation strategy $MaxConfidenceTIEvaluation$ (Section 3.6.6). For aggregating threat intelligence, Fides used the aggregation described in Section 3.7. The local Slips instance behaved like a confident correct peer outlined in Section 5.2.1.

The graph on top in Figure 6.1 shows the development of the *service trust* $st_{i,j}$ (Section 3.5.1) on the vertical axis over *time* on the horizontal axis. As mentioned in Section 5.3, the time is measured in *clicks*. The higher the service trust is for a peer, the higher impact it has on the final aggregated threat intelligence. One can see multiple peers that were involved in the simulation and their respective behavior. All possible behaviors are described in Section 5.2. There were four different peers that were communicating with the local instance of Fides, two of them were *confident correct*, one was an *uncertain peer* and the last one was a *malicious peer*. On the first graph, we can see that all peers were gaining the service trust at the beginning of the simulation and then their trust stabilized during the time. The exception is the malicious peer, its service trust was getting higher at the beginning of the simulation, but then it took a hit and was lowered when the peer started to lie.

The dotted line indicates the time when the malicious peers start lying. As mentioned, one can see that during this first period, when the malicious peers were not lying *(before the line)*, they were gaining the service trust. In the case of Figure 6.1 this happened at click 25 when the malicious peers started lying. After that, it is clear that they started to lose the service trust.

The second graph in Figure 6.1 shows the *target score* during the time (*clicks*). The target score $S_T^k$ (Section 3.7) is the part of the aggregated network threat intelligence, that was computed from the scores and confidences provided by each peer. The score was calculated by Fides at click $k$ for target $T$. The score graph contains two different targets, one that is according to the ground truth malicious and a second one that was benign (imagine *google.com* and *evil.com*) and Fides computes the score for each of them: 1 means benign, $-1$ means malicious. We also included the moving average value (indicated as MM) within the window of 10 clicks to make the graph clear. In a perfect scenario, we would see two straight lines where for benign target (*google.com*) would be the line in $S_T^k = 1$ and for the malicious target (*evil.com*) in $S_T^k = -1$. However, in the imperfect world, we can see that the lines fluctuate according to the score the Fides received from the peers. In a case, when the lines cross, and the benign target ends up below

the red line ($S_T^k \leq 0$) or the malicious target above the red line ($S_T^k \geq 0$) the Fides misclassified the targets and the attackers were successfully able to influence the decision of the trust model.

Finally, the third graph, displays the aggregated confidence $C_T^k$ (Section 3.7) over time (*clicks*). The graph is similar to the score, we include raw values for each time window and target, as well as the moving average within the window of 10 clicks.

In this example output graph, it can be seen that Fides was clearly able to identify that the malicious peer started to lie after click 25 because of the service trust $st_{i,j}^k$ for this peer that fell down almost instantly. At the same time, we can see that on the score graph, the $S_T^k$ for both targets were skewed and started to get closer to 0 because the malicious peer had already gained service trust and thus the threat intelligence provided by it had an impact on the final $S_T^k$. However, after Fides identified that the peer is lying, it lowered the service trust for this peer, and the score started to recover closer to the baseline.

## 6.2 Evaluation of Fides Resilience

To evaluate the resilience of Fides in different scenarios, we need to find the optimal configuration for the following parameters in Fides: interaction evaluation strategy (Section 3.6), threat intelligence aggregation function (Section 3.7), and initial reputation (Section 3.5.4). Each combination of parameters is evaluated in its capacity to correctly classify targets in *any* network topology.[1]

In this section, we are focusing on finding the best possible combination of parameters for the worst possible scenario. In other words, we want to identify a setup, where the Fides can guarantee that it is eventually going to provide the correct data and will classify the targets correctly even though the malicious actor controls most of the network.

We show two specific scenarios - one with no pre-trusted peers and one where there are 25% of peers part of some pre-trusted organization. Because even the scenario with only 25% peers shows, that in some cases is Fides able to defend itself against the rest of the network, we do not show scenarios with more pre-trusted peers, but we include them in the appendix (Figure A.6).

[1] Distribution of correct/uncertain/incorrect/malicious peers in the network.

### 6.2.1 Scenario With No Pre-Trusted Peers

In this scenario, there are no pre-trusted peers or organizations and Fides needs to determine trust in each peer by itself. We simulated environments starting with the 75% of confident correct peers (behavior from Section 5.2.1) up to 75% malicious peers (behavior from Section 5.2.4) and used all possible setups.

### Target Detection Performance

The target detection performance *tdp* (Section 5.4.1) is the most important metric because it evaluates how good is Fides in the target classification - if the Fides is able to correctly come up to a conclusion that *evil.com* is the malicious target and *google.com* is benign.

Figure 6.4 visualizes the target detection performance on three different graphs where each of the graphs is a single interaction evaluation strategy. Each graph then displays dots with different colors. Each color a single threat intelligence aggregation method in combination with different initial reputation values. A single dot in the graph is the value of *tdp* and in a case when the $tdp \geq 1$, it means that Fides made on average the wrong decision about the targets and classified them with the wrong label. In other words, if $tdp \geq 1$, Fides classified benign targets as malicious and the other way around. We included the *red line* that shows $tdp = 1$ so if a dot is above the *red line*, the Fides made an incorrect target classification. For that reason, we optimize the *dots* to be *below* the *red line* (classifications being correct).

The horizontal axis in each graph measures the environment hardness explained in Section 5.4.3. It is important to note, that hardness essentially expresses how many peers that can provide correct data are in the simulation. For example, if the hardness is 10, 100% of peers inside the simulation are providing correct data and behave like confident correct peers. Thus the higher the value of hardness is, the easier it is for the Fides to do correct classification.

Specifically, in Figure 6.2 we can see that in the *easy* environment, most of the *dots* are below the red line until the hardness gets close to 3. The metrics perform more or less the same as they are able to stay below the red line until $eh = 3$. In that situation, the best performance and thus the lowest *tdp* has $ThresholdTIEvaluation$ in combination with $WeightedAverageConfidenceTIAggregation$ and initial reputation of 0.95. Because the previous performance of all methods is almost similar, we recommend using this combination up to $eh3$ to ensure the best performance even in the harder situations.

Interestingly, the $DistanceBasedTIEvaluation$ in combination with 0 initial reputation and $AverageConfidenceTIAggregation$ for threat intelligence aggregation, shows the same target classification performance in each environment - $tdp = 0$. This suggests that the method was unable to determine any trust for any of the peers. This is then later confirmed by Figure 6.3.

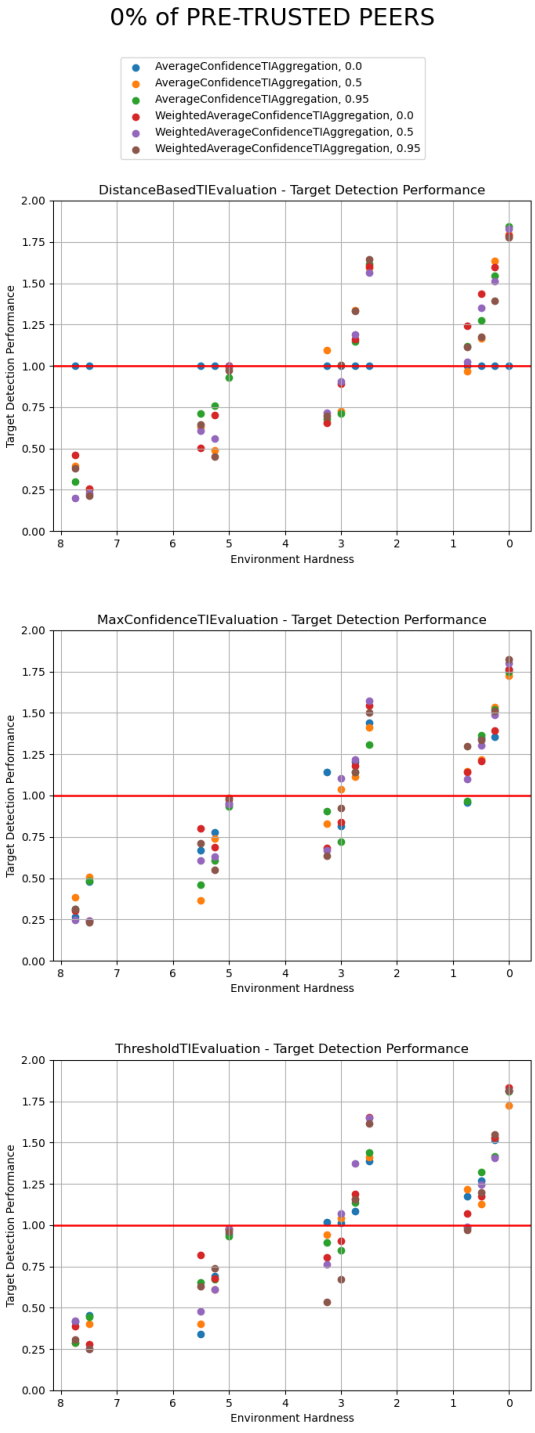


Figure 6.2: Target detection performance (vertical axis) for three different interaction evaluation strategies in different environments (horizontal axis) with **no pre-trusted peers**.

