Security Incident Response Automation for xPON Networks

Vaclav Oujezsky, Tomas Horvath, and Martin Holik

Original scientific article

Abstract—**This paper presents a developed tool for automated security incident reporting in passive optical networks. This tool interacts with our programmable development card, developed detection modules, and TheHive project. The custom implementation of the solution has resulted in anomaly reporting templates for xPON networks that can be universally applied and new definitions of indicators of compromise. The custom implementation consists of a collector and middleware layer between the programmable card and Apache Kafka.**

Index Terms—**Automation, CERT, Incidents, Reports, SIRAP, Tool.**

I. Introduction

THE constant need to respond to network incidents in real-time has led many security teams and vendors to develop comprehensive security incident reporting systems to respond effectively. In the past, systems were decentralized, and there were no definitions of critical infrastructure. Later, a need to deal with events in critical networks arose. This approach is the only way to defend against an ever-increasing number of network attacks. Gradually, modes of standard communication and formats of transmitted messages are beginning to emerge.

With the increasing popularity of PONs (Passive Optical Networks), there is a growing risk that attackers would target these networks or active elements of a given infrastructure. The problem of potential security incidents has currently been primarily addressed only at the level of ethernet networks. Internet providers, however, should be able to respond to security incidents already in access or metropolitan networks, which may also be vulnerable to cyber-attacks. So far, the possibilities in this area are minimal. The standards defining PONs serve only as recommendations and assume a certain level of security based on the nature of the technology and the transmission medium.

Nevertheless, these networks are no exception. Currently, several security risks have already been identified and documented. Thus, the reason to manage and analyze security events in passive optical networks is substantiated. Available tools that enable real-time analysis of PON networks are limited. The main objective of this paper is to present a designed and implemented automatic reporting system of security events from PONs, based on an evaluation of currently available options for reporting such events in general.

Manuscript received April 5, 2022; revised April 11, 2022. Date of publication April 25, 2022. Date of current version April 25, 2022.

V. Oujezsky, T. Horvath and M. Holik are with the Department of Telecommunication, Brno University of Technology, Technicka 12, 616 00 Brno, Czech Republic. T. Horvath is also with the Department of Optical Networks, CESNET a.l.e., Zikova 4, 160 00 Prague, Czech Republic (e-mails: oujezsky@vut.cz; horvath@vut.cz; xholik11@vutbr.cz).

Digital Object Identifier (DOI): 10.24138/jcomss-2022-0033

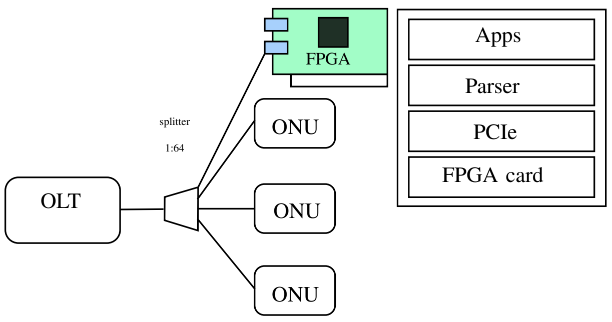


Fig. 1. The basic components of the system.

Today's analyzers, probes, and IDSs (Intrusion Detection Systems) for online services are primarily Ethernet-based. We are working with G.984 [1] and, at present, 10gigabit-capable passive optical network (XG-PON ITU-T G.987). To build a security system, we identified several potential security risks of the PON network, and these are the basis of the developed automated report.

To detect security vulnerabilities in a PON network, we use a system that we have developed within the project. The article's main contribution concerns the custom implementation of the so-called SIRAP (Security Incident Response Automation tool for xPON network), a novelty system designed to report security incidents automatically from the PON network perspective. We also present new definitions of indicators of compromise and compositions of template types for automated reports that are part of our solutions. We have introduced the details about all the inter-operable components of the entire system in [2]–[6]. We use our FPGA (Field Programmable Gate Array) network card developed by our team in cooperation with [7]. This card is connected to the laboratory PON network, see Figure 1. We have developed two types of the card. One type is as a standalone solution forwarding traffic from connected splitter to the development server equipped with Myricom adapter via crafted UDP (User Datagram Protocol) frame and the newest second type plugged directly into the development server with PCIe (Peripheral Component Interconnect Express). This card forwards traffic (PON frames) from the splitter to the development server via PCIe. The development server then contains a software frame parser and applications (modules) for traffic analysis. These

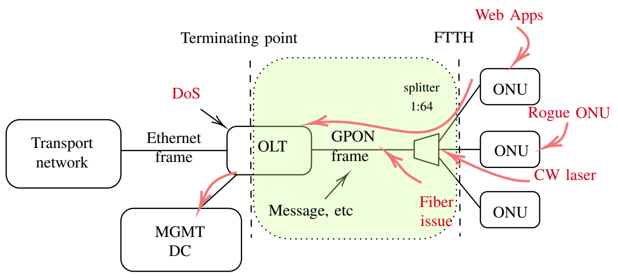


Fig. 2. Security risks in GPON network.

components are also described in our previous publications. We obtain outputs from this system in terms of detected security problems in the optical network, and we store reported incidents in Apache Kafka [8] using our message format. Then the events are processed using custom-defined templates for optical networks and SIRAP. Reports are created and sent to TheHive [9] using the API (Application Programming Interface).

The rest of this paper is structured as follows. Section II. provides an overview of related works, and our motivation and notes some shortcomings in the PON network analysis. Section III. discusses the details of the design of the Security Incident Response Automation tool for the xPON network and the concept of interconnected sub-modules. Section IV. presents the implementation of custom definitions and templates for automatically reporting security events in optical networks, including new security event definitions. The main sections of the source code and diagrams are provided for clarity. Section V. summarizes the functional testing of the proposed solution. Then Section VI. discusses the proposed solution and describes the advantages, disadvantages, and limitations. The conclusion summarizes our achievements and our future plans.

II. The State of the Art and Motivation

Our motivation consists of two aspects. The first aspect is the security risks in PON networks, and the second aspect is the existence of a security tool for PON networks to detect and analyze security risks.

