Abstract—Cybersecurity defenders need to protect all the vulnerabilities in their assets to prevent attacks. Since the security knowledge of each company is only a fraction of the knowledge of the whole network, defenders can try to improve their stance by sharing information with other companies. There are already standard ways of representing cyber threat information, vulnerabilities (CVE), cyber observables, malware attributes, among others. There are also transport mechanisms for these kinds of information, and scoring models (CVSS) that help to quantify the severity of the vulnerabilities that are found. Recently, the MISP platform, designed for sharing, storing and correlating information about attacks on specific vulnerabilities, has been made available as open source, providing a development testbed for increased security information sharing.

The main objective of this work was to design and implement mechanisms that augment MISP for automated data exchange. The platform was extended with anonymization techniques to preserve the confidentiality of the asset information, of its owner and of the environment where the information was collected, while still providing ways to match security information producers with consumers. In this way, there is an added incentive for data sharing because data will be under the control of its owner.

Our solution integrates the network vulnerability scanner OpenVAS with the MISP platform. The configurable solution automatically issues OpenVAS scans, extracts results from the reports generated, anonymizes the obtained results and imports them into the MISP platform. The solution was evaluated with a synthetic testbed and with another testbed provided by an air navigation service provider.

I. INTRODUCTION

Cybersecurity is a constant struggle between attackers and defenders. Attackers have a big advantage on their side: they need only to find vulnerabilities to exploit, whereas defenders need to protect all the vulnerabilities in their assets to reduce exposure and attack surface [1]. Companies are very concerned about security, and need to have knowledge about the security issues to properly address them. Since the security knowledge of each company is only a fraction of the knowledge of the whole network, defenders can try to improve their current knowledge through information sharing, reducing exposure. This information can be used to identify new indicators of compromise, early warning signs and allow defenders to react accordingly.

Recently, the Malware Information Sharing Platform (MISP) [2] and its data models have been made available as open source, providing a development testbed for increased information sharing. MISP is designed for sharing, storing and correlating Indicators of Compromises\(^1\) [3] on targeted platforms. The figure 1 illustrates an example on how the sharing can be done: the instance in the middle publishes an initial event and shares it with other two instances; the other two instances contribute with their own attributes; and this changes are synced throughout the sharing model.

Moreover, the MISP platform is not able to obtain results by itself. It needs a human operator to publish events manually, which is impractical with the amounts of data an organization has to process nowadays and with the constant changes in the information security field. A possible alternative to share large amounts of data is using a third party application to feed the MISP platform events through its Representational State Transfer (REST) Application Programming Interface (API). By integrating a vulnerability scanner with the MISP API, it is possible to automate the generation and publication of new events to the platform. This assures that the system of the organization is scanned for the latest vulnerabilities in a timely manner and that the subsequent information exchange happens as soon as possible. Human operators can still monitor the events, allow or deny the automatic publication and modify information, if necessary.

Despite of the advantages gained by sharing threat information among defenders, sensitive information from the system of a participant in the exchange might be exposed. Particularly a vulnerability in the system of a business might be exposed to the competition of the same business, effectively discouraging the information sharing and the participation in such platforms [5]. This issue is particularly relevant since the publication

\(^1\)Specific artifacts left by an intrusion or greater sets of information that allow the detection of intrusions or other activities conducted by attackers
of the NIS Directive\(^2\), which requires that businesses in the critical sectors of economy vital for the society and heavily reliant on information and communications technology, such as energy, transport, water, banking, financial market infrastructures and healthcare report serious incidents to the relevant national authority, which may lead to the exposure of their vulnerabilities to other parties or even to their business competitors.

A possible approach to mitigate the unwillingness to share sensible data mentioned above would be to anonymize the identity of the entity that is publishing the vulnerability. However, such procedure would remove value from the information published since the unknown publisher could be anyone, from a reputable individual or company to a new user or even an attacker that managed to get in the information exchange in order to clutter the sharing platform with useless information to obfuscate the useful one. Even with the application of effective anonymization techniques, the publisher can be identified by unique characteristics in his own published event, such as the Internet Protocol (IP) addresses of the machines, personal information of clients and/or employees, which can not be shared in some jurisdictions without the consent of the individual, and traces of the usage of proprietary software owned by the publisher. So, these other identifiers should be anonymized too.

