(19) Octrooicentrum Nederland

(11) **2020552**

(12) B1 **OCTROOI**

(21) Aanvraagnummer: **2020552**

(22) Aanvraag ingediend: **8 maart 2018**

(62) Afsplitsing van aanvraag , ingediend

(51) Int. Cl.:
H04L 29/06 (2018.01)

(30) Voorrang:

(41) Aanvraag ingeschreven:

(43) Aanvraag gepubliceerd:
-

(47) Octrooi verleend:
13 september 2019

(45) Octrooischrift uitgegeven:
15 november 2019

(73) Octrooihouder(s):
Forescout Technologies B.V. te EINDHOVEN

(72) Uitvinder(s):
Elisa Costante te EINDHOVEN

(74) Gemachtigde:
ir. J.C. Volmer c.s. te Rijswijk

(54) **Attribute-based policies for integrity monitoring and network intrusion detection**

(57) A method of detecting anomalous behaviour in data traffic on a data communication network, a first host and a second host being connected to the data communication network, the data traffic on the data communication network forming a link between the first host and the second host, the method comprising:

a) parsing the data traffic to extract protocol field values of a protocol message of the data traffic;

b) deriving, from the extracted protocol field values, attribute values of attributes of one of the first host, the second host, and the link;

c) selecting from a set of models, a model relating to the one of the first host, the second host, and the link, wherein the selected model comprises a plurality of attributes to describe the one of the first host, the second host, and the link, wherein at least one of the attributes is a semantic attribute, the semantic attribute expressing a semantic meaning for the one of the first host, the second host, and the link,

d) updating the selected model with the derived attribute values, if the derived attribute values are not featured in the selected model upon selection;

e) assessing if the updated, selected model complies with a set of attribute based policies, each attribute based policy defining a security constraint of the data communication network based on at least one of the attributes of the first host, the second host or the link, and

f) generating an alert signal in case the attribute based policies indicate that the updated selected model violates at least one of the attribute based policies.

NL B1 2020552

P33194NL00

Attribute-based policies for integrity monitoring and network intrusion detection

The present invention relates to a method of detecting anomalous behaviour in data traffic on a data communication network and an intrusion detection system configured to perform the method.

A data communication network, also identified as a computer network, may be understood as a group of possibly heterogeneous devices (e.g. personal computers, tablets, phones, servers, controllers, actuators) that are connected together and can exchange data with each other over a communication channel. Companies rely more and more on computer networks to develop and maintain their core business. Likewise, Industrial Control Systems (ICS) are also increasingly adopting Information Technology (IT) technologies to improve the efficiency of their processes. Although this trend improves efficiency and business possibilities, it also increases the exposure to cyber-attacks. Intrusion Detection system (IDS) are widely-deployed security tools used to detect cyber-attacks. A network-based IDS (NIDS) relies on the analysis of network traffic for its detection technique.

Typically, a network monitoring system, such as a Network Intrusion Detection System can be distinguished in: i) blacklisting systems and ii) whitelisting systems.
Blacklisting systems maintain a database of well-known attacks and raise an alert when malicious network events that match a known attack are detected. These systems have the advantage of exhibiting very few false positives. On the other hand, blacklisting systems are limited in the sense that they can only detect well-known attacks for which a well-defined specification is available. This means they are unable to detect 0-days attacks, namely attack that exploits vulnerabilities still unknown. Further, the number of signatures that must be managed and checked increases with the number of attacks.
On the other hand, whitelisting systems maintain a *model* of the normal behavior of a system and compare the current activities with it: in case a mismatch occurs, an alert is raised. Whitelisting solutions are becoming quite popular since they have the great potential of detecting both known and unknown attacks. Unfortunately, they typically generate a high number of false positives which are expensive to handle. Typically, whitelisting NIDSes adopt

an approach where a *model* of normal behavior is created (e.g. automatically learned or manually specified) over a few weeks' time period while the detection (that uses such model) goes on forever. This general approach to whitelisting may lead to the following problems. There is no structured way to deal with legitimate changes caused by the intrinsic dynamism of any monitored network where things keep changing, legitimate behavior evolves and new legitimate devices might appear. This lack of adaptability causes two issues: (i) every (legitimate) change generates false positives which are expensive to handle since they require operators' manual inspection; and (ii) the *model* used by the whitelisting system needs to be manually updated for each (legitimate) change in the monitored traffic, an expensive and complex process. Some of the existing solutions allow for small manual updates to the model over time but generally any behavior outside the model would trigger an alert.

The invention aims to provide an anomalous behavior detection that may be adapted more easily to changes, e.g. upgrades, in the network.

According to an aspect of the invention, there is provided a method of detecting anomalous behaviour in data traffic on a data communication network, a first host and a second host being connected to the data communication network, the data traffic on the data communication network providing a link between the first host and the second host, the method comprising:

a) parsing the data traffic to extract protocol field values of a protocol message of the data traffic;

b) deriving, from the extracted protocol field values, attribute values of attributes of one of the first host, the second host, and the link;

c) selecting from a set of models, a model relating to the one of the first host, the second host, and the link, wherein the selected model comprises a plurality of attributes to describe the one of the first host, the second host, and the link, wherein at least one of the attributes is a semantic attribute, the semantic attribute expressing a semantic meaning for the one of the first host, the second host, and the link,

d) updating the selected model with the derived attribute values, if the derived attribute values are not featured in the selected model upon selection;

e) assessing if the updated, selected model complies with a set of attribute based policies, each attribute based policy defining a security constraint of the data

communication network based on at least one of the attributes of the first host, the second host or the link, and

f) generating an alert signal in case the attribute based policies indicate that the updated selected model violates at least one of the attribute based policies.