A. *Security Risks in PON Networks*

The security features specified by the standards for the PON network are based on the assumptions that gaining physical access to the optical medium is difficult for an attacker, and eavesdropping on the signal is not a trivial task. Uplink transmission in the two-direction communication of the PON network is therefore considered secure. However, this may not always be true in a real environment. Splitters are commonly installed, for example, in basements where they are not further secured [10]. If some of the ports on the splitter are free, an attacker can easily connect to the network; otherwise, it is enough to disconnect a legitimate user from the network and connect an attacker's device. Several known types of security risks should not be ignored within PON networks. Figure 2 presents some of the security weaknesses of GPON (Gigabit Capable Passive Optical Network).

Passive intercept in xPON networks is due to the nature of the communication. The data in the downlink direction are broadcasted [11], and modified ONU (Optical Network Unit) in promiscuous mode (Rogue ONU) can intercept all communications in the downlink direction. Other optical detectors can also intercept this communication, but the intercepted signal must be further processed [12]. The actual implementation of the interception then depends only on the access to the network, e.g., by using a free connector in a splitter. Modified ONUs represent one of the most significant security risks in the xPON network. For example, they can be used for attacks such as TOS (Theft of Service), Masquerade, or Reply Attack [13]. Detecting these modified units in the network is challenging. The standard [14] generally addresses the principles and techniques that should be in place for the detection, isolation, and mitigation of these units.

DoS (Denial of Service) attacks make a network service unavailable to legitimate users. The blocking of upstream communication occurs when an ONU transmits outside of its allocated time slots. The root cause of a DoS attack also can be a hardware or software malfunction of an ONU. However, an attacker can deliberately modify an ONU to transmit continuously on a given wavelength and with sufficient transmit power to block the communication of other ONUs. The attack can be realized with a sufficiently powerful laser beam source [15]. Testing of the attack has been performed [16], and the results prove that with a sufficiently powerful laser source capable of transmitting at the exact wavelengths, the attacker can block the communication successfully. The OLT (Optical Line Termination) can detect the interfering signal, but there is no adequate protection against the attack. Due to the passive network, it is impossible to accurately identify the source of the interference and distinguish the disturbance from a targeted attack.

Both ONU and OLT are targets of attacks. The implementation of the PON protocol is not unified, and the firmware of devices from different manufacturers may be programmed contradictorily. Thus, such devices may contain security weaknesses. For example, the vpnMentor server published two critical vulnerabilities in DASAN home GPON routers in 2018 [17]. These vulnerabilities are *CVE-2018-10561* and *CVE-2018-10562*. Exploiting a combination of these vulnerabilities allows an attacker to bypass authentication altogether for device management access and subsequent malicious code execution using RCE (Remote Code Execution). Specific exploits are then available, at an example in the Exploit Database. The most recent contribution is code enabling RCE targeting home PON routers, published in 2021 [18].

As part of our research, we also investigated how an attacker could exploit exposed ONUs on the Internet, as well as their "Web App" interfaces, and how it would be possible to paralyze an internal network [19]. We focused on PLOAM (Physical Layer Operation Administration and Maintenance) messages used in the management communication between OLT and ONU. These messages are used to transmit control and monitoring instructions between the OLT and the ONU. We found that some vendors do not follow the standard, and thus, undefined messages are present in these PLOAM messages [20].

The attack can be automated, where a modified ONU can be programmed in such a way to send a message to the OLT to perform an action upon receiving instruction on the ONU Web App interface from a CC (Command and Control Center). For example, OLT is instructed to configure a bridge between the management and the user network. Thus, such configuration may provide the attacker direct access to the operator's management network.

Our motivation is based on the abovementioned security risks. We need an automated tool for reporting security risks for PON networks, not only for TCP/IP-based (Transmission Control Protocol/Internet Protocol) networks.

B. Tools for Processing Security Incidents

Security teams use a variety of tools to detect and resolve incidents. There is no single standard that mandates the use of specific tools, and it depends on the variety of individual teams' decisions. However, organizations such as ENISA (European Union Agency for Cybersecurity) or NIST (National Institute of Standards and Technology) provide teams with specific recommendations and procedures. The tools can be divided into a few general categories, including IDS (Intrusion Detection Systems), IPS (Intrusion Prevention Systems), SIEM (Security Information and Event Management), and SOAR (Security Orchestration, Automation and Response).

IDS systems analyze and monitor network traffic based on known signatures or anomalies. IPS systems extend the functionality of IDS with the possibility of actively blocking and filtering traffic. Then, both IDS and IPS systems generate notifications for further analysis. SIEM systems aggregate and analyze data obtained from various network or system logs. They also use machine learning for the analysis. The IDS builds a predictive model (i.e., a classifier) to differentiate between intrusion or attacks and regular connections. For example, the SIEM comes with prebuilt machine learning for anomaly detection to automatically detect host and network anomalies. Therefore, potential threats can be identified, and alarms can be automatically generated based on this data. SOAR systems collect and centralize data and alarms from different sources. Such an approach enables a holistic view of an ongoing event. They usually integrate additional tools to streamline the incident analysis and the resolution process. They also allow some of the processes to be automated.

The individual tools work with IoC (Indicators of Compromise), forensic data discovered during network or system monitoring indicating potential intrusion or malicious activity. In general, these can be, for example, IP addresses, malware signatures, domain names, malware file hashes, and more [21]. Another example can be unusual activities such as data found in system log entries, unusual network traffic, bundles of data in the wrong place, signs of DDoS (Distributed Denial of Service) activity, etc. IoCs are *not defined for PON networks*. According to our findings, most systems are designed for TCP/IP networks, but little attention is paid to the optical access network. For example, based on the detection of non-standard PLOAM messages we have found, these particular messages could be considered as IoCs. We propose this one and several possible IoC to be used for PON system automation and monitoring.