The main objective of this work was to design and implement mechanisms that augment MISP for automated data exchange by extending the platform to preserve the confidentiality of the information exchanged, while still providing ways to match security information producers with consumers. This work will presents a prototype of the Trusted Cooperative Exchange System, with a set of use cases that exemplify its usefulness in improving security awareness, for example, in response to zero-day vulnerabilities that allow attacks like ransomware and cryptojacking.

This article is structured as follows: section II presents the most relevant work related to threat assessment and sharing; section III describes the proposed solution; section IV presents the results obtained through the testbeds and the solution evaluation; and finally section V concludes with some final remarks about this work, achievements reached and possible future work.

II. RELATED WORK

This section presents an overview of the state of the art of vulnerability scanning and information sharing. It describes the process of capturing and exchanging security information, followed by the current challenges for these exchanges currently have.

A. Security Information Capture

Even though security specialists can detect and classify vulnerabilities through various methods such as: log analysis, fingerprinting, comparing between versions installed and versions provided by the publisher, checking password policies, among others; these tasks are repetitive, extensive and most be done often, particularly in large systems. To automate this process, there are a number of both open source and commercial vulnerability scanners.

A vulnerability scanner runs from a vantage point where it can inspect the assets in question. The software discovers alive hosts and enabled ports and proceeds to compare details about installed components and assets (such as banners containing versions) to a database of information about known vulnerabilities, anomalies in packet construction, and potential paths to exploitable programs or scripts. The scanner software lists known vulnerabilities according to the versions of the services detected and, in some situations, attempts to exploit them (unless configured otherwise). Even though some of these exploits are intrusive, they are non-malicious - e.g. trying to connect through Secure Shell (SSH) using common default credentials like root:root.

Examples of vulnerability scanners are Nessus\(^3\), a proprietary vulnerability scanner developed by Tenable Network Security; and OpenVAS \(^4\), which is an open source fork from the Nessus before it changed its license to proprietary.

B. Security Information Exchange

As the number of the vulnerabilities increases and, consequently, the attack surface expands [7] - direct methods to exchange this information individually are not scalable. Not only the number of different sources and individual threats keeps increasing rapidly, but there are also no mechanisms in these methods to group related information, leading to the repetition of the publication of same vulnerability multiple times, for example. Furthermore, in order to study a certain vulnerability, associated threats and its related information; looking through email lists or browsing through different blogs are methods that do not allow the automation.

In order to fix these issues, community sharing platforms, such as Collaboration Research into Threats (CRITs)\(^5\) and MISP \([2]\), were created. MISP in particular, combines a searchable repository with a multidirectional information sharing mechanism. Where possible, MISP also provides automation mechanisms that enable the automatic import and export of data from and to other systems. The aim is to speed up the detection of incidents and the production of defense countermeasures, especially for malware that is not blocked by anti-virus protection, or that is part of sophisticated targeted intrusion attempts \([8]\).

Finally, MISP data sharing is granular and discretionary. It allows users to share what they want with whom they want. In other words, even using MISP, organizations are allowed to keep events containing sensitive data within the organization.

C. Challenges to Security Information Exchange

Automation of access control to shared data is possible for small groups but, as the group size grows, the feasibility

---

\(^2\)Directive on security of network and information systems \([6]\).

\(^3\)https://www.tenable.com/products/nessus-vulnerability-scanner

\(^4\)http://www.openvas.org/

\(^5\)https://crits.github.io/
of controlling access to shared data using access control mechanisms diminishes. All it would take is one organization in a group to have its access control credentials compromised or one organization in a group to disclose credentials and then shared data would be exposed. The compromise of the credentials of one of the sharing organizations in the same group would compromise the security of the data sharing process. Anonymization is a solution that can provide practical levels of assurance that shared data cannot be used to cause harm. However, the use of anonymization entails trade offs that must be tailored to the participating organizations and specific situations [9].