On the data communication network, data communication takes place between a first host and a second host. The connection between the first host and the second host forms a link between the hosts. The hosts may be any device capable of data communication, such as computers, mobile telephones, wireless devices, programmable devices such as PLC, or any other device. The hosts may take any possible role, such as a server, a client, a printer, a camera, a PLC, an HMI, a SCADA server, etc. The hosts may be a sender or a receiver in the data communication, or both a sender and a receiver. The data communication network may be any data communication network, such as a wired or wireless network and may make use of any type of communication protocol. The data communication (the link) between the first host and the second host may be a one-to-one link, or a one-to-many, such as may be the case when broadcasting data from a first host to plural second hosts.

The terms data traffic, link, network message, network message field, attribute, host model, link model, model update, attribute based policy and alert may be defined as follows:

- **data traffic**: data traffic (or network traffic) is the data moving across a network at a given point of time. Data traffic is mostly comprised of network messages.
- **host**: a network host, or simply host, is an element (computer or other device) connected to a network;
- **link**: a network communication taking place between two network elements. Typically, a link has a host source and a host target of the communication;
- **network message**: an element of data traffic, typically an information unit that is transmitted by one host to another over a network;
- **network message field**: network messages are composed of several fields that carry the message information. To every field it is associated a length (in bit) for its representation, a value and a data type. For instance, source IP (32 bits or 128 bits, depending on the IP version either 4 or 6) and destination IP (32 bits or 128 bits, depending on the IP version either 4 or 6) are examples of fields in an IP message.

- **attribute** an attribute is a feature representing a host, a link or the contextual environment. The attribute is obtained by mapping one or several network message fields (possibly over several network messages at different points in time) to a more abstract category. The attributes describe protocol generic features, i.e. features in terms which are generic, thus not dependent of a specific protocol as may be applied.
- **host model**: a list of attributes used to describe the host wherein at least one attribute is a semantic attribute
- **link model**: a list of attributes used to describe the link wherein at least one attribute is a semantic attribute
- **model update**: the process of mapping network message fields to the host- and the link-models. At every new network message, the model update process applies heuristics and/or aggregation and/or classification techniques to update the current host- and link-models by considering previous values of the attributes and the new information carried by the current network message.
- **attribute-based policy**: a statement in the form if-condition-then-action[obligation], where the condition contains attributes referring to the host and/or link models, the action defines what has to be done in case of a positive match of the policy with the data traffic (e.g., deny or permit) and obligation (optional) defines additional actions to be taken in case of a match (e.g., send email= name@domain.com, set priority=high, set criticality = medium). The attribute based policies define the condition and what has to be done in a protocol generic way, i.e. not specific to the protocol as may be applied in the communication on the data communication network, e.g. the communication between the first host and the second host via the link.
- **whitelist policy**: an attribute-based policy that describes acceptable data traffic (action ==permit);
- **blacklist policy**: an attribute-based policy that describes unacceptable data traffic action ==deny);
- **alert**: semantic-enriched and context-aware information that expresses a possible situation of danger of the system. For instance, an alert can express the fact a certain threat to the network has been identified.
- **policy-check**: the process of matching attribute-based policies against data traffic and take the action described by the policy in case of match.

- **protocol:** a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two and more computers (hosts). Networks follow these rules to transmit data.

The data traffic is parsed, such as by a parser. The parser extracts protocol fields and their corresponding values from the data stream. The parser may relate to any suitable protocol language. The term parsing may be understood as an analysis by a computer of a sentence or other string of words into its constituents, resulting in a parser tree showing their syntactic relation to each other, which may also contain semantic and other information.
Network protocols may include, as non-limiting examples, one or more of the following: low-level protocols such as Ethernet frames, wireless (like WI-FI, Bluetooth or LTE) frames or serial-bus (RS-232/485, CAN bus) frames; IP headers; TCP/UDP/ICMP/IGMP headers; higher-level protocols such as HTTP, SMTP, FIX, LDAP SSL/TLS: protocols intended for carrying industrial data, such as DNP3, MODBUS/TCP, MODBUS/RTU, BACnet, ANSI C12.22, IEC61850; and others.

Attribute values are derived from the extracted protocol field values. The attributes values may be derived from data (protocol fields) at any layer of the protocol. The attribute values express values of attributes. The attributes represent features of the first host, the second host and/or the link. For example, some of the attributes may relate to the first host, some of the attributes may relate to the second host and some of the attributes may relate to the link between the first host and the second host. The attributes values are protocol generic, i.e. the values form descriptions that are not specific to a certain protocol. For example, a read operation is formed by 4 in the Step 7 protocol, 45 in the IEC-104 protocol, and 23, 3 in the Modbus protocol. The term read operation thus forms a protocol generic description of 4 in the Step 7 protocol, 45 in the IEC-104 protocol, and 23, 3 in the Modbus protocol. Hence the attribute values provide generic descriptions of the first host, the second host and the link.
Examples of host attributes are: operating system, vendor, role, firmware, model, applications, services, type of network.
Examples of link attributes are: protocol, source port, destination port, function, message type, payload parameters.
Plural models are provided. One of the models represents the first host, one of the models represents the second host and one of the models represents the link between the first host and the second host. Each model comprises a plurality of attributes, each model may be