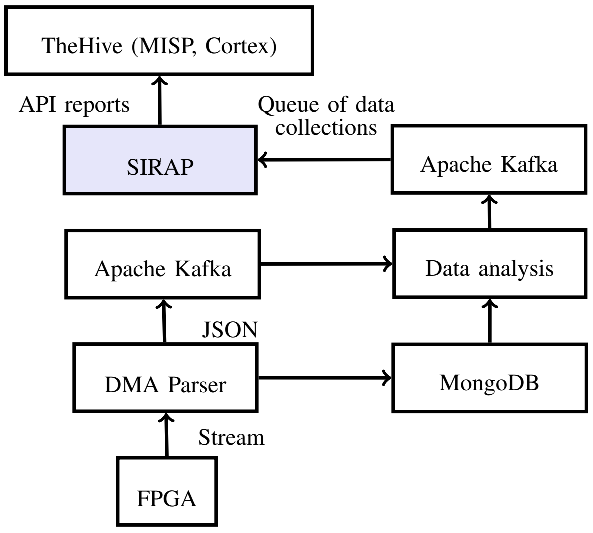


Fig. 3. The diagram of the individual parts of the solution.

III. Proposal of the SIRAP

SIRAP, as we call it, consists of an API design and parts that provide a collection and analysis of reports from PON networks. The SIRAP also uses modules that connect it to existing incident reporting tools. Still, these are primarily designed for ethernet networks. Therefore, it is also necessary to create new and specific IoC identifiers. The SIRAP is a middleware between the rest of our modules' processing data and tools used for reporting.

As mentioned in the introduction, we use our developed FPGA card for data acquisition, already available for commercial use [7]. The Figure 3 shows the cooperation of the individual parts of the whole system with the SIRAP module, which is the subject of this paper. The rest of the components were presented in [2]–[6], [22].

For a better overview, we introduce the basic concept of each interconnected module with the SIRAP. The FPGA card plugged into the PCIe slot of the development server transfers frames from the optical network (downlink and uplink direction) to the server's DMA (Direct Memory Access). The frame parser we designed and developed in C# formats the received frames according to the frame fields and stores them for further processing in the MongoDB database and for direct analysis, which is also in Apache Kafka. The data analysis module is based on TensorFlow, meaning that it is a TensorFlow detector. The analysis of the frames focuses on two areas of the audit:

- *syntax verification* – examining the message to see if it complies with the standard, verification of each field content in GPON header, and whether or not it is similar to patterns from baseline traffic. We use OneClassSVM and the autoencoder.
- *sequence verification* – controls the continuity of individual messages and the content of respective fields between

messages, as well as the analysis of patterns found in message sequences verifying whether the analyzed protocol uses the same message in the same order and with similar content. We use LSTM (Long Short-term Memory) and the autoencoder.

After the detection module analysis, this module sends the identified events to Apache Kafka. It reads SIRAP events and formats the messages according to the developed templates before sending them to the API's event reporting system.

After evaluating the available options, the TheHive project is selected for the subsequent implementation into our solution as an endpoint for reporting security incidents – specifically, a version of the system designated as TheHive4 [9]. The main selection criteria are the possibility of creating custom indicators and a solid connection with MISP (Malware Information Sharing Platform) and Cortex systems [23]. With the ability to create custom indicators, data types can be defined. The advantage of the connection with the MISP system is the easy sharing of IoCs, both internally and between other security teams. The system is currently widely used by several organizations, particularly in Europe, and the enabling of incident reporting to CERT (Computer Emergency Response Team) using MISP presents interesting possibilities. A basic comparison of the tools is shown in Table I. As mentioned earlier, the main criteria are both customization possibilities and a widely used tool.

The design concept for processing and data storage for PON frameworks is based on the Apache Kafka [8] and the MongoDB technologies. Based on our testing of the write speed to MongoDB, it is determined that this database is not feasible for storing data from real-time traffic. We test the number of GEM (GPON Encapsulated Method) frames we can write and store per second into the database using different approaches. The maximum of GEM frames we reached is $\approx 10^5$. Due to this limitation, we have proposed to use Apache Kafka as the buffer in our solution. The analysis module reads the data from both MongoDB and Apache Kafka, doing an analysis using defined approaches and methods [27] and sending back results to Apache Kafka buffer for the SIRAP. Because TheHive allows

TABLE I
A COMPARISON OF INCIDENT MANAGEMENT TOOLS.

	FIR [24]	*Wazuh [25]*	*Cyphon [26]*	*TheHive [9]*
Programming Language	Python	Python	Python	Python
Installation	Linux Docker	Linux Cloud Kubernetes Docker	Linux Windows Mac OS Docker	Linux Docker
Graphical User Interface	Web UI Dashboard	Web UI Dashboard	Web UI Dashboard	Web UI Dashboard
Database	MySQL	Elasticsearch	PostgreSQL	Elasticsearch
Integration		Elastic Stack	Spunk VirusTotal Snort	Cortex MISP Webhooks
Application Programmable Interface	Python	Python Curl PowerShell	Django Logstash	thehive-4py (Python) Curl
Advantages	Simple to use	Official support	Twitter API IoT data	Templates Custom identificators Community support

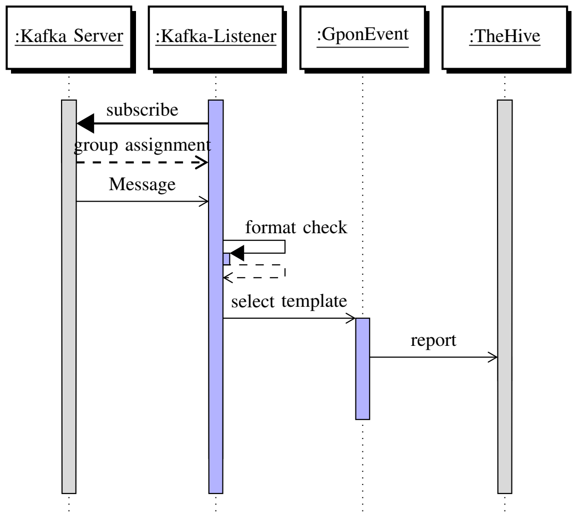


Fig. 4. The sequential diagram of SIRAP.

working with templates, SIRAP includes custom templates designed for specific anomalies defined for PON networks, automatically generating reports for TheHive. These templates and the format of messages for Apache Kafka are created universally to add additional incident types within the open community. The detailed implementation is presented in the following section.