### Peer's Behavior Detection Performance

Figure 6.3 displays two important metrics which are related to how much Fides trusts the peers in the network. The first is the peer's behavior detection performance metric *pbdp* (Section 5.4.2) and the second is the peer's average trust.

On the left side, one can see the peer's behavior detection performance metric that measures how good was Fides in estimating the peer's behavior. The lower value of *pbdp* the better because the Fides's service trust for the peer was closer to the real value used in the simulation.

On the right side, we show the peer's average trust metric. That is an average trust of Fides for each peer. We include this metric in order to see how much trust was Fides able to obtain for the peers in the network. It is important to note, that there is no *correct* or *desired* value of this metric, because for example in the environment where there are all peers confident correct, the peer's average trust should be high, but in the environment with all byzantine peers, this metric should be low because Fides should not trust incorrect and malicious peers.

As suggested in the previous section while measuring target detection performance, the right graph for $DistanceBasedTIEvaluation$ in combination with $AverageConfidenceTIAggregation$ shows, that this setup is unable to determine trust for the peers and has average peer's trust close to 0. This means that the trust model will almost always aggregate threat intelligence to score 0 with confidence 0 making it, in a fact, useless.

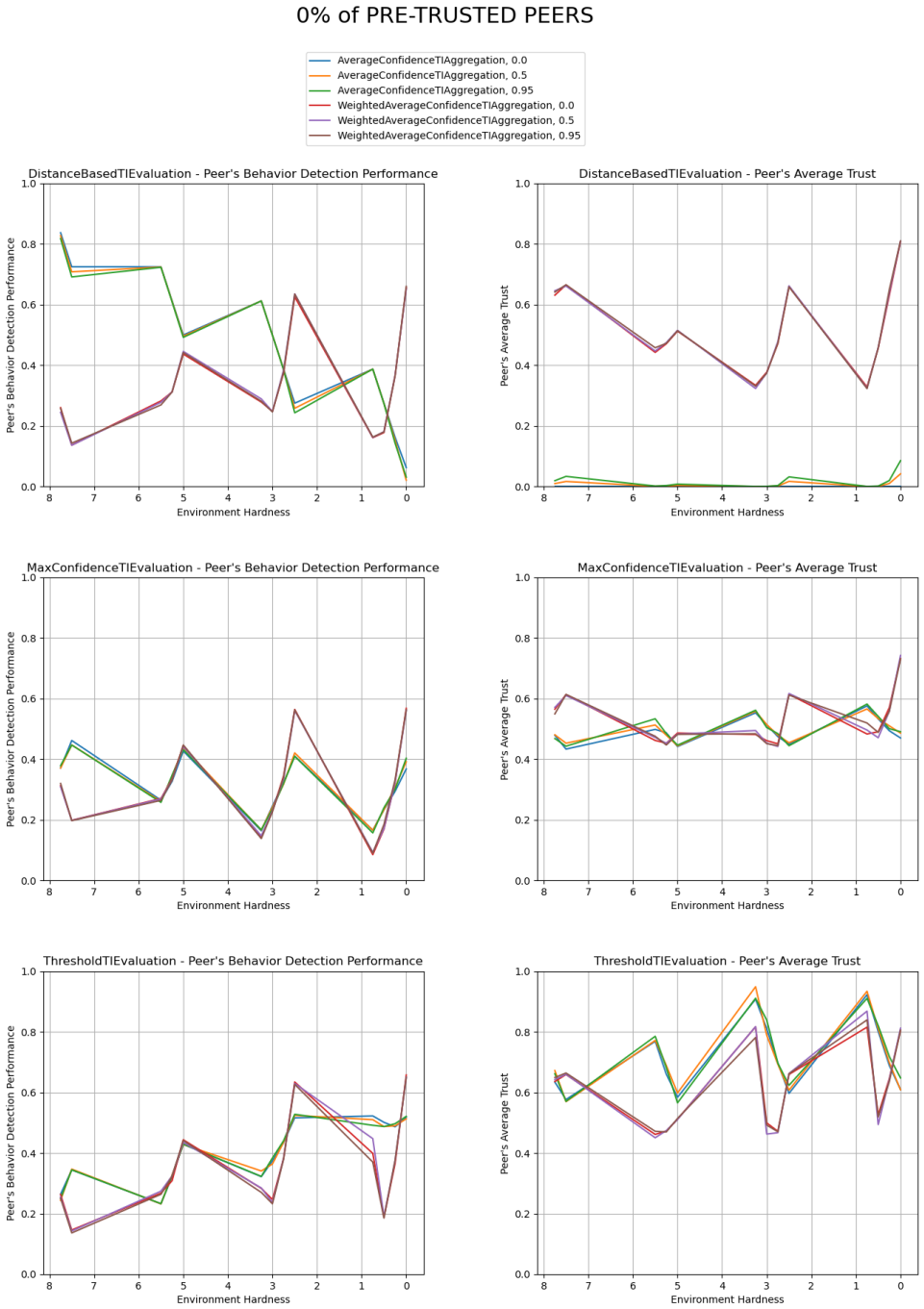


Figure 6.3: The behavior of peer's trust metrics in the different environments for different Fides's setups with **no pre-trusted peers**. On the left side peer's behavior detection performance, and on the right side peer's average trust.

### 6.2.2 Scenario With 25% of Pre-Trusted Peers

In this scenario, Fides assumes that there are 25% of pre-trusted peers. We simulated environments starting with the 75% of confident correct peers (behavior from Section 5.2.1) up to 75% malicious peers (behavior from Section 5.2.4) and used all possible setups.

#### Target Detection Performance

Specifically in Figure 6.4 one can see that until the environment hardness $eh \geq 3$, all strategies help Fides to classify the targets correctly. In general, the $DistanceBasedTIEvaluation$ performs the best in almost all situations except the easiest one, where the $MaxConfidenceTIEvaluation$ has slightly better results.

The situation changes after the environment hardness goes over 2.5 ($eh \leq 2.5$) when the $ThresholdTIEvaluation$ and $MaxConfidenceTIEvaluation$ misclassify the targets. In that case, all threat intelligence aggregation methods are the same and all of them misclassify the targets no matter what initial reputation is used. However, the $DistanceBasedTIEvaluation$ strategy in combination with $AverageConfidenceTIAggregation$ method is able to still classify the targets correctly and maintain the $tdp \leq 1$ even under the toughest conditions where there are 75% of adversarial peers in the simulation. We include a more detailed visualization of this case in the appendix in Figure A.1.

When Fides is used in a similar situation with the threat intelligence aggregation method $WeightedAverageConfidenceTIAggregation$, it misclassified the targets in one simulation when working in the hardest environment. Thus, this method does not provide a guarantee that Fides will end up with correct classifications for every target.

To summarize, the results have shown that with 25% of pre-trusted peers is Fides able to classify all the targets properly all the time if the interaction evaluation strategy $DistanceBasedTIEvaluation$ is used in combination with the $AverageConfidenceTIAggregation$. In such case, Fides provides a guarantee that no matter what the adversaries do, it calculates the correct threat intelligence.