formed by a collection of attributes. The models may for example initially be empty Each model may be associated, by an identifier, with a particular host or a particular link. When data traffic has been parsed, providing attribute values of the first host, second host and/or link, the respective model that relates to the respective one of the first host, second host and link is updated by adding the attribute value, in case the attribute value is not featured in the respective model yet. Otherwise the model is left as is, or a level of confidence of the respective attribute value may be increased. The model of a host may include, but not being limited to, one or more of IP addresses, MAC address, MAC vendor, model, firmware, serial number, applications, ports, protocols, services, sent data, received data, role, operating system, first recorded activity, last recorded activity, network, criticality, sensitivity, owner, geographical location, labels, username, agent name, URL(s), etc.. The model of a link may include, but not being limited to, one or more of source port, destination port, Layer 1 protocol, Layer 2 protocol, Layer 3 protocol, Layer 4 protocol, Layer 7 protocol, message code, message type, number of connections, number of bytes, etc., where Layer refers to the ISO/OSI Layers . The models are held against a plurality of attribute based policies. Each policy, i.e. each attribute based policy, may define an outcome in case a condition is met, the condition being defined in terms of one or more of the attributes of the first host, the second host and/or the link. The outcome of the attribute based policy relates to an allowability of the data traffic. Each attribute based policy thus defines security constraints on the basis of one or more host attributes and/or link attributes. For example, if one or more of the policies indicate unallowable behavior, an alert may be raised.

Examples of attribute based policies may comprise: a guest device may not send reprogram commends to the e-mail server. The printer device may not transmit a scanned document to an external e-mail server. The printer device may send operating status data to a remote maintenance printer server. A Windows computer may not act as an e-mail server. A Windows computer running a user profile may not have administrator rights. A Windows computer running an administrator profile may send an update command to an e-mail server. The policies may thus form whitelisting and/or blacklisting policies. The policies define conditions and actions in a protocol generic way. Although the above examples provide the policies in a generic description, them may be drafted in terms of if condition then action. For example, the policy “The printer device may not transmit a pdf document to an external e-mail server” may be noted as “If printer device transmits a pdf document to an external e-mail server” then “generate alert message”. In this example, the printer device and the external e-mail server may form host attributes, while pdf document transmission may refer to link attributes. The set

of policies may comprise policies relating to various hosts, links, etc. Thus, for a particular protocol message, a particular model, etc., only a subset of the attribute based policies may be relevant. The set of attribute based policies may define a security policy of the data traffic on the data communication network.

The outcome of the policies hence provides a result relating to the allowability of the data traffic. If the data traffic is found to be not allowable, an alert signal may be generated. The alert signal may form a warning transmitted to an operator, may form an alert message logged in a database, may form an optical warning (such as on a display) and/or an audible warning (such as a beep or other acoustical alarm) to a user/operator, etc.

The process of parsing, extracting protocol field values, deriving attribute values, building models, comparing against the attribute based policies and generating the alarm signal may be performed automatically, i.e. in an embodiment, the process does not require human intervention.

As the detecting of anomalous behavior is performed on the basis of attribute based policies, the criteria for data traffic to be allowable or not may be defined in terms of attribute values. Hence, the criteria may be defined at a higher level of abstraction, which may provide that the criteria as set by the policies not just apply to one specific device/circumstance, but may have a more general scope. The attribute based policies may apply rules at a higher level of abstraction, and may correspondingly judge events that occur, even when changes have been carried out in the data communication network.

For example, in case an attribute based policy holds that a printer/scanner may not forward a document to an external device, the policy may not just be applied to one specific printer/scanner, but may apply in any situation where a network host is recognized as being a printer/scanner, and a model being correspondingly updated based on the attribute values.

As another example, in case a policy defines that a reprogram command may not be sent from an external device to a PLC (i.e. may not be sent from a device external to a local network), then any host that is modelled as being a PLC may be subject to this policy. Thus, when the network is updated and a new PLC is connected, the policy may apply to this new PLC in case the attribute of being a PLC can be extracted from the new PLC also. Similarly, in case the new PLC appears to make use of a different protocol, e.g. appears to communicate using the IEC-104 protocol instead of for example the Modbus protocol, the existing policy may be applied, despite the fact that a similar operation may have a different coding in different protocols. Thus, in case the attribute based policy would define that a PLC may respond to a read operation

only when the read operation has been sent from a local device, the policy may be applied even when a new PLC which communicates using a different protocol would be found in the network.

Thus, by building models of the host and link which define attributes of the host and link, attribute based policies may asses the allowability of the data traffic at a higher level of abstraction, enabling to define policies at a relatively high level of abstraction. Hence, the policies may be applicable when changes in the network have taken place, as similar or the same attributes may be extracted from data traffic, in the case of some changes to the network. Thus, making use of the attribute based policies for monitoring, intrusion detection, detecting anomalous behavior etc, may promote to detect an intrusion even when a certain degree of changes in the network have been carried out.

In the prior art the *model* of normal behavior is typically defined per single network entity (i.e. host) generally identified by its IP address. With this approach, the presence of a new host would be always initially considered a threat even though it might just be the result of a network upgrade. Although some NIDSs do support the creation of *models* for classes of elements (e.g., allow FTP access to all File servers), the classes of elements need to be defined (and maintained) manually, an expensive and error-prone task. This means that it is not possible to define models for classes of similar elements or elements with similar behavior. This also means that, every time a new host is added to the network, it is necessary to define a new set of models that determine what is acceptable and what is not acceptable for the given host.