IV. IMPLEMENTATION

This section is focused on the implementation of the SIRAP description, enabling the creation of reports in TheHive system for PON. The application consists of two main modules, an auxiliary core module, a configuration file, and files to enable deployment to Docker solution. The main modules, Figure 4, are the Apache Kafka client `KafkaEventListener` as the consumer and the defining the message template class `GponEvent`. The auxiliary core module `Logger` then defines data classes and formatting of information and debug statements.

First, a connection is established with the Apache Kafka server. The application as a client in the role of a consumer subscribes to receive messages from the topic and, subsequently, messages containing data describing detected security

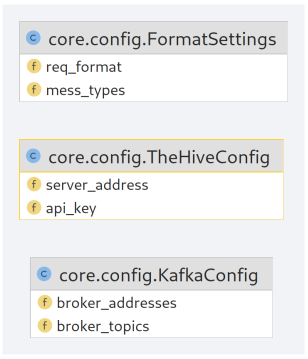


Fig. 5. The configuration core module diagram.

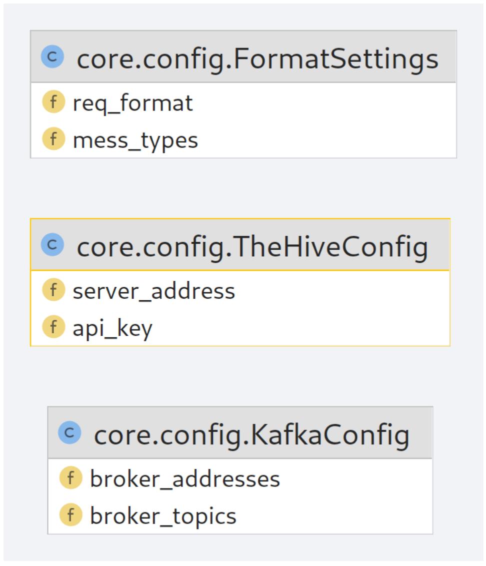


Fig. 6. The main configuration module diagram.

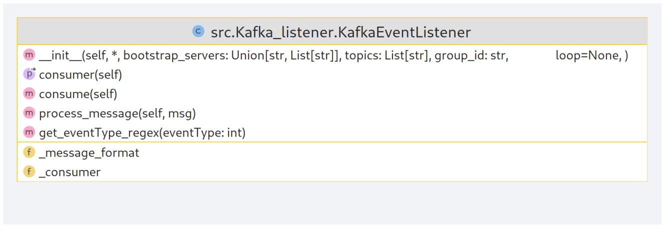

src.Kafka_listener.KafkaEventListener
__init__(self, *, bootstrap_servers: Union[str, List[str]], topics: List[str], group_id: str, loop=None,)
consumer(self)
consume(self)
process_message(self, msg)
get_eventType_regex(eventType: int)
_message_format
_consumer

Fig. 7. The KafkaEventListener module diagram.

events. When a message is received, its format is first checked. A report is generated based on the message type according to the defined templates if the format of a message is correct. The report is sent to the TheHive system, where it can be analyzed, and further actions can be taken at the operator's discretion.

The core module, shown in Figure 5 [28] declares data classes for the configuration file within the `config.py` file. The various Kafka, TheHive, and message format configuration attributes within these data classes are set. The main configuration file `Config.py` is used to set the application parameters. It uses the core module and declared data classes. The settings of the Kafka server, the general Kafka message format, and the TheHive system are defined in the configuration file, which can be seen in Figure 6.

For Apache Kafka, it is necessary to set the IP address and the port where the server is available and the topic from which messages should be received. The `MsgConfig` data class defines the general format of incoming messages from Kafka, i.e., the required fields and the number of templates are defined.

The IP address and port where the system is accessible and the user's API key needs to be added by a user. The `observable_types` dictionary is an additional setting for the TheHive system. This dictionary binds the type of report (from the defined `GponEvent` templates) to the type of compromise indicator. It is necessary to determine the data type under which it indicates the data for each report type. In the case of using defined indicator data types, they must first be specified directly in the TheHive.

The templates defined in `GponEvent` for TheHive are bound to the `observable` dictionary. Such a dictionary binds a report type to a compromise identifier type. A custom data type is defined under which event data is inserted into the template for each report type. The `KafkaEventListener` module shown in Figure 7, defines an asynchronous Kafka client as a consumer and methods for processing received messages. The `aiokafka` library is used to define the client. The library allows creating an asynchronous consumer that enables multiple consumers to spread the load of topics and allows for more efficient processing of received messages. The class constructor allows initializing the consumer. The IP address of the Kafka server (boostrap_servers), the list of topics, and the group id (group_id) are expected on the input of the method.

The `consume()` method is used to subscribe the consumer to messages from the Kafka server. The consumer establishes a connection to the Kafka server using this method and starts receiving messages if it successfully subscribes to the subscription group.

The received messages are then processed using the `process_message(msg)` method. Incoming messages are in JSON (JavaScript Object Notation) format, and are first uploaded into the dictionary. Such an approach allows working with individual fields of the original message. Subsequently, a message is verified by correlating the contained fields with the defined required fields in the dictionary. The proposed generic message format contains the event type (`EventType`), the super-frame counter (`SuperFrameCounter`), and the data describing the event itself (`IncidentData`). The content of the `IncidentData` field is different for each type of incident and enables the transmission of any number of compromise indicators (artifacts) in JSON format. The generic format is adopted to reduce overhead on the producer side. If the message format is valid, the existence of a message template for the event type is verified. A regular expression matching operations perform the validation. A string of one digit in the range of 1 to the number of defined report templates is searched as `"^[1-" + str(eventType) +"]$"`. If a template for the received message type is defined in the `GponEvent` module, the `GponEvent` template object is initialized. The template is passed the contents of each field of the received message and the source reference. The reference combines the received message origin data from the Kafka server, which is the subject, partition, and offset. Finally, based on the completed template, a report is created in TheHive.