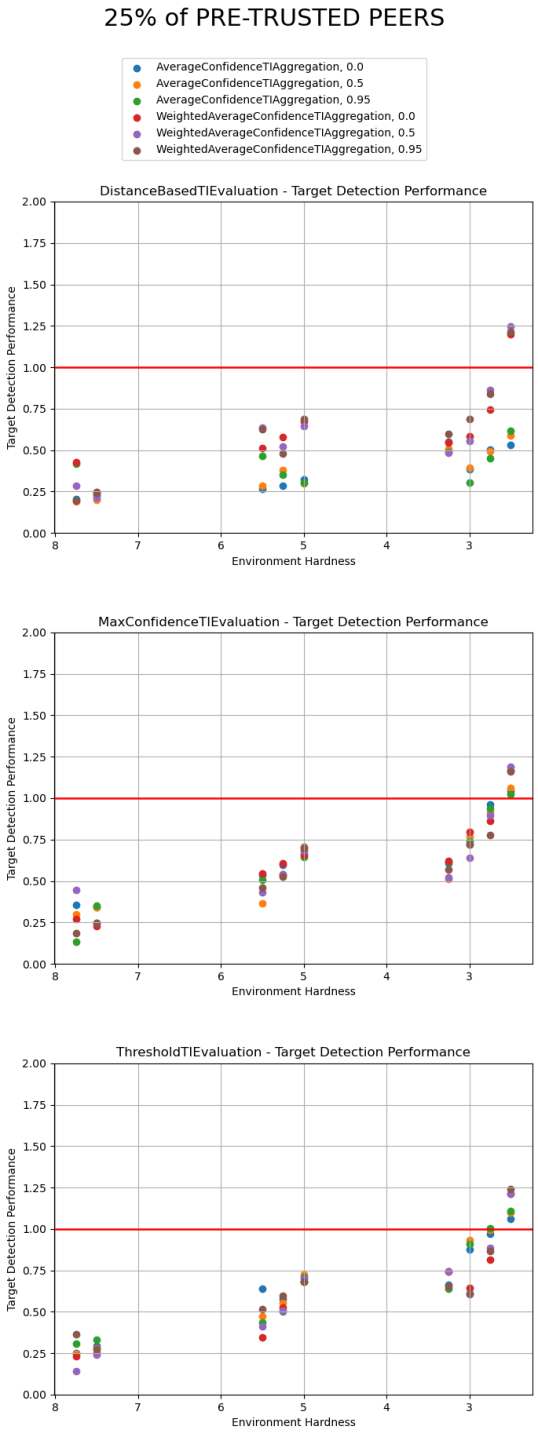


Figure 6.4: Target detection performance (vertical axis) for three different interaction evaluation strategies in different environments (horizontal axis) with **25% pre-trusted peers**.

### Peer's Behavior Detection Performance

Figure 6.5 shows the peer's behavior detection performance *pbdp* on the left side and the peer's average trust on the right side. When we compare Figure 6.3 (no pre-trusted peers) with this Figure 6.5 (25% pre-trusted peers), we can clearly see that, especially the *pbdp* metric improved and in all environments it holds that $pbdp \leq 0.4$. This means that Fides's ability to identify the true behavior of the peers greatly improved for all interaction evaluation strategies.

The biggest improvement was in strategy $DistanceBasedTIEvaluation$ which had poor performance with no pre-trusted peers in Figure 6.3. However, in the situation with 25% pre-trusted peers it is now able to detect the true behavior of the peers with the similar precision as the other strategies.

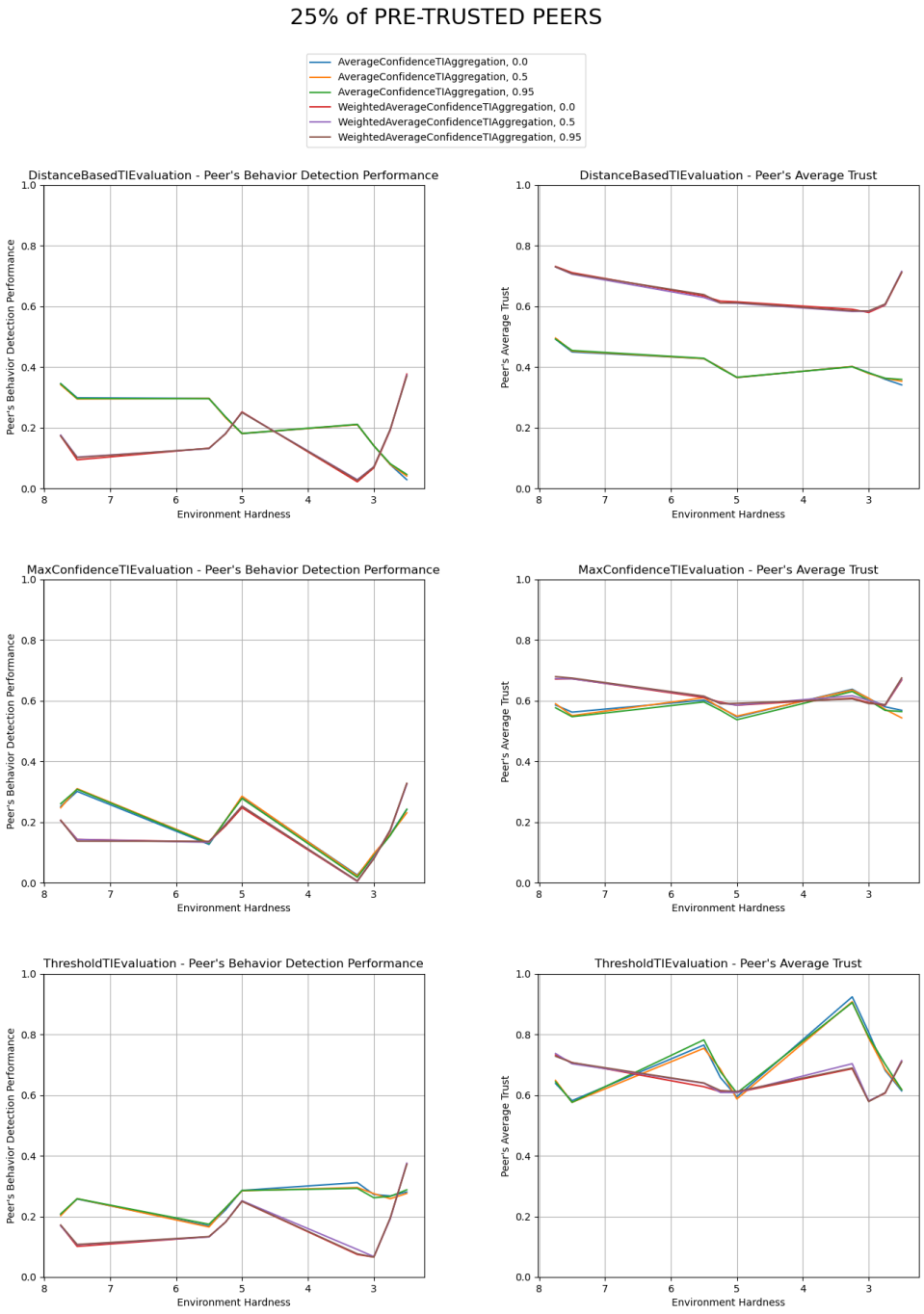


Figure 6.5: The behavior of peer's trust metrics in the different environments for different Fides's setups with **25% pre-trusted peers**. On the left side peer's behavior detection performance, and on the right side peer's average trust.

## 6.3 Considerations when Evaluating Trust Results

Even though the results from the previous Section 6.2 suggest that a combination of *DistanceBasedTIEvaluation* for evaluating the interactions in combination with *AverageConfidenceTIAggregation* is the best, there are corner cases where this is not always true.

For example, recall Figure 6.1 from Section 6.1, where the presented situation uses *MaxConfidenceTIEvaluation* and it is able to correctly detect all types of peers as well as correctly determine the score for the target. However, if we take the same environment and the only difference is using *DistanceBasedTIEvaluation* for evaluating interactions, we get the following graph for the service trust in Figure 6.6. The graph for confidence as well as target score for the situation from Figure 6.6 can be seen in the appendix in the Figure A.3. This is also the same behavior that we described when we were describing Figure 6.3 in the previous Section 6.2.1.