Furthermore, using attribute based policies allows a passive observation, thus not introducing additional data traffic and not disturbing a behavior of the hosts and the data network.
As attributes of the hosts and/or links may be used, a behavior (i.e. who does what?) may be observed, as the sender, receiver and the link between sender and receiver may be taken into account, and as the host and link attributes allow to define the policies in terms of behavior of the hosts.

The detecting of anomalous behavior may include one or more of: intrusion detection, discovering of network assets, characterizing network assets, identifying malicious activities by users, identifying malicious activities by network assets, etc. In an embodiment, the detecting of anomalous behavior provides for intrusion detection.

Applications may include the Internet of Things, Home Automation, Building Automation, Industrial Control Systems networks (oil&gas, electric-power generation, transmission and distribution, drinking and waste water, pharmaceutical/life sciences, chemical and petrochemical, entertainment, etc.), Industry 4.0 and manufacturing networks, office and IT networks, data center networks, in-car networks, car-to-x communications.

Some illustrative examples are provided below:

In an Internet of Things application, attribute extraction may be used to understand if a host is a blood pressure sensor, a step counter, a geo-location device and to detect intrusions or malicious activities by matching against attributes-based policies such as 'Patient's biometrics (ECG, heart rate, respiration rate and activity Level) can only be sent to authorized devices' or 'Wearable devices can only have outbound communications'

In a Home Automation application, attribute extraction may be used to understand if a host is a presence sensor, a fridge, an oven, a coffee machine, a thermostat, a smart phone etc., and to detect intrusions or malicious activities by matching against attributes-based policies such as 'A fridge cannot be connected to the Internet' or 'The oven can only be switched on or off from a known smart phone'

In a Building Automation application, attribute extraction may be used to understand if a host is a light, a thermostat, a controller, an IP-camera a card reader etc., and to detect intrusions or malicious activities by matching against attributes-based policies such as 'An IP-Camera cannot receive inbound communication' or 'A light cannot receive more than one switch on/off command per minute' or 'A controller can only be reprogrammed over a secure connection (e.g., SSL, HHTPS)';

In a Industrial Control Systems application, attribute extraction may be used to understand if a host is a SCADA, a PLC, an engineering workstation, an HMI, etc., and to detect intrusions or malicious activities by matching against attributes-based policies such as 'A PLC can only be reprogrammed by an engineering workstation' or 'a PLC can only be queried by a SCADA system';

In an Industry 4.0 and manufacturing application, attribute extraction may be used to understand if a host is a robot arm, a 3D printer, a final product, a controller, a DCS, etc., and to detect intrusions or malicious activities by matching against attributes-based policies such as 'A robot arm can only receive commands from a DCS in the same network'

In an office and IT application, attribute extraction may be used to understand a host is a printer, a workstation, a virtual server, a rack server, a smart phone, an FTP server, etc., and

to detect intrusions or malicious activities by matching against attributes-based policies such as 'A printer can only receive print messages' or 'An FTP server can only receive files from workstation and not from smart phones'

In a car-to-x application, attribute extraction may be used to understand if a host is a vehicle, a traffic light, a controller, speed meter, a pedestrian, etc., and to detect intrusions or malicious activities by matching against attributes-based policies such as 'A traffic light can receive command only from a controller' or 'A vehicle can speak to another vehicle only to communicate its location'

At least one of the attributes is a semantic attribute, the attribute value of the semantic attribute being a sematic attribute value, the semantic attribute expressing a semantic meaning for the host and/or link to which the sematic attribute relates.

The term semantic attribute may be understood as an implicit attribute, it may not be derived directly from a field of the protocol message. An attribute is implicit if it can only be derived by performing an analysis that could possibly involves multiple messages or multiple communications. Implicit attributes can be referred to as semantic attributes, since they allow the semantic-enrichment of the information we have about hosts and/or links. For instance, a host's role, its Operating System (OS) or its Criticality are example of implicit attributes. In fact, to derive the role of a host one should analyze multiple network activities, understand what layer 7 protocols it speaks, whether it only starts or also serves new connections, which ports are used involving that host, etc. In the same way, attributes such as the message type (e.g., whether the link represents a request for data, a command, a reprogramming action, etc.) or the link type (e.g., whether the link is between hosts in the same collision domain or it goes across networks) are examples of *implicit* attributes for the link.

Accordingly, the term semantic attribute may be defined as follows:

A semantic attribute is a feature representing a host, a link or the contextual environment that is obtained by mapping one or several network message fields (possibly over several network messages at different points in time) to a more abstract category. This representation leads to the creation of a high-level semantic model of network activities. For instance, the role of a host and the message type of a link are examples of semantic attributes: they can be obtained by looking at several message field values over several messages and help to reason over network activities with a higher level of abstraction. For instance, rather than saying *IP 1 cannot communicate with IP 2* (low level of abstraction), one can say *the PLC can only communicate with the SCADA* or *the coffee machine cannot communicate outside of the local network.*

A semantic attribute may be described as an attribute which value cannot be extracted from a direct map to a protocol field value. The extracting of an attribute value of a semantic attribute may require to take into account for example the context, previous history, other protocol field values and/or other attributes value, in order to be able to assign a value to the semantic attribute. Examples of semantic attributes include but are not limited to:

- the role of a host (e.g., if it is a PLC, a master, a slave, a coffee machine, a printer, a workstation, a web server, etc.)
- the operating system of a host (e.g., if it is a Linux, a Windows, a proprietary operating system, etc.)
- the type of network of a host (e.g., if the host is in a public or private network, in an enterprise or control or field network, etc.)
- the vendor of a host (e.g., if it is a Cisco, a Siemens, a Dell device etc.)
- the criticality of a host (e.g., if it is a high, medium or low criticality asset)
- the operation of a link (e.g., if it is a Read, a Write, a Notify, a Reprogram, operation, etc.)
- the type of protocol of a link (e.g. if it is an operational, a vendor specific or an open protocol, etc.)
- the message type of a message

Using semantic attributes, a higher level semantic model of network activities may be created, enabling to define the attribute based policies as semantic based policies, thus allowing to define the policies at a higher level of abstraction.