The GponEvent module, which is the diagram shown in Figure 8, serves as a template for creating reports. The class defines all mandatory attributes and methods for creating reports. Within the template, `thehive4py` python library is used for communication with the TheHive system. This library provides access to the TheHive API and methods to manage reports. An API object is defined to establish a connection to TheHive. The interface object is populated with the necessary settings from the configuration file, and thus, there is no need to modify the template.

Reports can be created by using the defined API. The combination of the `type`, `source`, and `sourceRef` attributes must be unique for each report. If a report with the same combination of these three attributes already exists in the system, creating a new report is rejected. The defined reports carry an external `type` since they come from an external source of the `Gpon_Analyzer`, i.e., the FPGA card. A combination of values describing the origin of the message received from the Apache Kafka server is used as a reference. The defined consumer passes this value. The individual attributes are defined using a constructor with input parameters, such as `eventType`, `incidentData`, `superFrameCounter`, and a reference is expected. Some of the attributes are set in general for all report types, and some are explicitly defined for a given report type based on the received `eventType` value.

The SIRAP defines five basic report types so far, but we are continuously working on its extension. These message definitions are definitions of problems (incidents) that the `data analysis` module can identify in the PON network based on PON frame analysis and machine learning, which is presented in [6], [27], [29]. The basic report types are:

- *PLOAMd Anomaly* – notification of anomaly detection in a PLOAMd message, i.e., a deviation from the messages specified by the standard,
- *Activation process anomaly* – notification of anomaly detection in activation process,
- *Non-standard Frame structure* – notification of anomaly detection in GTC (G-PON Transmission Convergence Layer) frames,
- *OMCI Anomaly* – notification of anomaly detection in OMCI (ONU Management and Control Interface) messages or the OMCC (ONU Management and Control Channel) activation process channel,
- *Non-specified error* – the empty report, used as a template for the specification of a new report type.

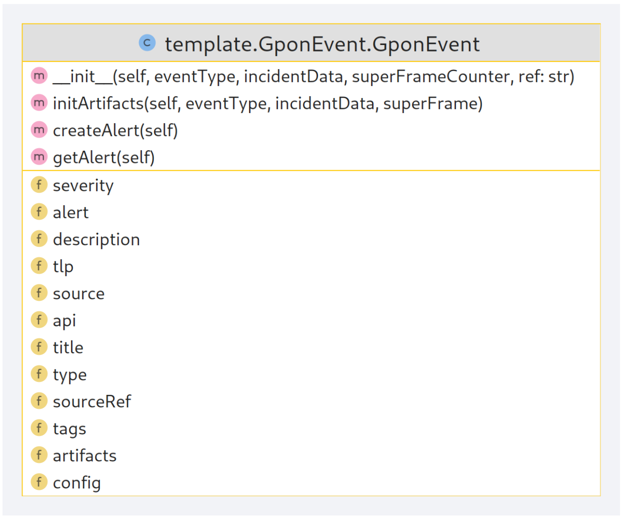

template.GponEvent.GponEvent
__init__(self, eventType, incidentData, superFrameCounter, ref: str)
initArtifacts(self, eventType, incidentData, superFrame)
createAlert(self)
getAlert(self)
severity
alert
description
tlp
source
api
title
type
sourceRef
tags
artifacts
config

Fig. 8. The GponEvent module diagram.

The report template can be easily extended because of the defined empty report, and new report types can be added. The new report also makes it possible to redefine generally defined attributes. In that case, an entry must also be added to the `observable` dictionary with the label of the new `eventType` value associated with the required data type of the indicator of compromise (artifacts) in the report.

For filling the artifacts attribute into the TheHive system, the `initArtifacts()` method is used. The application adds two indicators to the report: the detected data and the `superframe` value.

The datatype parameter at the input of the `AlertArtifact()` method specifies the datatype of the indicator in the TheHive environment. For the application itself, custom-defined datatypes are used. The list of defined datatypes is given in the configuration file. These are not data types at the programming language level; they are indicator labels in the TheHive environment for further analysis. In general, only data in string format can be passed. However, it is possible to pass multiple values within the JSON format. Thus, generic data types are defined for each type of report. A set of detected indicators in JSON format is then passed as data. All data to populate the indicators of compromise are given to the template by the defined consumer from the received report.

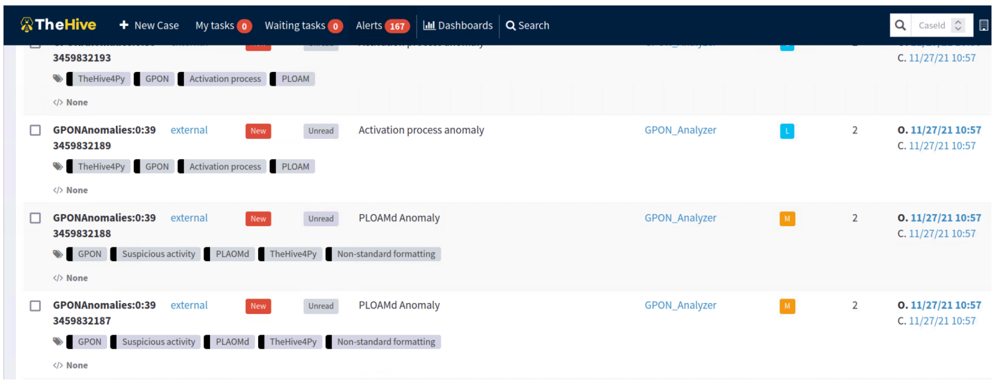
The proposed reporting types are based on possible scenarios in a PON network. The individual events and the content of the defined messages are based on the previously analyzed anomalies. The severity of incidents has been created somewhat experimentally for the time being. However, after consideration, the individual report attribute values can be redefined. The creation of the reports is based on the general form of a message transmitting data describing an event. Thus, it is possible to pass arbitrary indicators on the producer side to the application. The condition is that the parameter's content transmitting this data (`IncidentData`) is in JSON format. The reports are all embedded under one data type defined for the report. However, it is possible to modify the template and add individual indicators from the parameter (`IncidentData`) to the report under its data types.