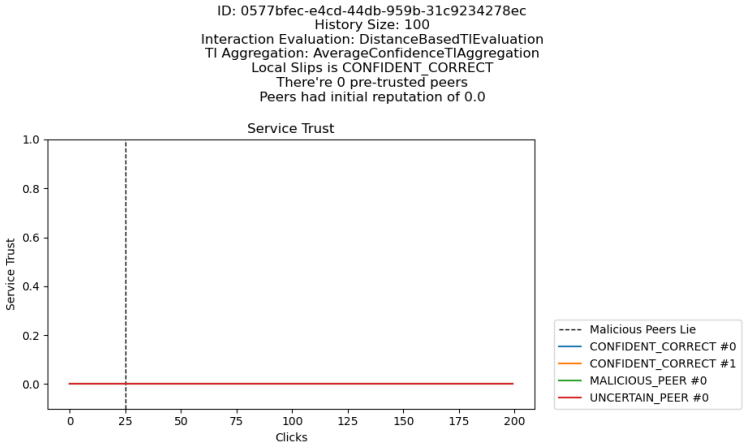


Figure 6.6: *DistanceBasedTIEvaluation* in the situation from the figure 6.1

The service trust graph in Figure 6.6 suggests that Fides didn't gain any trust for any peer in the network. This happens because the evaluation strategy didn't have enough information at the beginning to evaluate the received data properly. That leads to peers never gaining any trust and thus not producing any valid outputs because, with no trust, the target score and confidence ended up being 0 as well.

Another thing to consider is that during the simulations from the previous Section 6.2, the local Slips did not know anything about the targets. Which means that whenever Fides requested threat intelligence from Fides,

it responded as uncertain peer (Section 5.2.2). This simulates situations that are close to the reality when Slips is asking about the targets it does not know anything about. However, it also means that $MaxConfidenceTIEvaluation$ strategy will not live to its full potential as it is also using information from the local Slips.

## 6.4 Discussion

We discovered that actually there **exists** a particular setup that guarantees that Fides is able to eventually classify the targets correctly in a very adversarial situation. When Fides communicates with at least 25% of pre-trusted peers from pre-trusted organizations ($0.25 \cdot |P|$ are pre-trusted) and uses $DistanceBasedTIEvaluation$ (Section 3.6.2) for evaluating the interactions in combination with $AverageConfidenceTIAggregation$ (Section 3.7.1) for aggregating the threat intelligence; then Fides is able to correctly classify the targets no matter how many adversarial peers are in the network (up to filling the remaining 75%) or how hard they lie.

We included the graph of this case, similar to the Figure 6.1, with this particular *"winning"* setup in the most hostile environment to the Appendix in Figure A.1. For the explanation of the graph see Section 6.1.

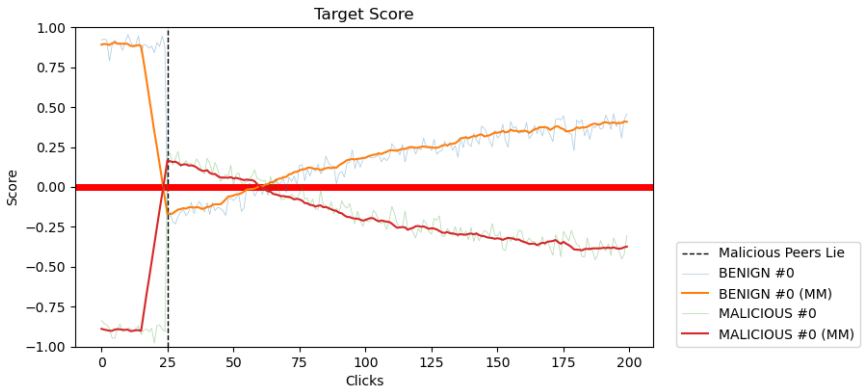


Figure 6.7: Score in figure A.2.

Interestingly, in this particular case, the initial reputation does not affect the final outcome of the simulation, but it does affect the progress as when using an initial reputation higher than 0, Fides provides wrong scores in the situation when the malicious peers started to lie. However, *in time* it discovers that the peers are lying, which decreases their service trust and is able to eventually recover the correct labels for the targets. The score value

over time for this situation can be seen in Figure 6.7. We included the whole graph in the Appendix in Figure A.2.

With no pre-trusted peers in the network, the results of each configuration vary and they highly depend on the network topology as well as on the knowledge of the local Slips instance. The results for the no pre-trusted scenario are shown in Appendix Figure A.4.

In the scenario of 50% pre-trusted peers, no matter the configuration, Fides was eventually able to determine the correct target classification with a high precision of $tdp \leq 0.7$. Moreover, Fides was able to correctly identify the peer's behavior with the precision of $pbdp \leq 0.2$. This is a very favorable situation for the administrator, where you trust the peers so much that it is not possible for the adversarial peers to modify the belief. The results for this scenario are shown in Appendix Figure A.6.

In general, for a case with no pre-trusted peers and organizations, one should use $ThresholdTIEvaluation$ because it proved to be slightly better than the others. However, in cases when the local Slips instance has some local knowledge about the targets the $MaxConfidenceTIEvaluation$ might be a better choice.

For cases where the Fides communicates with some pre-trusted peers, one should use the $DistanceBasedTIEvaluation$ threat intelligence evaluation strategy in combination with $AverageConfidenceTIAggregation$. This combination is even able to guarantee that with at least 25% of pre-trusted peers, it is able to eventually determine correct threat intelligence for all the targets.

# Chapter 7

# Conclusion

In this thesis, we proposed, designed, and implemented **Fides**, a generic trust model for highly adversarial global peer-to-peer (P2P) networks of intrusion prevention system (IPS) agents, fine-tuned for sharing security threat intelligence. Fides implementation allows IPS agents to cooperate against malicious actors, also taking care of the security and privacy of the local environment. Although Fides was designed for threat intelligence sharing, it is generic and can be easily modified to share any other type of data.

Fides is the first trust model that is implemented and ready to use, designed for global P2P networks of defenders that consider memberships of peers in organizations, evaluation of how much to trust peers based on interactions, aggregation of contradictory information of threat intelligence and adversarial peers trying to manipulate the system.

Fides was implemented and evaluated by simulating thousands of scenarios to find the best trust model parameters for each of them. We found that Fides helps obtain the truth about the shared data, in the presence of adversarial peers lying about the data, in most scenarios and conditions. This is possible due to the correct combination of trust methodologies and the new concept of pre-trusted organizations in Fides. Pre-trusted organizations follow the very human tendency of having people that we know and trust more than others. Even though Fides can work without pre-trusted organizations, having them greatly improves its resilience.

We conclude that Fides is a viable solution for managing trust relationships in peer-to-peer networks and it is robust enough to work under highly-adversarial situations. Even in these situations, with 75% of the peers malicious, it can correctly identify the truth of the shared threat in-

telligence data, and also identify the truth about how much to trust other peers.

Thanks to all the configurations and optimizations that Fides has for different organizations, we believe that Fides enables the Slips Intrusion Prevention System adoption across a broader spectrum of organizations. We believe that Slips, combined with Fides, will enable more possibilities for sharing threat intelligence over the internet and improve the overall security of the networks that use Slips for the defense against intruders.

## 7.1 Future Work

Even though we achieved our goal of designing a resilient trust model for sharing threat intelligence, there are multiple areas where the model can improve either by exploring different approaches or by implementing new additions to the data flow.

Most importantly, it is clear from the evaluation of simulations in Chapter 6, that Fides performs differently with different setups under different conditions. Even though we are able to manually pick settings that ensure the best performance, this is not a perfect solution for real-world scenarios. For that reason, we propose to explore further the following approaches.

### 7.1.1 Exploring Interaction Evaluation Strategies

The trust model developed in this thesis is generic and it is heavily relying on the interaction evaluation function. This means that the performance of the trust model is as good as the evaluation function.

In this work, we explored evaluation methods that are using only data from a single time window to evaluate the interactions. However, the local peer might store the complete interaction history with all other peers and whenever it finds out that some peer reported threat intelligence, that proved to be correct after some time, the reporter's service trust should benefit from such discovery. Another way might be storing the whole interaction history and using machine learning techniques, to discover irregularities in provided data during all communication windows.

In other words, all interaction evaluation strategies described in Section 3.6 do not utilize the knowledge from the past, or the history of the interactions. We believe that this is an interesting space to explore more as it might lead to better performance of Fides in real-world scenarios.

### 7.1.2 Exploring Threat Intelligence Aggregation Methods

No threat intelligence aggregation methods explored in this thesis, described in the Section 2.3, use any other information than ones provided by the network at a single time window. By incorporating information from the recent history, the aggregation might improve the overall detection performed by the trust model.

Moreover, there might be better ways how to aggregate the threat intelligence, or maybe the combination of multiple approaches to one might improve the final confidence in trust model decisions. As we saw in Chapter 6,

the combination of interaction evaluation and threat intelligence aggregation influences the performance of the trust model to great extent.