To anonymize data efficiently, a balance between the privacy and utility of the data provided must be established. On one hand, anonymized data with too much focus on privacy, even though it makes attempts to find the owner identity harder, might not provide enough information to be useful after it is analyzed. On the other hand, “anonymized” data that focus too much on utility can provide extra information that is negligible to the analysis but which compromises the identity of the owner.

Anonymization of logs is especially important due to the sensitivity of the information contained within them. Access to computer and network logs can provide intruders with special views of a network not visible from the outside, even with scanning tools. Information gleaned from these logs could indicate potential bottlenecks for denial of service attacks or could even contain passwords (as often happens when a user accidentally types a password into the username field) [10].

D. Summary

This section introduces the definition of a vulnerability scanner and how it can be used to capture information, the introduction of the tool this work will aim to augment (MISP) and the challenges to the threat information exchange, namely due to the lack of trust in all participants of the sharing platform.

III. Solution

The goal of this work is to augment an existing threat sharing platform in order to incentivize participants to use it. In order to accomplish this goal, the solution needs to make participants willing to provide their sensible information to the platform and to be able to gather information from the platform that they can trust, published by other participants.

The chosen platform to augment was MISP [2] since: its free and open source software; has features that assist users in creating, collaborating and sharing threat informations such as flexible sharing groups, automatic correlation of attributes and event distribution; supports multiple export formats; supports several export formats that can be used other tools; and provides a flexible API to integrate MISP with user-made solutions [4], [11].

A. Architecture

To augment MISP, our solution takes advantage of the feature that allows MISP instances (also called MISP communities) to be synced with each other, allowing a steady increase in the scale of the sharing without compromising the identity of the event publisher.

Since MISP follows a peer-to-peer model, it allows multiple connections with different MISP instances and each connection can have custom rules for what events are pushed or pulled, each instance can participate in multiple sharing communities providing enough scalability and flexibility to accommodate future needs.

Finally, it is possible to avoid losing information due to anonymization efforts by relying on the MISP feature that allows different distribution settings to different attributes for the same event. MISP supports four different settings for the distribution [11]:

- **Your organization only** - distribution is limited to members of the organization of the publisher. Events and attributes with this setting will not be synchronized to other MISP communities;
- **This community only** - distribution is limited to users registered in a MISP instance, all members of all organizations that are part of that instance can access the event. Events and attributes with this setting cannot be pushed to other instances and only organizations registered in both instances and with pull permissions can obtain the information, however the distribution information is automatically downgraded to Your organization only, in order words the distribution of the information is set to Your organization only after being pulled;
- **Connected communities** - distribution is limited to all users in MISP instances connected to the instance where it was published and to users of the host instance. Events and attributes with this setting can be pushed or pulled but are automatically downgraded to This community only
- **All communities** - distribution is not limited, and the information is freely propagated from one server to the next. Events and attributes can be pushed or pulled without restrictions or downgrades.

By keeping the distribution of sensible information limited to the publisher organization of the event (Your organization only) no information is lost for the publisher organization and minimum utility is lost for further participants.

In order for the MISP information exchange to be automated, it must have first something to share. As a source of events it was proposed to have a network vulnerability scanner OpenVAS (OpenVAS host) which executes continuous scans of each machine of the network (Non-Functional Requirements (NFR)1).

After a successful scan, the OpenVAS host should store the scan results in a structured file format, which can be parsed and imported by a MISP instance. Then, the OpenVAS host sends a message to a predetermined machine hosting the MISP
instance of the organization (MISP host), informing it that the OpenVAS host has results ready to be imported. Finally, the process is repeated but the target host changes in order to provide analysis from a different machine. This process is illustrated in figure 2.

After receiving a message from OpenVAS indicating that it completed a scan and that it has a collection of results ready, the machine hosting an instance of MISP gets the file containing the results, parses the results into separated events and resorts to anonymization techniques to filter sensible information from the same events. This is done by the OpenVAS to adapter represented in a different color in the figure 2.