The messages are created by the `createAlert()` method. The API implements the creation of a report using its `api.create_alert()` method, which is passed the defined report object.

An asynchronous `main()` function is defined to run the SIRAP that reads the configuration file and sets the format of the Logger information dumps. In addition, the client is initialized as a consumer, the server address and topic are taken from the configuration file. The application uses an asynchronous Apache Kafka client. Thus, the main method executes the defined main function in an asynchronous loop.

V. Testing

The testing within this article is focused on the SIRAP module, i.e., the functionality of reading information from Apache Kafka, creating messages according to the information content, and sending them to the server with the TheHive



3459832193 C. 11/27/21 10:57
TheHive4Py GPON Activation process PLOAM
None

GPONAnomalies:0:39 3459832189 external New Unread Activation process anomaly GPON_Analyzer L 2 O. 11/27/21 10:57 C. 11/27/21 10:57
TheHive4Py GPON Activation process PLOAM
None

GPONAnomalies:0:39 3459832188 external New Unread PLOAMd Anomaly GPON_Analyzer M 2 O. 11/27/21 10:57 C. 11/27/21 10:57
GPON Suspicious activity PLAOMd TheHive4Py Non-standard formatting
None

GPONAnomalies:0:39 3459832187 external New Unread PLOAMd Anomaly GPON_Analyzer M 2 O. 11/27/21 10:57 C. 11/27/21 10:57
GPON Suspicious activity PLAOMd TheHive4Py Non-standard formatting
None

Fig. 9. The TheHive GUI with received alarms.

application. The results of testing the remaining parts of the whole concept, such as the FPGA card, parser, and artificial learning module, have already been published in the references provided.

The testing is performed on a real GPON network. It is a polygon from ONUs and OLT. A computer communicating to the Internet is connected to the ONU. The FPGA card captures the traffic and forwards it to the Parser module, which parses the data into the individual fields of the GPON frame and then sends it for analysis. If the parser module evaluates an error or anomaly, it sends a message in a predefined JSON format to the Apache Kafka queue. The SIRAP then reads the events from the queue and processes them.

Created reports are available through TheHive's web interface under the *Alerts* tab. The received reports are displayed in a list, which can be seen in Figure 9. Successfully received incident reports of the `PLOAMd Anomaly` and `Activation process anomaly` can be seen. It is then possible to filter the reports according to the specified criteria in the list. Using *Preview and Import*, the report content and detailed information can be viewed, including the contained indicators that have been defined in the templates, as seen in Figure 10. In the detailed report, the overview can be seen. It includes the description of the detected event, the source of the report, and especially the discussed indicators of compromise defined for the PON network.

The stability of the implementation is also reviewed. Each part of the solution is stable over the long term without substantial outages. As each part of the system uses a different database (SIRAP, Apache Kafka, TheHive, MISP), the whole solution is quite a memory-intensive $\approx$ 21.84 GB RAM (Random-Access Memory), 1.1 GHz CPU (Central Processing Unit). Minor problems can occur when migrating data between versions when deploying TheHive using Docker. The stability of the system and the application are also tested in case of a high number of events. Simulated security events messages are periodically sent to the Apache Kafka server during the test. The SIRAP receives and processes approximately $\approx$400 messages per minute. The messages produced as part of the message processing are displayed within seconds on the TheHive system's web interface. The slight delay is due to a list refresh after receiving a certain number of reports. The refresh rates can be adjusted as necessary. However, such a large number of security events at one time is not usually expected, but systems are nevertheless capable of handling such a load.

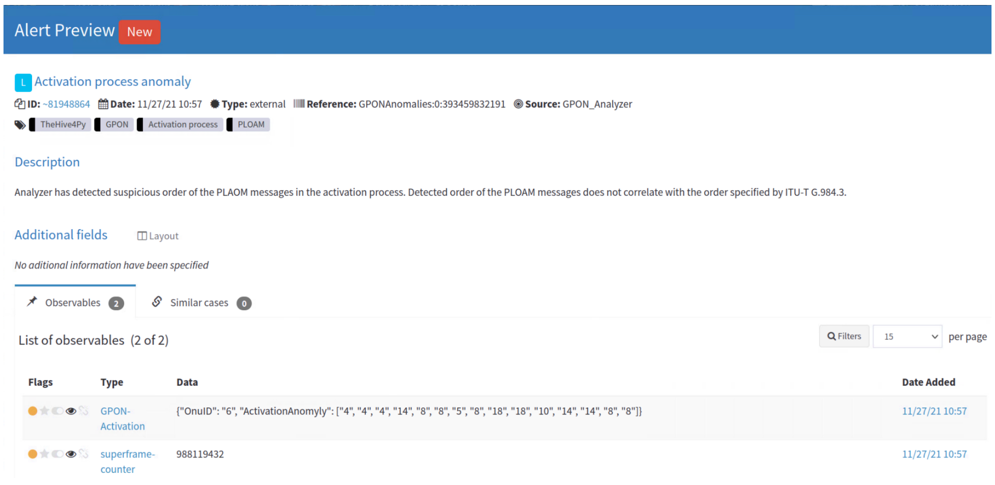
When automatically exporting an event to the MISP system, there may be a problem with defined indicator data types. MISP requires a standard data format for the indicators of compromise and specifies standard categories. However, it is possible to export indicators under the MISP format manually and add them manually to an event in the MISP system using the menu *Add attribute*. Nevertheless, it is not a very automated function for now.

VI. Discussion

The limitations of the system need to be discussed. In principle, SIRAP is not limited to being used with the FPGA card and our analysis module if another system uses our incident reporting templates. However, it primarily represents one part of our entire solution under development, which includes the FPGA card, the data parser, and the analysis module. The last-mentioned module analyzes and generates specific reports to send to Apache Kafka for SIRAP. If another such tool is used, there is no problem using SIRAP for other purposes as a middleware.