Attributes extraction may be described as to continuously and passively extract **explicit** and **implicit** attributes from the network traffic. Attributes can refer either to network assets, i.e., hosts, or to network communications, i.e., links. Attributes are derived by passively listening to the traffic, without injecting any additional traffic that although could help the analysis, could also disturb the normal operation of the network. The extraction of implicit attributes allows to enrich network events with semantic and contextual information. For instance, rather than having an event saying that *IP*1 speaks to *IP*2, thanks to the attributes extraction we can say we have seen a host of role *Terminal* speaking to a host of role *PLC* with a link containing a *reprogramming* message type. In turns, this opens the door to the writing of attribute-based detection policies.

Attribute-based policies may be described as to discern acceptable from non-acceptable network activities by relying on non-obvious, automatically extracted hosts and links attributes. An example of attribute-based policy is the following: *'Only hosts with role Engineering Workstations (EW) can send reprogram messages to host with Role PLC'*. i) the attributes on which the policies is defined are automatically extracted from the network. This relieves the user from the burden of manually associating a role to each host; ii) using attribute-based rather than IP-based policies allow to write semantic-aware policies that, in a single statement, might encompass several hosts (i.e. all those matching the attributes of the policy); and iii) by using attributes, policies are expressed at a higher level of abstraction which increase their understandability and portability.

An example of the above described process of parsing, attribute extraction, model updating, holding the model(s) against attribute based policies, etc. is provided below.

Network traffic on a data network is monitored.
A host meta-model is provided which comprises the following attributes: {IP, operating system, vendor, role, firmware, type of network}. A link meta-model is provided which comprises the following attributes: {source IP, destination IP, protocol, source port, destination port, operation, number of occurrences}
A network link L1 from host H1 to host H2 is observed. The network traffic is parsed and the following protocol fields are extracted from the network message: {protocol name = Modbus, source IP = 10.1.1.1, destination IP = 10.1.1.2, source port = 502, destination port = 502, function code = 16 }
The host and link models related to respectively H1, H2 and L are selected. Assuming the models retrieved have the following current state:

Model for H1 = {IP = 10.1.1.1, operating system=?, vendor = Siemens, role=?, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for H2 = {IP = 10.1.1.2, operating system=?, vendor = Siemens, role=?, firmware=?, type of network=? }

Model for L is empty since L1= { source IP=?, destination IP=?, protocol=?, source port=?, destination port=?, operation=?, number of occurrences =0} where the symbol '?' means there is no value for the attribute

Attribute values are extracted from the protocol field. For instance:

attributes for H1 = { IP = 10.1.1.1, role=master}. Note that the attribute role is a semantic attribute since it is extracted by (in the present example) relying on heuristics such as, '*if there is a Modbus link with function code equal to 16 from host Hx to host Hy, and the vendor for role Hy is equal to Siemens, then Hx has role equals to master while host Hy has role slave*'

attributes for H2 = { IP = 10.1.1.2, role=slave}. Note that the attribute role is a semantic attribute since it is extracted by (in the present example) relying on heuristics such as, '*if there is a Modbus link with function code equal to 16 from host Hx to host Hy, and the vendor for role Hy is equal to Siemens, then Hx has role equals to master while host Hy has role slave*'

attributes for L1 = { source IP=10.1.1.1, destination IP=10.1.1.2, protocol=Modbus, source port=502, destination port=502, operation= change setpoint, number of occurrences =1}. Note that the attribute operation is a semantic attribute since it is extracted by (in the present example) relying on heuristics such as '*If there is a Link L where protocol name is equal to Modbus and function code is equal to 16 then the value for the semantic attribute operation is equal to 'change setpoint''*

The selected models for H1, H2 and L are updated with the new attribute values. Hence:

Model for H1 = {IP = 10.1.1.1, operating system=?, vendor = Siemens, **role=master**, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for H2 = {IP = 10.1.1.2, operating system=?, vendor = Siemens, **role=slave**, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for L1= { source IP=10.1.1.1, destination IP=10.1.1.2, protocol=Modbus, source port=502, destination port=502, **operation=change setpoint**, number of occurrences =1} where the symbol '?' means there is no value for the attribute

It is assessed if the updated models comply with existing attribute-based policies. For instance, assuming that the attribute-based policy set consist of one policy saying that 'If the source host *vendor* is equal to *Siemens* and its *role* is equal to *master* and the destination host *vendor* is equal to *Siemens*, then the source host can send '*change setpoint*' operations to the host destination'. In this case the updated models comply with the policy hence no alert is generated.

Another example is provided below.

Network traffic on a data network is monitored.

A host meta-model comprises the following attributes: {IP, operating system, vendor, role, firmware, type of network}.