Once all the results are parsed into events and all its fields are filtered, these events are imported to the MISP instance and stored. In the end, the MISP instance of the organization should send a message to the next external (outside the organization’s control) instance of MISP in the hierarchy notifying that its database was updated recently. This process is also illustrated in figure 2.

B. Implementation

To implement a solution that integrates OpenVAS with MISP the implementation had to: communicate with the both MISP and OpenVAS, adapt the data of the reports received from OpenVAS to import it to MISP as events, and anonymize the results from OpenVAS to pass as events; had to be found.

To solve this, a Python script was developed. Python was chosen as the script language since it is a flexible general-purpose language that has a good diversity of libraries available and, among them, PyMISP\(^6\) [2] and lxml\(^7\). In short, the script issues a OpenVAS scan on the target subnet or host(s), waits for the scan to finish, gathers the results from the report obtained from OpenVAS, and, finally, transforms the results into MISP events.

The script developed also provides a configuration file named config.py that allows a user to easily change some of its variables. These variables allow the definition of:

- Which ports, scanning configurations, targets, tasks or hosts should be used during the vulnerability scanning as well as the creation of new targets or tasks in case they do not exist;
- The minimum base Common Vulnerability Scoring System (CVSS) score each event needs to have in order to be shared with other communities, as well as a customizable value to determine the level of threat (low, medium or high) of the event based on the base CVSS score;
- A boolean that determines if the anonymization of attributes from results should be active or not, if this feature is activated a blacklist of names or regular expressions is provided to hide occurrences of the same name or anything else that matches a regular expression, the blacklist is case insensitive;
- The IP address of the host of the script, reachable by OpenVAS instance, and a port not used or blocked by the host itself or the network in order to “wake up” the script when the OpenVAS scan ends;
- The configurations of the MISP instance, including its host address, the desired user API authentication key\(^8\) and the Secure Socket Layer (SSL) certificate of the instance.

The script used the module OpenVAS Connector (III-B1) to communicate with OpenVAS, issue scans and import results; the description of each event was anonymized with pseudoanonymization (III-B3); and for each result an event was created in MISP and the parameters of each result were split in attributes and added to the same event. Each event and attribute had a distribution setting (III-B2) and tags (III-B4) dependent on its sensitivity or base score.

1) OpenVAS Connector: is a Python module used to manage OpenVAS Servers using the OpenVAS Management Protocol (OMP) 7.0\(^9\) through the OpenVAS Command Line Interface (OpenVAS-CLI). It was implemented as part of this work because the official OpenVAS Python module(openvas.omplib) was not compatible with most recent versions of the OMP nor with Python 3.

OpenVAS Connector relies on the OpenVAS-CLI command omp --xml=’ where the argument xml is attributed to XML data in the format of plain text which is then parsed as a command and in the lxml module which provides a range of tools for XML processing.

2) Distribution settings: The distribution settings (the same distribution settings explained in the section III-A) of the events depends on the CVSS base score of the result they

---

\(^6\)PyMISP is a Python library to access MISP platforms via their REST API.

\(^7\)lxml is a Python library to process data in the eXtensible Markup Language (XML) or Hypertext Markup Language (HTML) formats.

\(^8\)An API key is the name given to some form of secret token which is submitted alongside web service (or similar) requests in order to identify the origin of the request. The key may be included in some digest of the request content to further verify the origin and to prevent tampering with the values.

\(^9\)http://docs.greenbone.net/API/OMP/omp-7.0.html
were based on when compared with the \texttt{MIN\_CVSS\_BASE}
of the configuration file: if the CVSS base score is lower than
the minimum score, then the distribution of the event is limited
to the organization. If not, the distribution of the event is set
to all connected communities. In other words, the event is
propagated to all MISP instances connected to the instance
that creates the event.