However, the limitation remains on what we can detect and report. We are primarily concerned with verifying GPON frames and whether the PON standards are being followed. We are not concerned with the user data within the frame in the data field that, for example, encapsulates ethernet. We are also concerned with anomaly detection directly on the optical fiber. There is no comprehensive database of "vulnerabilities" or anomalies for these specific problems yet. Therefore, our report sets and templates are based on our testing and our findings.

Another factor is the speed of frame processing and verification. We are already talking about several thousand frames per second in high-speed networks like XG-PON. This requires implementing data analysis as much as possible in the hardware part. There is a specific type of limitation on the side between the data parser that sends data to Apache Kafka and the analysis module, which may not efficiently read the frames in a timely manner, and the queue can get very congested.


Alert Preview New

Activation process anomaly

ID: ~81948864 **Date:** 11/27/21 10:57 **Type:** external **Reference:** GPONAnomalies:0:393459832191 **Source:** GPON_Analyzer

TheHive4Py GPON Activation process PLOAM

Description

Analyzer has detected suspicious order of the PLAOM messages in the activation process. Detected order of the PLOAM messages does not correlate with the order specified by ITU-T G.984.3.

Additional fields Layout

No aditional information have been specified

Observables 2 Similar cases 0

List of observables (2 of 2) Filters 15 per page

Flags	Type	Data	Date Added
●	GPON-Activation	{"OnuID": "6", "ActivationAnomyly": ["4", "4", "4", "14", "8", "8", "5", "8", "18", "18", "10", "14", "14", "8", "8"]}	11/27/21 10:57
●	superframe-counter	988119432	11/27/21 10:57

Fig. 10. The detail of one of the GPON alarms.

There is no problem between the analysis module and SIRAP. It is not expected to have the same number of incidents as the original data frames. However, to avoid problems, it is necessary to consider optimizing and distributing traffic between Apache Kafka and SIRAP. Clustering could also be considered.

Another section for improvement is the section that compares whether there is an existing template for a given incident report. The simultaneous use of regex reduces the performance of the solution. Here it is possible to focus on more efficient features, such as using a tree search strategy.

Since the solution given represents the first possible solution for frame verification and is still under development, further issues may arise.

VII. Conclusions

The article presents the approach, design, and application that enables automated reporting of security events from PON networks with a focus on security and current potential security threats.

A survey of available open-source systems enabling incident management is performed to select suitable systems to be used within the proposed solution. A set of TheHive, Cortex, and MISP applications are chosen based on the performed comparison. The systems are selected mainly because of their integration and the customization options they provide. The proposed set of systems is then used to interact with our application to automate reporting.

The central part of the work includes designing and developing a custom application to enable the creation of security event reports in the TheHive system for passive optical networks, standard G.984 and G.987. The FPGA network card analyzes the communication in the optical link among the ONUs and the OLT, and subsequently, by analysis modules, which pass data to the Apache Kafka server based on detecting a possible security event. The result of our work is a custom application called SIRAP, which represents the middleware between TheHive systems and the analysis tools and the parser, which come from our previous work.

The SIRAP implements the Apache Kafka client as a consumer that retrieves messages from the Apache Kafka server containing a description of the captured events. It then creates reports in TheHive based on the type of event using a custom-created library that serves as a report template. The result of the proposal is thus a custom application that enables the automatic creation of security reports in TheHive for PON networks and brings its definitions and templates of security events in PON networks.

Our future work will focus on solution efficiency, processing speed, and testing of xPON networks to define other possible security weaknesses to extend the defined template types for our system.

Acknowledgment

This article is based upon the grant of the Ministry of the Interior of the Czech Republic, Program of Security Research, VI20192022135, “Deep hardware detection of network traffic of next-generation passive optical network in critical infrastructures” and the project – e-Infra LM2018140.

References

[1] *Gigabit-capable passive optical networks (G-PON): Transmission convergence layer specification*, 1st ed., ITU-T, Geneva, 2014. [Online]. Available: https://www.itu.int/rec/T-REC-G.984.3 (Accessed 2021-10-02).

[2] V. Oujezsky, T. Horvath, M. Jurcik, V. Skorpil, M. Holik, and M. Kvas, “Fpga network card and system for gpon frames analysis at optical layer,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. Budapest, Hungary: IEEE, 2019. doi: 10.1109/TSP.2019.8769054. ISBN 978-1-7281-1864-2 pp. 19–23. [Online]. Available: https://ieeexplore.ieee.org/document/8769054/

[3] T. Horvath, M. Jurcik, V. Oujezsky, and V. Skorpil, “Gpon analyzer - frame parser module,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. Budapest, Hungary: IEEE, 2019. doi: 10.1109/TSP.2019.8768882. ISBN 978-1-7281-1864-2 pp. 748–752. [Online]. Available: https://ieeexplore.ieee.org/document/8768882/

[4] M. Holik, T. Horvath, and V. Oujezsky, “Application for gpon frame analysis,” *Electronics*, vol. 8, no. 6, p. 11, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/6/700

[5] M. Jurcik, T. Horvath, V. Oujezsky, V. Skorpil, and M. Holik, “Gpon parser for database analysis,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. Budapest, Hungary, Hungary: IEEE, 2019. doi: 10.1109/TSP.2019.8768849. ISBN 978-1-7281-1864-2 pp. 347–350. [Online]. Available: https://ieeexplore.ieee.org/document/8768849/

[6] V. Oujezsky, A. Tomasov, M. Holik, V. Skorpil, T. Horvath, and M. Jurcik, “Gpon traffic analysis with tensorflow,” in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020. doi: 10.1109/TSP49548.2020.9163575 pp. 69–72.

[7] “Dfc we make electronics: Cecilie – xpon module,” DFC Design, s.r.o, Brno, 2022. [Online]. Available: https://www.dfcdesign.cz/en/cecilie-xpon-module (Accessed 2022-03-28).

[8] N. Garg, *Apache Kafka*. Packt Publishing, 2013. ISBN 1782167935

[9] N. K. Nabil Adouani, Danni Co. (2022) Thehive project. Cortex. [Online]. Available: https://thehive-project.org/ (Accessed 2022-03-28).

[10] P. Kim. (2016) It security research by pierre. [Online]. Available: https://pierrekim.github.io/blog/2016-11-01-gpon-ftth-networks-insecurity.html (Accessed 2021-11-24).