A link meta-model comprises the following attributes: {source IP, destination IP, protocol, source port, destination port, operation, number of occurrences}

A network link L2 from host H3 to host H4 is observed.

The network traffic is parsed.

The following protocol fields are extracted from the network message: {protocol name = Modbus, source IP = 10.1.1.1, destination IP = 10.1.1.2, source port = 502, destination port = 502, function code = 16 }

The host and link models related to respectively H1, H2 and L are selected. It is assumed the models retrieved have the following current state:

Model for H2 = {IP = 10.1.1.1, operating system=?, **vendor = Dell**, role=?, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for H3 = {IP = 10.1.1.2, operating system=?, **vendor = Dell**, role=?, firmware=?, type of network=? }

Model for L2 is empty since L= { source IP=?, destination IP=?, protocol=?, source port=?, destination port=?, operation=?, number of occurrences =0} where the symbol '?' means there is no value for the attribute

Attribute values are extracted from the protocol field. For instance:

Attributes for H3 = { IP = 10.1.1.1, role=?}. Note that the attribute role in this case is not assigned **since the heuristic**, '*if there is a Modbus link with function code equal to 16 from host Hx to host Hy, and the vendor for role Hy is equal to Siemens, then Hx has role equals to master while host Hy has role slave*' **is not met**

Attributes for H4 = { IP = 10.1.1.2, role=?}. Note that the attribute role in this case is not assigned **since the heuristic**, '*if there is a Modbus link with function code equal to 16 from host Hx to host Hy, and the vendor for role Hy is equal to Siemens, then Hx has role equals to master while host Hy has role slave*' **is not met**

Attributes for L2 = { source IP=10.1.1.1, destination IP=10.1.1.2, protocol=Modbus, source port=502, destination port=502, operation= change setpoint, number of occurrences =1}. Note that the attribute operation is a semantic attribute since it is extracted by (in the present example) relying on heuristics such as '*If there is a Link L*

where protocol name is equal to Modbus and function code is equal to 16 then the value for the semantic attribute operation is equal to 'change setpoint''

The selected models are updated for H3, H4 and L2 with the new attribute values. Hence:

Model for H3 = {IP = 10.1.1.1, operating system=?, **vendor = Dell**, role=?, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for H4 = {IP = 10.1.1.2, operating system=?, **vendor = Dell**, role=?, firmware=?, type of network=? } where the symbol '?' means there is no value for the attribute

Model for L2 = { source IP=10.1.1.1, destination IP=10.1.1.2, protocol=Modbus, source port=502, destination port=502, **operation=change setpoint**, number of occurrences =1} where the symbol '?' means there is no value for the attribute

It is assessed if the updated models comply with existing attribute-based policies. For instance, assuming the attribute-based policy set consist of one policy saying that 'If the source host *vendor* is equal to *Siemens* and the destination host *vendor* is equal to *Siemens*, then the source host can send '*change setpoint*' operations to the host destination'. In this case the updated models **does not** comply with the policy hence **an alert is generated**.

In an embodiment, at least one semantic attribute value is derived from a combination of protocol field values obtained from at least two protocol messages transmitted over the data communication network at different points in time. Hence, the semantic attribute may be derived by observing a behavior and/or by combining data as transmitted at different points in time, thus to be able to derive attributes that cannot be derived from a single protocol field as such. For example, attributes relating to a behavior of a second host may be derived form a protocol message sent from the first host to the second host in combination with the response as sent from the second host to the first host. The semantic attributes derived at different points in time may hence allow to derive more attribute related information and/or to derive higher level abstractions from the information retrieved from the protocol data fields.

In an embodiment, the set of models comprises a model for the first host, a model for the second host and a model for the link, wherein each of the models comprises at least one semantic attribute. The models of a host respectively a link may be built using attribute values of semantic attributes of the host respectively the link. Accordingly, the models may be built taking semantic attribute values of semantic attributes into account. Using the semantic attributes, the models may be more informative, more generic, and the models may enable the creation of higher level of abstraction policies.

In an embodiment, the policies each define an outcome in case a condition is met, the condition being defined in terms of a respective at least one of the attributes having a defined attribute value, the outcome of the attribute based policies indicating if the selected model is allowable or not allowable. By writing down the policies in the form of a condition and an outcome in case the condition is met, the policies may conveniently be programmed and be amended when learning, combining, updating policies as will be described in further detail below. The outcome of the policy may for example comprise "allowable"' or "not allowable" or "quarantine" or "log", or "quarantine, log".

In an embodiment, the condition of each policy comprises at least one semantic attribute value. Hence, each policy may have a higher level of abstraction, may be more easily understood/interpreted by a user, and may be more generic.

In an embodiment, b) comprises applying rules to the protocol field values, the rules assigning attribute values to attributes based on the protocol field values. Rules may allow to automatically assign attribute values on the basis of extracted protocol field values. An example of such a rule could be that hosts using the operating system "Linux" and having incoming messages of protocol IMAP are to be assigned the attribute of having the role of an email server. The rules may be domain specific and may be provided based on domain specific knowledge.

In an embodiment, b) comprises direct mapping protocol fields on attribute values. Attributes may be derived by direct observation of network data traffic, Attributes such as IP address may be directly derived from the protocol field values.

In an embodiment, b) comprises applying a heuristic to the data traffic and deriving the semantic attribute value using the heuristic.
In another embodiment, b) comprises applying a classifier to the data traffic and deriving the semantic attribute value using the classifier.
The method may further comprise determining a level of confidence of the classifier and wherein the attribute value is only derived from the classifier when the level of confidence is above a predetermined confidence level. The attribute value obtained using the heuristic may in an embodiment have priority over the attribute value obtained using the classifier.