Attributes can also have their own distribution, however this
distribution setting is limited by the distribution setting of the
event that contains the attribute, for example, a attribute can
have its distribution setting set to all communities (unlimited),
but if the event that contains it has its distribution limited to
the organization only, the attribute is not distributed freely. The
distribution of attributes is useful to limit the distribution of
more sensitive attributes, keeping them within the organization
even if the event is shared.

The distribution of the attributes is always the same:
\texttt{target\_machine} (or \texttt{host}) and \texttt{pseudonyms} (III-B3)
is always limited to the organization; while the rest of the at-
tributes are shared to connected communities. This maximizes
the utility of the information shared while keeping information
that might jeopardize the organization within itself.

3) \textit{Pseudoanonymization}: If the \texttt{ANONYMIZATION} flag in
the configuration file is set to \texttt{True}, when extracting the
description from the XML report all occurrences of
domain names, emails or words in the blacklist of the config-
uration file are replaced by pseudonyms. These pseudonyms
and their corresponding occurrences are added as key-values
to a dictionary called \texttt{pseudo\_dict} which is then added
to the dictionary of attributes. The distribution setting of these
attributes (pseudonyms) from the \texttt{pseudo\_dict} is then
limited to the organization so it stays within the its borders
even if the event is shared.

4) \textit{Tagging}: MISP from version 2.4 onwards supports the
usage of tags to create arbitrary links between attributes and
events as well as filter events based on the same tags allowing
both manual analysis or automation processes.

In order to take advantage of the tags, the developed script
adopts existing taxonomies in MISP to tag all generated events
and attributes based on their values. In particular, it uses
Traffic Light Protocol (TLP) amber and green to classify
events and attributes restricted to the organization and shared
to connected communities respectively; TLP Chatham House
Rule was a extension let participants know that the reporter of
the information must not be disclosed which was one of the
requisites of the work and is applied to all events and attributes
by default; the source type tag is applied to all events, letting
users know that the event was created autonomously; and
finally each event is tagged with the corresponding MISP
threat level based on the severity of the OpenVAS result in
its origin. The following tags were used in our solution:

- \texttt{TLP:AMBER} - Information exclusively given to an
  organization; sharing limited within the organization to
  be effectively acted upon;

- \texttt{TLP:GREEN} - Information given to a community or a
  group of organizations at large. The information cannot
  be publicly released;

- \texttt{TLP:EX:CHR} - Information extended with a specific tag
called Chatham House Rule (CHR). When this specific
CHR tag is mentioned, the attribution (the source of
information) must not be disclosed. This additional rule
is at the discretion of the initial sender who can decide
to apply or not the CHR tag;

- \texttt{osint:source-type-automatic-collection} - Event origi-
nated from automatic analysis including dynamic analysis
or sandboxes output;

- \texttt{misp:threat-level="risk"} - Threat level of event depend-
ing on the circumstances of the event. In this case, the
threat level depends on the CVSS base score of the
original result of OpenVAS. The “risk” can be “no-risk”,
“low-risk”, “medium-risk” or “high-risk”.

C. \textbf{Summary}

This section contains the description of the solution to
automatically generate events by integrating OpenVAS with
MISP. After an initial setup, this solution is able to feed
MISP instances based on the results obtained through the
OpenVAS scan over the targeted hosts. It not only provides all
the information obtained to the organization allowing internal
processing of the data but also limits the distribution of
more sensitive attributes and shares events according with the
configuration file automatically. The usage of tags also allows
the analysis of events with higher granularity.

IV. \textbf{Evaluation}

This section describes the testbeds used to develop and
measure the solution (IV-A) and the results of the performance
obtained (IV-B).

A. \textbf{Testbeds}

In order to develop and test the proposed solution that inte-
grates MISP and OpenVAS a target environment - with hosts
and network - was required. Networks with some complexity
were required. Therefore, two distinct testbeds were used to
help to prove the viability of the concept as well as to measure
performance metrics of the platform.

1) \textit{SmallCo}: A fictitious company was created - we called
this company “SmallCo”, an abbreviation of Small Company
- to provide a practical testbed (in the sense that it is easier to
test and debug) for the planned system a virtual network with
several machines for a fictitious company was created.