### 7.1.3 Possible Mitigation of Sybil Attack

When analyzing the possible attack vectors for Fides, we described the Sybil attack in Section 3.4.4 and we stated that Fides is vulnerable to this type of attack. However, results have shown that this is true only in cases when the attacker owns more than 75% of the network. If the attacker controls 75% of the network or less, Fides has a way how to defend itself and make the correct decisions about the threat intelligence.

There are essentially two possible ways how to mitigate this type of attack even for cases where the attacker controls more than 75% of the network.

One option is to introduce restrictions to Fides, where it uses data at most from the 75% of peers that are not pre-trusted, and the rest, 25% comes from the pre-trusted peers and organizations. All other data from other peers in the network would not be considered during the threat intelligence aggregation. That would ensure that Fides always only considers a limited amount of data so it will not be vulnerable to an attacker who controls more than 75% of the network.

The second possible solution would be making it *computationally hard* for new peers to join the peer-to-peer network or generate a new peer identity. So the attacker would need to spend either time or computational resources when generating new peer identities. This directly does not prevent a malicious actor to perform this type of attack, but it makes it significantly harder. However, this mitigation would need to be implemented in Iris instead of Fides.

### 7.1.4 Adding Threat Intelligence Challenges

Fung et al [15] explore an interesting idea of creating initial trust in remote peers by giving them *computational challenges*. In our case, challenges might be asking the remote peer for threat intelligence about the target, that the local Slips know very well and with high confidence. When the trust model receives a response it uses the interaction evaluation strategy described in Section 3.6.4 to evaluate the received data. The trust model can then ask multiple times to have more interactions and thus effectively *probing* the remote peer and getting an estimate about its future behavior.

By using this approach, one can either replace the recommendation system or greatly improve it in cases when the trust model does not have

enough pre-trusted peers. The disadvantage might be a higher amount of messages sent across the network when the new peer is registered by the network, which could eventually lead even to a denial of service attack on the newcomers.

This method should be explored more in detail and as the Fides is quite flexible, it can be easily incorporated into the trust model as well.

### 7.1.5 Threat Intelligence Sharing Motivation

The designed peer-to-peer network for threat intelligence sharing unfortunately does not promote data sharing at all. The peers sharing threat intelligence are not gaining any benefit other than gaining trust in the eyes of other peers. As the gained trust brings little to no benefit by itself, there is a lack of incentive for the peers to share their threat intelligence with the network.

However, we can see data and threat intelligence filtering as one of the ways how to motivate other peers to share their threat intelligence. As we described in Section 4.2.3, the Fides administrator can choose what threat intelligence is shared with which peers by configuring confidentiality levels with required service trust. Using this approach, the threat intelligence is shared only with high trusted peers. That would result in an incentive for the peers to gain the service trust which they do by sharing their own threat intelligence with the network.

Another approach is exploring the idea of *payments* for the threat intelligence where the peers use some sort of cryptocurrency to *pay* for the received threat intelligence and they receive payments for their own threat intelligence they shared. This would give the peers an incentive to share the threat intelligence and ask for it only when they need it.

# Bibliography


[1] Tigist Abera et al. "Sadan: Scalable adversary detection in autonomous networks". In: *arXiv preprint arXiv:1910.05190* (2019).

[2] AbuseIPDB. *AbuseIPDB making the internet safer, one IP at a time.* URL: `https://www.abuseipdb.com/`. (accessed: 15.05.2022).

[3] Inc. Amazon.com. *What is an API?* URL: `https://aws.amazon.com/what-is/api/`. (accessed: 14.05.2022).

[4] Rebecca Gurley Bace, Peter Mell, et al. *Intrusion detection systems.* 2001.

[5] Ahmet Burak Can and Bharat Bhargava. "Sort: A self-organizing trust model for peer-to-peer systems". In: *IEEE transactions on dependable and secure computing* 10.1 (2012), pp. 14–27.

[6] Ingrid Alina Christensen and Silvia Noemi Schiaffino. "A hybrid approach for group profiling in recommender systems". In: (2014).

[7] Software Freedom Conservancy. *Git –everything-is-local.* URL: `https://git-scm.com/`. (accessed: 24.04.2022).

[8] Dataplane.org. *Data Plane.* URL: `https://dataplane.org/`. (accessed: 15.05.2022).

[9] Binary Defense. *Binary Defense Systems Artillery Threat Intelligence Feed and Banlist Feed.* URL: `https://www.binarydefense.com/banlist.txt`. (accessed: 15.05.2022).

[10] Conda Developers. *Conda.* URL: `https://docs.conda.io/en/latest/`. (accessed: 24.04.2022).

[11] John R. Douceur. "The Sybil Attack". In: (2002). Ed. by Peter Druschel, Frans Kaashoek, and Antony Rowstron.

[12] Nicolas Falliere. "Sality: Story of a peer-to-peer viral network". In: *Rapport technique, Symantec Corporation* 32 (2011).

[13] Lukas Forst. *Fides, A Trust Model for Collaborative Defense in Highly Adversarial Networks*. URL: https://github.com/stratosphereips/fides.

[14] Python Software Foundation. *Python*. URL: https://www.python.org/. (accessed: 24.04.2022).

[15] Carol J Fung et al. "Trust management for host-based collaborative intrusion detection". In: *International workshop on distributed systems: operations and management*. Springer. 2008, pp. 109–122.

[16] Sebastian Garcia, Alya Gomaa, and Kamila Babayeva. *Slips, behavioral machine learning-based Python IPS*.

[17] Inc. GitHub. *GitHub - Where the world builds software*. URL: https://github.com/. (accessed: 24.04.2022).

[18] Rui He, Jianwei Niu, and Kai Hu. "A Novel Approach to Evaluate Trustworthiness and Uncertainty of Trust Relationships in Peer-to-Peer Computing". In: *The Fifth International Conference on Computer and Information Technology (CIT'05)*. 2005, pp. 382–388. DOI: 10.1109/CIT.2005.30.

[19] Ing. Dita Hollmannová. "Trust Models on Adversarial Distributed Security Agents". MA thesis. Aug. 2020. URL: https://dspace.cvut.cz/bitstream/handle/10467/90252/F3-DP-2020-Hollmannova-Dita-final.pdf.

[20] Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. "An integrated trust and reputation model for open multi-agent systems". In: *Autonomous Agents and Multi-Agent Systems* 13.2 (2006), pp. 119–154.

[21] json.org. *Introducing JSON*. URL: https://www.json.org/json-en.html. (accessed: 14.05.2022).

[22] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks". In: *Proceedings of the 12th international conference on World Wide Web*. 2003, pp. 640–651.

[23] Eleni Koutrouli and Aphrodite Tsalgatidou. "Taxonomy of attacks and defense mechanisms in P2P reputation systems—Lessons for reputation system designers". In: *Computer Science Review* 6.2 (2012), pp. 47–70. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2012.01.002. URL: https://www.sciencedirect.com/science/article/pii/S1574013712000093.

[24] Wenjuan Li, Weizhi Meng, et al. "Design of intrusion sensitivity-based trust management model for collaborative intrusion detection networks". In: *IFIP International Conference on Trust Management*. Springer. 2014, pp. 61–76.

[25] Redis Ltd. *Redis*. URL: `https://redis.io/`. (accessed: 04.05.2022).

[26] Vasileios Mavroeidis and Siri Bromander. "Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence". In: *2017 European Intelligence and Security Informatics Conference (EISIC)*. 2017, pp. 91–98. DOI: `10.1109/EISIC.2017.20`.

[27] Isaac Pinyol and Jordi Sabater-Mir. "Computational trust and reputation models for open multi-agent systems: a review". In: *Artificial Intelligence Review* 40.1 (2013), pp. 1–25.

[28] Bc. Martin Řepa. "Global P2P Network for Confidential Sharing of Threat Intelligence and Collaborative Defense". MA thesis. May 2022. URL: `https://www.stratosphereips.org/thesis-projects-list/2022/3/12/global-permissionless-p2p-system-for-sharing-distributed-threat-intelligence`.

[29] S. Gummadi K.P. Gribble S.D. Saroiu. "Measuring and analyzing the characteristics of Napster and Gnutella hosts." In: *Multimedia Systems* (2003), pp. 170–184. DOI: `10.1007/s00530-003-0088-1`.