Each semantic attribute can be derived by applying either heuristics or classifiers. Typically, the results obtained by heuristics have priority over classifiers. An example of heuristic used to infer the attribute Role is as follows: *"If a conversation using the DNS protocol is observed, then the target host of such link has role DNS server"*. Clearly, several such heuristics can be put in place. In addition, heuristics can be global or local to a specific deployment (e.g., heuristics that are true only in a given sector). On the other hand, a classifier is a model that given a set of features as input is able to associate a value (or class) to an attribute as output. Classifiers are typically generated by applying machine learning algorithms to labeled datasets, namely datasets where the association between features and class is known. Later, these classifiers can be used to infer the class on un-labeled data. Generally, classifiers associate a confidence level to their guessing: to keep our attribute extraction component highly reliable, we only resolve the attribute if the class guessed by the classifier has a high level of confidence (e.g., more than 90%). Note that, it is possible that certain implicit attributes can stay unresolved for a certain time, e.g., until enough information to assess their value is available.

In an embodiment, no stimulus is injected into the data communication network. By passively listening to the data communication taking place on the network, no disturbance of the data communication on the network takes place. Thus, on the one hand no additional load is placed in the network, while on the other hand a better insight may be gained into the behaviour of the data communication network. As no stimuli are injected, it may take some time to gather desired information. Some messages may only be sent sporadically. Hence, the data traffic may be monitored over a relatively long period of time, e.g. hours, days, weeks, to extract the protocol field values that may enable to derive the attribute values as described.

In an embodiment, steps b), c), d) and e) are performed for the first host, for the second host and for the link, the set of models comprising a model relating to the first host, a model relating to the second host and a model relating to the link, the attribute based policies of the set of attribute based policies defining conditions in terms of the attributes of the first host, the attributes of the second host and the attributes of the link.

In an embodiment, the set of attribute based policies comprises whitelist policies, the outcome of the whitelist policies indicating if the selected model is allowable. Hence, the whitelist policies enable, in case of changes in the network configuration, such as updates, new devices, etc., if allowable behaviour is observed, as the attribute based models and attribute based policies

may allow to define allowable behaviour at a high level of abstraction. Hence, legitimate changes, such as upgrades or legitimate new devices, may, using the attribute based policies, be found to show behaviour similar to the behaviour monitored for existing devices, respectively before the upgrade, thus may be found to exhibit similar or the same attributes, thus to comply with the whitelist policies. Hence, false alarms as a result of regular updates, upgrades, etc, may to a large extent be avoided.

In another embodiment, the set of attribute based policies comprises blacklist policies, the outcome of the blacklist policies indicating if the selected model is not allowable. Similarly to above, once a particular behaviour of a host has been defined as malicious, similar attacks or similar attempted attacks may provide that similar or the same attributes may be derived there from, allowing to detect possibly malicious activities, even if the activities do not literally correspond to the activities corresponding to the blacklist.

In case the model relating to the first host, the second host or the link cannot be matched to any of the attribute based policies, the data communication relating to the respective one of the first host, the second host and the link is stored in a quarantine.

The goal of the Attribute extraction component is to find as much information as possible about the monitored network, a single host, a communication link, etc., by continuously and passively monitoring the network traffic.

Attribute extraction can happen in different ways:

1. via an expert's direct assignment: attributes are user-determined, through direct assignment (e.g., *"host X has vendor equal to Honeywell and role equal to PLC version Y"*) or through the use of rules *"all PLC at location Y have value K for attribute Z"*).
2. via direct network mapping: attributes can be derived by direct observation of network traffic (e.g., the IP address can be extracted from a specific network message field).
3. via heuristics and classifiers: there are rules mapping observables to attributes; where observables may be unsolicited (passive sniffing) or solicited (active discovery). Rules can be either created by an expert (heuristics) or extracted from the data with machine learning or data mining algorithms (classifiers).

Attributes can refer either to host or to communication links, and they can be either *explicit* or *implicit*. An *explicit* attribute can be directly observed, e.g., from network packets fields. For example, the MAC and the IP addresses are *explicit* host attributes since they are contained in specific fields of layer 2 datagrams and layer 3 packets. Likewise, the protocol, the port, or the

source and target are examples of *explicit* attributes for a link. To extract explicit attributes, one can apply in-depth protocol inspection techniques. On the other hand, an attribute is *implicit* if it can only be derived by performing an analysis that could possibly involves multiple messages or multiple communications. Implicit attributes can be referred to as **semantic attributes**, since they allow the semantic-enrichment of the information we have about hosts and/or links. For instance, a host's role, its Operating System (OS) or its Criticality are example of *implicit* attributes. In fact, to derive the role of a host one should analyze multiple network activities, understand what layer 7 protocols it speaks, whether it only starts or also serves new connections, which ports are used involving that host, etc. In the same way, attributes such as the message type (e.g., whether the link represents a request for data, a command, a reprogramming action, etc.) or the link type (e.g., whether the link is between hosts in the same collision domain or it goes across networks) are examples of *implicit* attributes for the link.

The present invention may adopt a passive approach that extract attributes by passively sniffing network traffic. Using active probes has the main disadvantage of generating additional traffic that in environments with limited resources and strong constraints such as industrial networks might not be acceptable. In addition, by using a passive approach, the attribute extraction might not be so straightforward: the network events necessary to match a heuristics might never show up, hence the present invention may take countermeasure to address this inconvenience (e.g., by using machine learning techniques as described elsewhere in the present document).