The virtual machine monitor (or hypervisor) used to simu-
late the system was the VMware Workstation Pro 12.5.8 build-
7098237 on a Windows 10 Pro (version 1709) PC with a Intel
Core i5-7600K CPU @ 3.8 GHz and 16GB of RAM, with
a Local Area Network (LAN) segment called “SmallCoLan”
connecting the guest machines within the network and bridged
adapter connected to the network of the host, simulating the
Internet or another Wide Area Network (WAN).

In our scenario, SmallCo is a small clinic that hosts their
own web server that lists medical consultation prices and
allows patients to make appointments. The architecture of

5
this network consists in 5 machines: 2 workstations, 1 web server, 1 security server and 1 gateway. The figure 4 shows the topology of the SmallCo network when it was scanned with OpenVAS (only one workstation was online).

2) NAV Portugal: NAV Portugal E.P.E. (Navegao Aria de Portugal) is the Air Navigation Service Provider both on mainland Portugal and in the autonomous regions of Azores and Madeira. NAV is responsible for service provision on both Lisboa Flight Information Region (FIR) and Santa Maria FIR (as shown in the map of figure ??). The main mission of the company is to ensure a safe and efficient provision of Air Navigation Services, contributing for the creation of value and well-being for the society, while assuming a paramount key role in the aviation sector [12].

NAV gave us access to a subnet in its Test and Development System (SDT), a platform that replicates its real systems, which allowed us to have a larger testbed to test our solution on.

B. Performance metrics

This subsection describes what measurements and how they were made as well as a discussion of the results.

1) OpenVAS Scanning Duration: In order to test the average time OpenVAS takes to complete a vulnerability scan in a subnet, multiple scans with the default settings and different tasks were made. The only differences between the tasks were the network masks of the target subnet, the less bits in the netmaks, the more hosts are targeted in the scan.

All these scans were made over the virtual LAN in the SDT of NAV to use the same hardware specifications and keep the results comparable. The SmallCo testbed (section IV-A1) could not host the number of machines necessary to perform these measurements.

From the results obtained it is possible to conclude that the duration of a regular OpenVAS scan scales linearly with the

Both figures 6 and 7 show the average time to complete a scan when comparing with the number of hosts alive in the former and with the number of bits in the netmask for the later. The time scale is set to hundreds of seconds.
number of hosts with a small error margin where OpenVAS checks if hosts belonging to a target subnet are alive or not.

2) MISP Event Loading: Our next experiments were designed to compare the time it takes to load events from a report coming out of OpenVAS to MISP, with and without anonymization. A XML file of a previous report was modified to have 100 results and used as an input of 5 sets of 10 consecutive report uploads to MISP for both non-anonymized and anonymized events. The instance was reseted after each set.

During the initial testing, it was noticed that consecutive report loadings to MISP were slowing it down considerably, needing a reset after each set. It was hypothesized and confirmed through another set of measurements, and later with an issue in the MISP bug tracker that the main culprits of the decrease in upload speed of the report were the correlations between the attributes of each event. Since the report had always the same results, everytime a new report uploaded a new event that was already published MISP would correlate its attributes with all the attributes of the published event(s), proving the system unreliable due to scalability issues in this situation. The solution found was correlating only the Network Vulnerability Test (NVT) name and the target-machine since the utility gained from correlating all attributes was minimal and not worth the trade-off in performance because the same NVT always generates the same NVT attributes so correlating its name is enough and the tagging system is able to categorize more efficiently the CVSS base score.

The chart in figure 8 shows the difference between correlating all attributes and just correlating the target-machine and NVT name. The Y axis represents the average time it took 10 reports with 100 events each loaded consecutively to complete during 5 sets. The X axis represents the position of the report insertion in each set, ranging from the 1st report to the 10th report uploaded.

Fig. 8: Average time of loading consecutive anonymized reports to MISP

The bar chart in figure 9 represents the average time of the sets for anonymized and non-anonymized events with correlation in all attributes and for anonymized and non-anonymized events with correlation only in target-machine and NVT name take to load a report with 100 results to a MISP instance.