[30] R. Schollmeier. "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications". In: (2001), pp. 101–102. DOI: `10.1109/P2P.2001.990434`.

[31] S Udhaya Shree, Dr Basha, and MS Saleem. "An exhaustive survey of trust models in P2P network". In: *arXiv preprint arXiv:1411.3294* (2014).

[32] YAML Language Development Team. *YAML Ain't Markup Language (YAML™) version 1.2*. URL: `https://yaml.org/spec/1.2/`. (accessed: 04.05.2022).

[33] Cynthia Wagner et al. "Misp: The design and implementation of a collaborative threat intelligence sharing platform". In: *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. 2016, pp. 49–56.

[34] Yao Wang and Julita Vassileva. "Trust and reputation model in peer-to-peer networks". In: *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*. IEEE. 2003, pp. 150–157.

[35] Wikipedia contributors. *Fides (deity) — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Fides_(deity)&oldid=1086924565`. [Online; accessed 15-May-2022]. 2022.

[36] Li Xiong and Ling Liu. "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities". In: *IEEE transactions on Knowledge and Data Engineering* 16.7 (2004), pp. 843–857.

[37] Xinyou Zhang, Chengzhong Li, and Wenbin Zheng. "Intrusion prevention system design". In: *The Fourth International Conference on-Computer and Information Technology, 2004. CIT'04.* IEEE. 2004, pp. 386–390.

# Appendix A

# Simulation Graphs

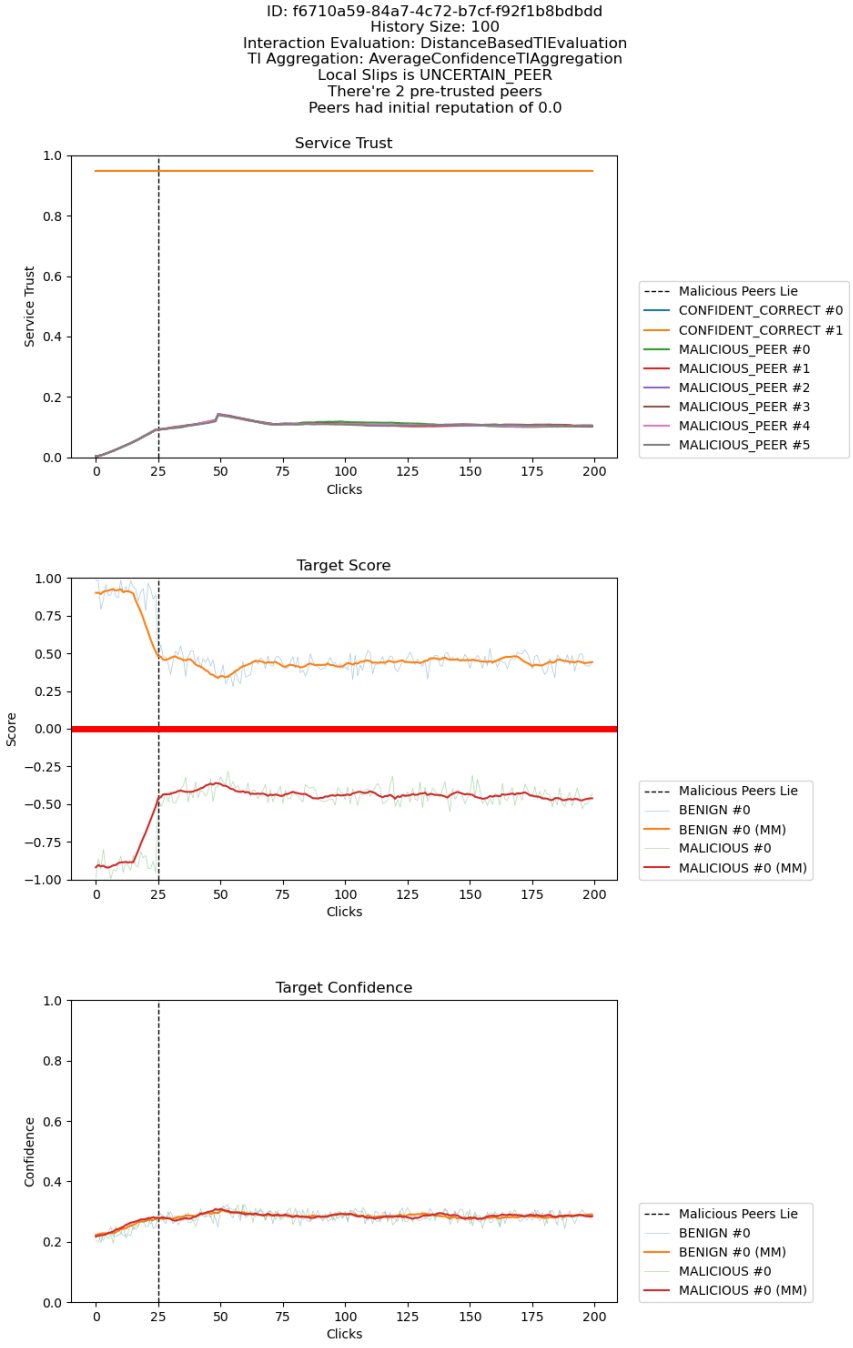


Figure A.1: The scenario where there are 75% of the peers malicious and 25% are from the pre-trusted organizations. The second graph show that the Fides was able to defend itself an still classify the targets correctly even though the malicious peers lied about the targets.

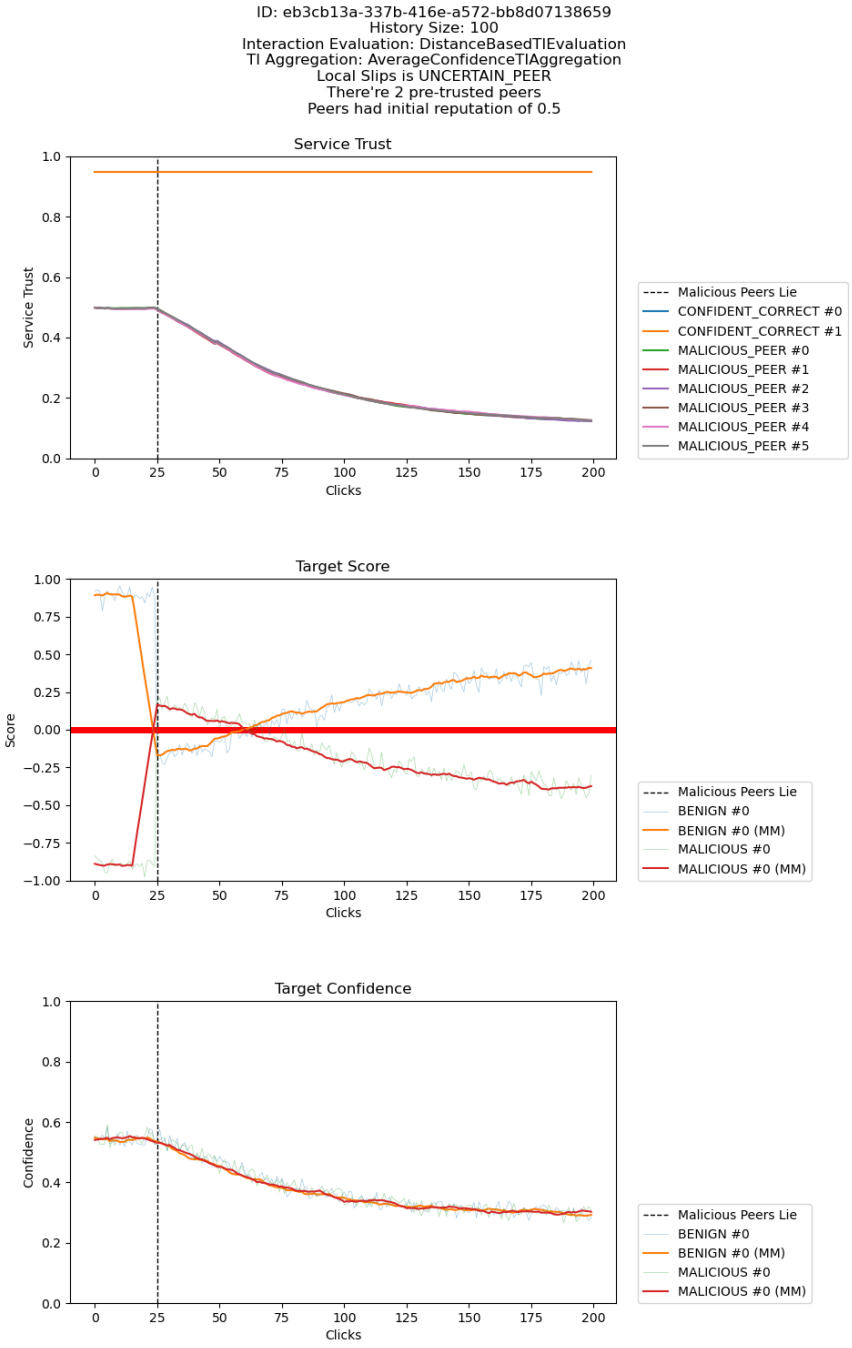


Figure A.2: The scenario when the malicious peers overcame Fides after they started lying but only for a short period of time. Once Fides realize that the peers are lying, it lowered their service trust and the it was able to recover the score to the correct values.