[11] I. Cale, A. Salihovic, and M. Ivekovic, “Gigabit passive optical network - gpon,” in *2007 29th International Conference on Information Technology Interfaces*. Cavtat, Croatia: IEEE, 2007. doi: 10.1109/ITI.2007.4283853. ISBN 953-7138-09-7. ISSN 1330-1012 pp. 679–684. [Online]. Available: http://ieeexplore.ieee.org/document/4283853/

[12] T. Horvath, P. Munster, and J. Vojtech, “Deployment of pon in europe and deep data analysis of gpon,” in *Telecommunication Systems*, I. A. Alimi, P. P. Monteiro, and A. L. Teixeira, Eds. Rijeka: IntechOpen, 2019, ch. 4. [Online]. Available: https://doi.org/10.5772/intechopen.82679

[13] Y. Yan, S. Yamashita, S.-H. Yen, P. Afshar, V. Gudla, L. Kazovsky, and S.-W. Wong, “Invited paper,” *IET Optoelectronics*, vol. 5, no. 4, pp. 133–143, 2011-08-01. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-opt.2011.0027 (Accessed 2021-11-26).

[14] *ITU-T: SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS: Rogue optical network unit (ONU) considerations*, 2nd ed., ITU-T, 2011. [Online]. Available: https://www.itu.int/rec/T-REC-G.Sup49/en (Accessed 2021-11-26).

[15] D. Gutierrez, J. Cho, and L. G. Kazovsky, “Tdm-pon security issues: Upstream encryption is needed,” in *OFC/NFOEC 2007 - 2007 Conference on Optical Fiber Communication and the National Fiber Optic Engineers Conference*, 2007. doi: 10.1109/OFC.2007.4348474 pp. 1–3.

[16] J. Šimoník and T. Horváth, “Gpon network with modified end unit,” *Elektrorevue*, vol. 2018, no. 4, p. 6, 2018. [Online]. Available: http://www.elektrorevue.cz/cz/clanky/komunikacni-technologie/0/gpon-sit-s-modifkovanou-koncovou-jednotkou/

[17] S. Newman. (2018) Critical rce vulnerability found in over a million gpon home routers. VpnMentor. [Online]. Available: https://www.vpnmentor.com/blog/critical-vulnerability-gpon-router/ (Accessed 2021-11-26).

[18] “Exploit database - exploits for penetration testers, researchers, and ethical hackers,” OffSec Services Limited, 2022. [Online]. Available: https://www.exploit-db.com/ (Accessed 2022-03-26).

[19] V. Oujezsky, D. Chapcak, T. Horvath, and P. Munster, “Security testing of active optical network devices,” in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019. doi: 10.1109/TSP.2019.8768811 pp. 9–13.

[20] V. Oujezsky, V. Skorpil, and T. Horvath, “Gpon frame analysis with artificial immune system,” in *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2019. doi: 10.1109/ICUMT48472.2019.8970923 pp. 1–4.

[21] O. Catakoglu, M. Balduzzi, and D. Balzarotti, “Automatic extraction of indicators of compromise for web applications,” in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW ’16. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016. doi: 10.1145/2872427.2883056. ISBN 9781450341431 p. 333–343. [Online]. Available: https://doi.org/10.1145/2872427.2883056

[22] M. Holik, T. Horvath, V. Oujezsky, P. Munster, A. Tomasov, and S. Valach, “Mongodb database as storage for gpon frames,” in *Sensors*, vol. 20, no. 21, 2020. doi: 10.3390/s20216208. ISSN 1424-8220. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6208

[23] “Misp – open source threat intelligence platform & open standards for threat information sharing,” 2022. [Online]. Available: https://www.misp-project.org/ (Accessed 2022-03-28).

[24] “Fir – fast incident response,” CERT Societe Generale, 2022. [Online]. Available: https://github.com/certsocietegenerale/FIR (Accessed 2022-03-28).

[25] “Wazuh - the open source security platform,” Wazuh Inc., 2022. [Online]. Available: https://wazuh.com/ (Accessed 2022-03-28).

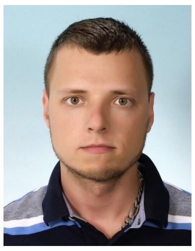
[26] “Cyphon,” Dunbar Security Solutions, Inc., 2018. [Online]. Available: https://cyphon.readthedocs.io/en/latest/index.html (Accessed 2022-03-28).

[27] A. Tomasov, T. Horvath, P. Munster, M. Holik, and V. Oujezsky, “Fpga xpon traffic analysis,” in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 2021. doi: 10.1109/TSP52935.2021.9522610 pp. 58–61.

[28] O. Kupka, “Gpon network security incident reporting software,” Master Thesis, Brno University of Technology, Brno, 2021. [Online]. Available: https://dspace.vutbr.cz/handle/11012/196915?locale-attribute=en

[29] A. Tomasov, M. Holik, V. Oujezsky, T. Horvath, and P. Munster, “Gpon ploamd message analysis using supervised neural networks,” in *Applied Sciences*, vol. 10, no. 22, 2020. doi: 10.3390/app10228139. ISSN 2076-3417. [Online]. Available: https://www.mdpi.com/2076-3417/10/22/8139

Vaclav Oujezsky was born in Brno, Czech Republic. He currently works as an assistant professor at Masaryk University and a researcher at Brno University of Technology, Department of Telecommunications. Working actively on projects of security and transport networks. His research interests include computer networks, network programming, software-defined networking. His focus is on network behaviour, intelligent networks, network analysis, and security.



Tomas Horvath was born in Havirov, Czech Republic in 1989. He is a young researcher at Brno University of Technology and a researcher at CESNET. He received his PhD degree in communications and informatics from Brno University of Technology in 2017. His record shows more than 40 peer reviewed proceedings and journal papers. His current research interests include software-defined optical networking, passive optical networks, and sensing.

Martin Holik is a young researcher and post graduate student at Brno University of Technology, Department of Telecommunications. His current research interests include passive optical networks, database systems, and virtualization.