Fig. 9: Average time loading a report of 100 results to MISP

From the results obtained it is possible to infer that the payload of anonymizing events of a report is insignificant compared to the time it takes to load an event (a payload of around +3%); it was also possible to observe how much correlations can slow down the system without a proper selection of which attributes should be correlated or not. An alternative approach to this problem from the one taken is to reduce the maximum number of correlations per attribute in the MISP configurations.

C. Summary

This section presented an evaluation of performance of the implemented solution.

With the experiments made, it was possible to observe that while the solution has a minimum additional payload (around +3%) the solution loses performance fast in the current version of MISP (2.4) if too many results have attributes that can be correlated with each other. To mitigate this issue the number of attributes that can be correlated was reduced to only the most indispensable attributes, to mitigate the issue even further the number of maximum correlations in the MISP configuration can be reduced.

V. Conclusion

The work described in this article tackled one of the main problems in threat sharing: how an organization can share information automatically while still keeping its sensitive data under its control. The document started by doing a study of the current state of the art of threat sharing. It described how is security information captured and exchanged, and what are the current challenges for the exchange of cybersecurity data.

While existing tools can provide a working platform for the exchange of security intelligence, they do not possess the means to provide enough trust for the users nor to facilitate the integration of independent sources of events from the platform. In our work we augmented MISP by integrating the OpenVAS vulnerability scanner, allowing MISP events to be created from OpenVAS reports autonomously, publishing and distributing them to other MISP communities while limiting the distribution of sensible information. The implementation is configurable so it can fit the requirements of each organization.
The implementation was evaluated with two testbeds, a synthetic one, called SmallICo, and a real-world one, NAV SDT. The evaluation shows that the solution meets the requirements defined in section III-A, and that the data control mechanisms, including the anonymization through pseudonyms, is efficient (results in subsection IV-B2).

With our solution it is possible to participate in an threat sharing platform autonomously without the organization taking risks about the exposure of its identity and infrastructure.

A. Achievements

As a result of this work, the following achievements were accomplished:

- A Python connector module for integrating with OpenVAS, relying on the OMP, was developed from scratch to create and get OpenVAS objects (such as tasks, targets or reports), start and stop tasks, and wait for an alarm. The module will continue to be developed in the future as an open source project10;
- The solution integrates OpenVAS and MISP seamlessly, transforming reports generated in OpenVAS into MISP events with categorization through tags and correlation mappings of attributes that allow related events from the same or other organizations to be found;
- The solution is also able to anonymize sensible data while minimizing utility lost and with a minimal payload on the average event loading time.

B. Future Work

In regards to the implementation of the solution, the following points can be improved the in the future:

- Implement delegation of events through the MISP API when available. This would be able to conceal the event publisher when pushing events into other communities by keeping a common organization in all communities to which events could be delegated to before sharing with other instances. However, the publisher organization would still need to have a copy of the event with distribution limited to the organization for internal analysis and to keep sensible data within the grasp of the organization;
- Extension of the future gamification of MISP to incentivize organizations to participate positively in the sharing platform. The gamification of the MISP platform is a planned feature to encourage users to contribute by offering recognition for their efforts and keeping track of the contributions in a profile [13]. However, this system is planned for individual users only;
- Integration of other tools that are able to provide information security data instead of just integrating the vulnerability scanner OpenVAS, e.g. the vulnerability scanner Nessus11 or the Security Information and Event Management (SIEM) Splunk12;

- Improve performance. A specific improvement would be to create events with attributes and tags locally and do a single Hypertext Transfer Protocol (HTTP) POST request instead of multiple HTTP POST requests to create empty events and add attributes and tags one-by-one. This is possible in the newer versions of PyMISP [14];
- The system would benefit from further validation, with use cases involving more companies.
- The full lifecycle of the security incident response - from detection, through mitigation, and until final resolution - should also be assessed together with the impact of a MISP-based information sharing. Ideally we would compare similar organizations handling the same problems, one with MISP, the other without it.

References