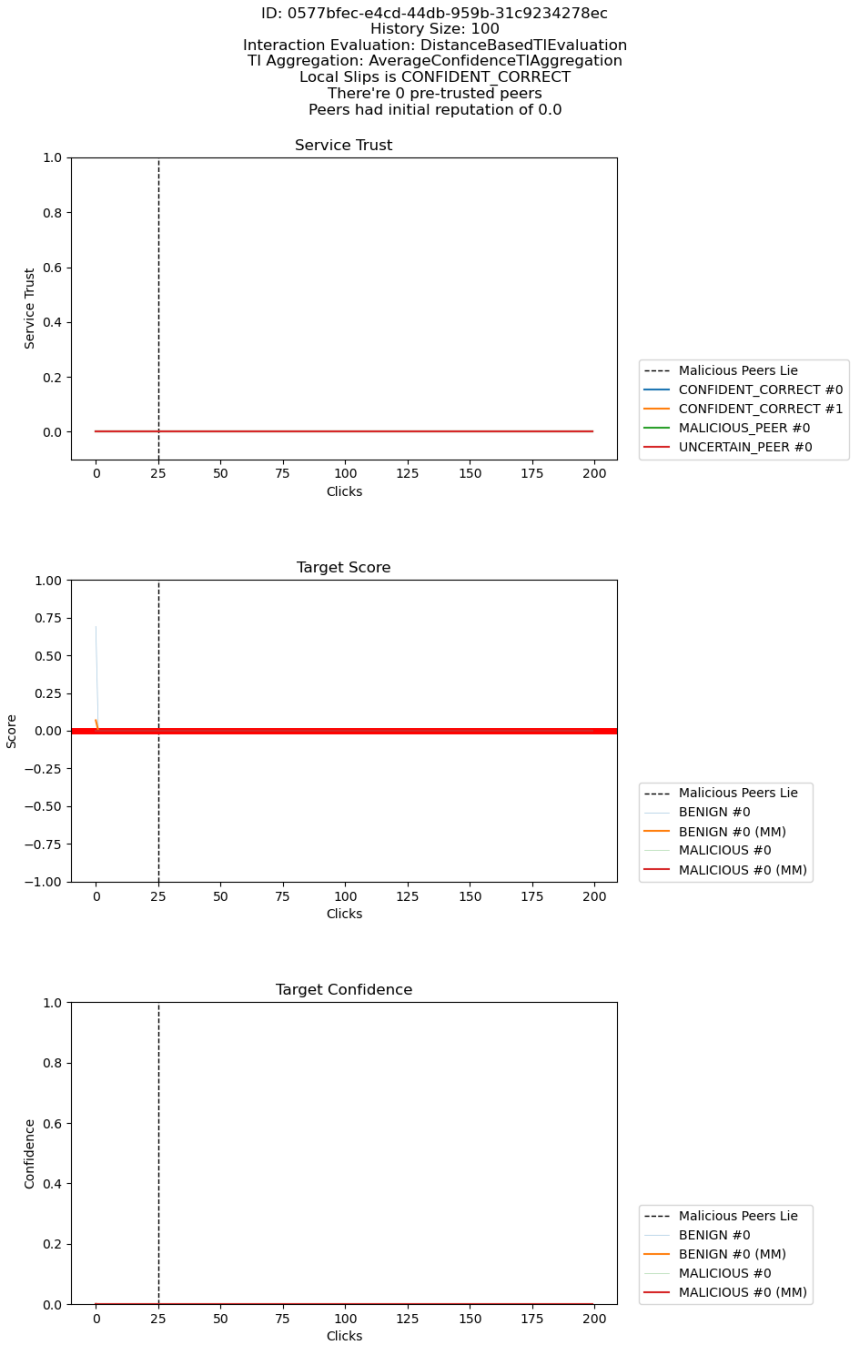


Figure A.3: The scenario when the Fides was unable to gain trust for any of the peers in the network. It is an example of how does the initial reputation matter during the Fides setup.

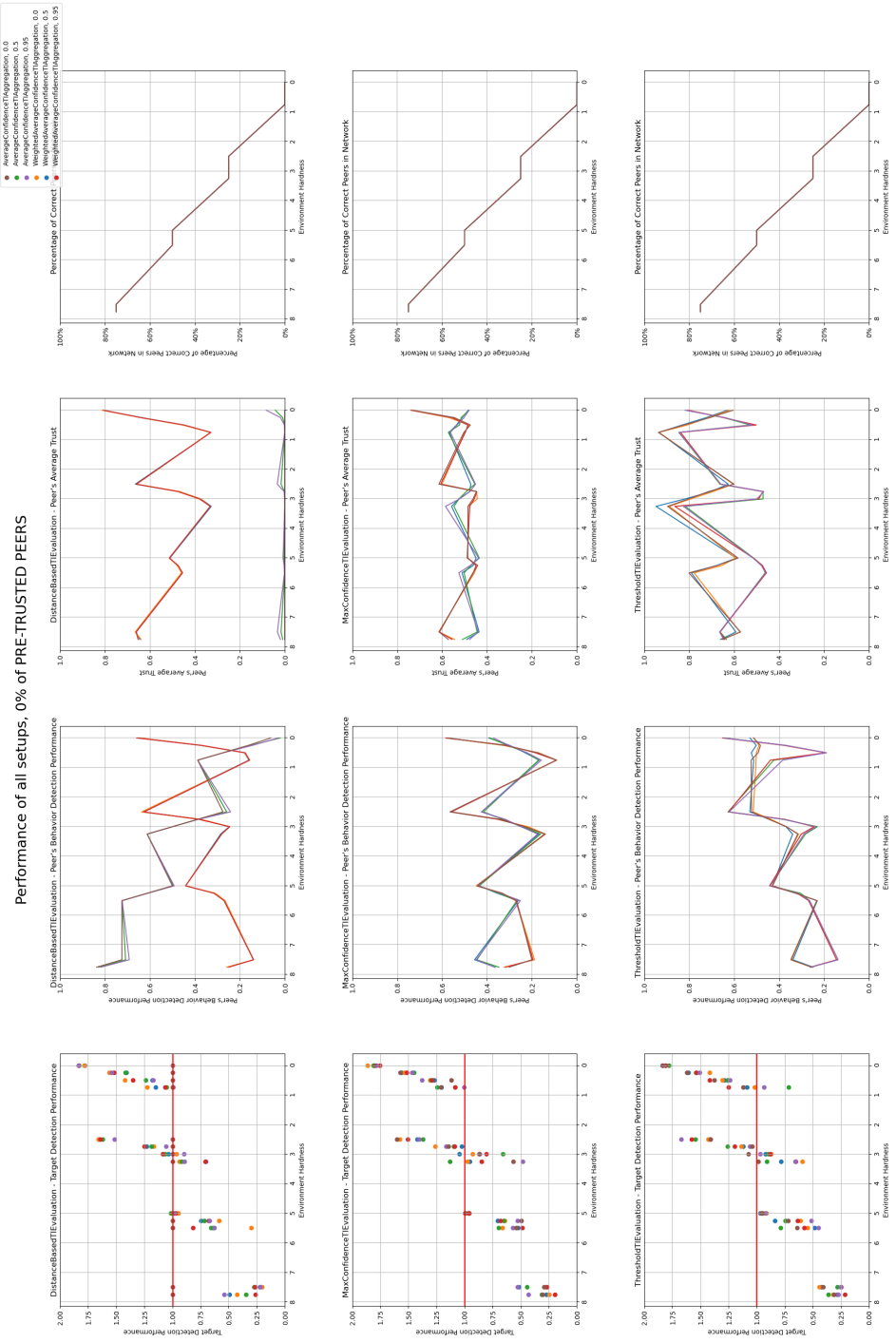


Figure A.4: All metrics in a single graph with **no pre-trusted peers**.

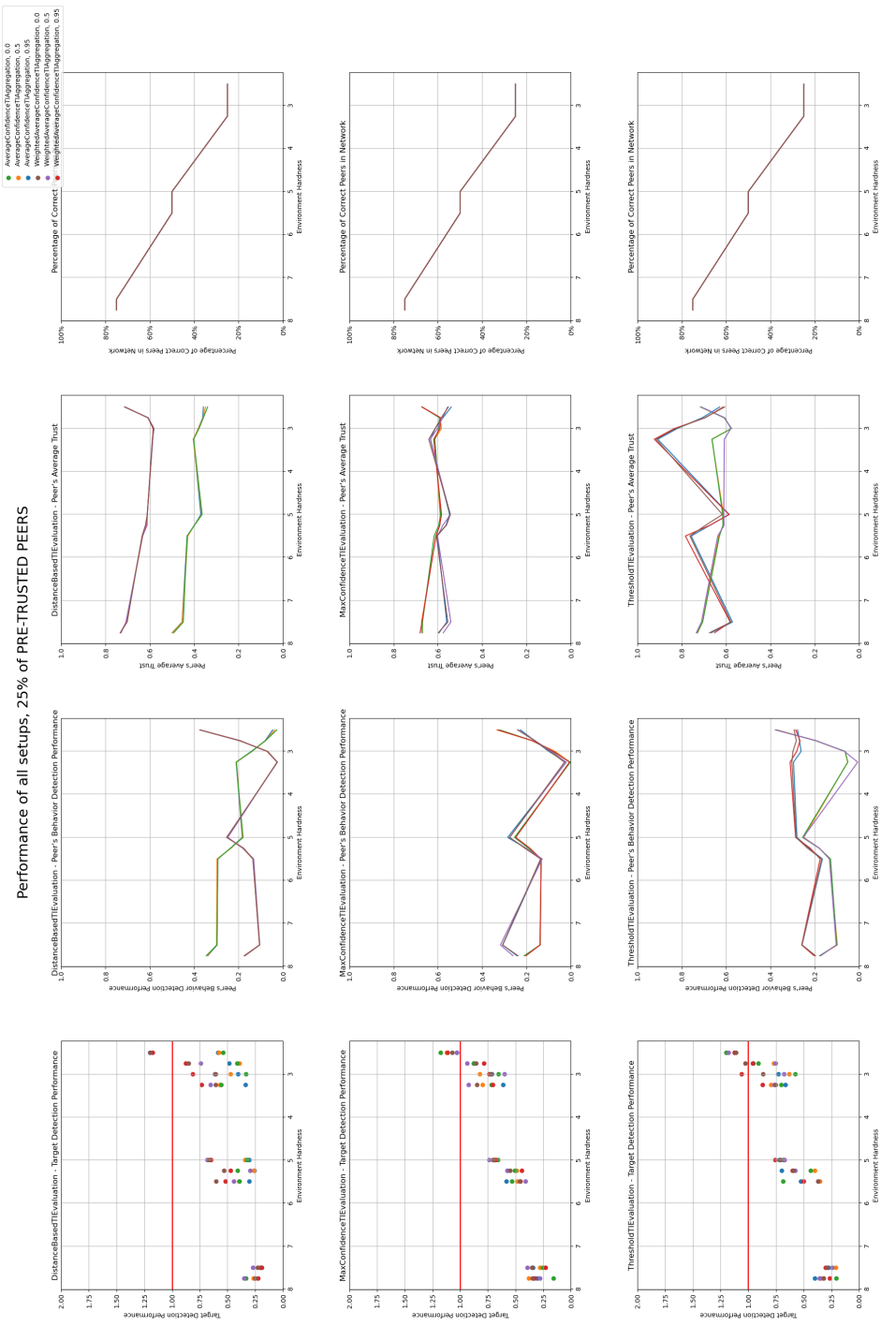


Figure A.5: All metrics in a single graph with **25% pre-trusted peers**.

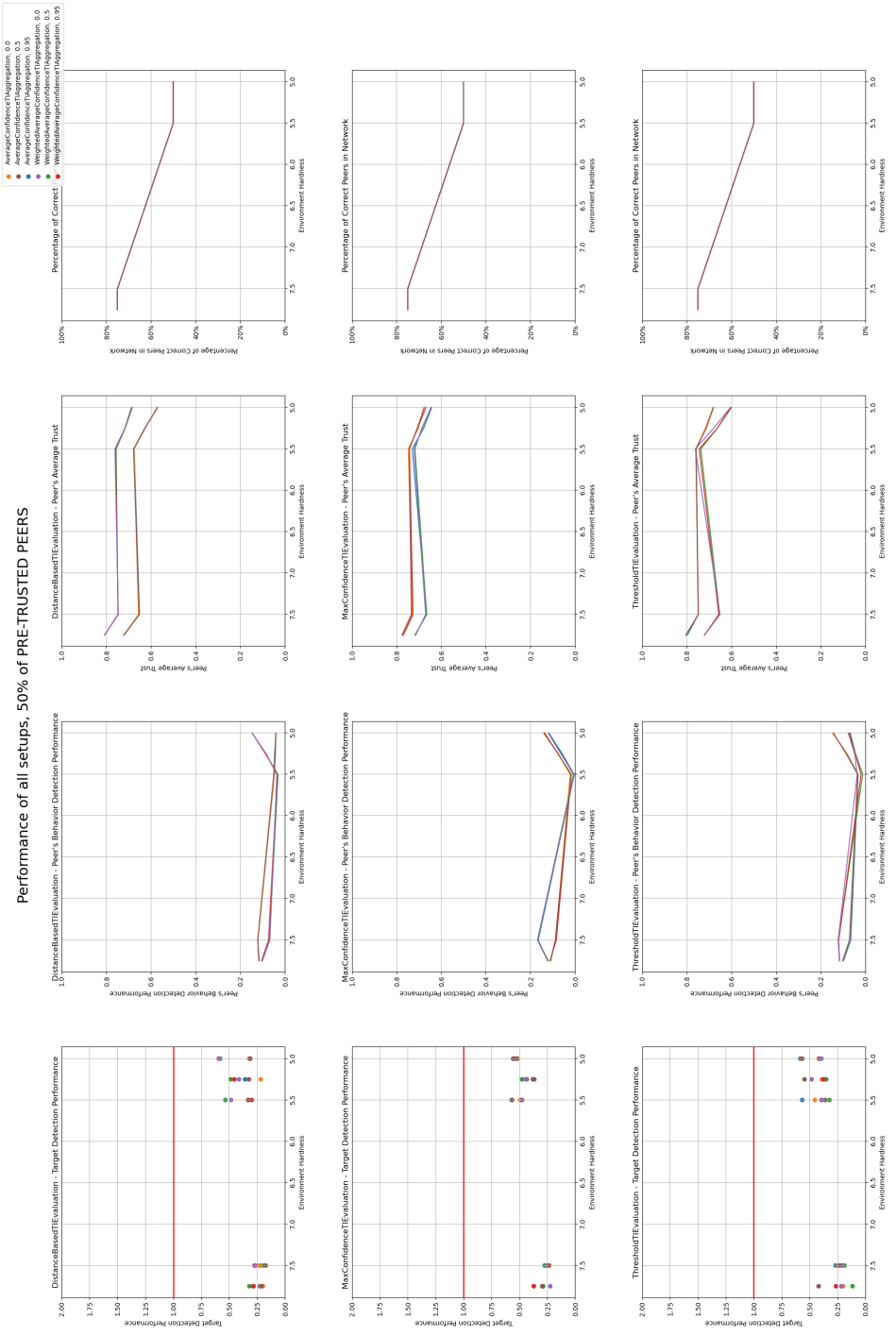


Figure A.6: All metrics in a single graph with **50% pre-trusted peers**.

# Diagram of Fides Trust Model

## From the point of view of peer i