The present invention may make use of the role (and the other attributes) to characterize the normal behavior of a host, and detects deviations from normality as possible infection (attacks). This means the present invention does not need to define malicious roles, since the violations of whitelist policies will automatically detect infections.

The present solution may infer implicit attributes based on characteristics of the traffic by adopting a passive rather than an active approach. Using active probes generates additional traffic that might not be acceptable in certain environments;
The present solution may apply an approach (heuristics and classifier) not only to infer role but to extract several implicit attributes (e.g., role, vendor, operating system, location, functionalities, and message type).

The present solution uses the role (and the other attributes) to characterize the acceptable behavior, then we detect deviations from normality as possible infection (attacks). This means the present solution does not need to define malicious roles, since the violations of whitelist policies will automatically detect infections.

The attribute-based policy detection aims to enforce policies based on explicit and implicit hosts and links attributes.

Attribute-based policies are used to express domain-specific security constraints. For instance, in industrial control system, typically only Engineering Workstations can change the logic of Programmable Logic Controller (PLC), while SCADA or DCS only issue commands to read or write PLC memory. This domain knowledge can be translated into attribute-based policies and an intrusion detection system can raise an alert in case a policy is violated. Example of attribute-based policies are as follows:

- Only hosts with role Engineering Workstation are allowed to send a reprogram command to host with role PLC.
- A reprogram command cannot be send during the night.
- A reprogram command cannot be send over a cross-network link.

As can be seen, to express these policies, both host and link attributes are necessary. In addition, it is desirable to define a language that is simple to understand, highly expressive, able to identify conflicting policies and scalable with the increase of number of policies. Also, such language may be needed to be able to account for both positive and negative policies. In fact, positive policies permit the listing of a whitelist of acceptable behavior, while negative policies permit the definition of a blacklist of non-acceptable behavior.

A possible way that can be used to express policies is in form of if condition then action statements. Conditions can be boolean conditions that can evaluate true or false and can be defined on link, hosts, or context attributes (e.g., time, location). Actions can be used to express whether the network activity is acceptable or not acceptable (e.g., allow or discard). In addition, policies can optionally have obligations (what to do in case the policy is matched). Example of obligations include send email to admin user Jeff, set priority to medium, set criticality to high.

Attribute-based policies can be created in two-ways, a) by an expert with domain knowledge and b) automatically from the raw traffic. In the first case, it is intended to provide visual aid to the human expert to craft policies. For instance, the user could pick objects and attribute from a graphical interface rather than writing the policy as a text. In the second case, is aimed at

developing an algorithm that automatically learns policies from network data, leveraging the semantic enrichment provided by the attribute extraction component.

The Attribute-based policy P may be defined as follows:

Given a set of host- (A_H), link- (A_L) and context-related (A_C) attributes $A = A_H \cup A_L \cup A_C =< a_1 = v_{a1},\ a_2 = v_{a2},..,a_n = v_{an} >$ where $a_i \in A$ has value $va_i \in V_i$ with $i \in [1,n]$, we define an attribute-based policy as:

P = < if < ATTRIBUTE OP VALUE [{LOGICOP ATTRIBUTE OP VALUE}]> *then* ACTION [{, OBLIGATION}]>

where:

- ATTRIBUTE can be any host-, link- or context-based attribute $a_i \in A$;
- OP can be any comparison operation (e.g., $>, <, \neq, =$, etc.);
- VALUE is any value $va_i \in V_i$ the attribute a_i can take;
- LOGICOP is any logical operator (e.g.,and, or, not)
- ACTION defines what has to be done in case of a positive match of the policy (e.g., deny or permit)
- OBLIGATION defines additional actions to be taken in case of a match (e.g., send email= name@domain.com, set priority=high, set criticality = medium).

Note that the argument of the *if* clause is an attribute-based boolean expression that refine the applicability of the policy: meaning that only if the evaluation of the expression is true the action will be enforced. Also, since the formalism [...] indicates optional elements and {...} indicates repetition of elements, an attribute- based policy can have one or more attributed-based expressions (i.e., ATTRIBUTE OP VALUE) and zero or more obligation. At least one of the attributes in the if clause may be an implicit (i.e., semantic) attribute.

As such, attributes based policies may be used in Access Control (AC), where access to a resource is granted or not according to the attributes possessed by the requester. In access control, attributes are usually substantiated by certificates and according to how attributes are modeled, different languages have been proposed and adopted. However all the available solutions rely on the manual definition of attributes and no solution that defines attributed-based policies by relying on the automatic extraction of attributes from network data is available.

Thanks to the wide number of host and link attributes that may be taken into account, the policies as described in the present document can be much more fine-grain than traffic-flow based policies. For instance, while with flow-based policies one could only say that two hosts can exchange a certain number of bytes over a certain port, with the presently disclosed attribute-based policies one can express constraints that go much deeper, and that can say what kind of messages can or cannot be exchanged. For instance, one could say that PLC cannot be reprogrammed during the night, or that PLC cannot be reprogrammed over a cross-network link.
in the present invention, host and link attributes may be automatically extracted from the network traffic. This mean that no credentials mechanism needs to be in place to adopt the attribute-based policies.

According to the invention, attributes do not need to be manually set per each component since they can be automatically inferred from raw traffic by using the attribute extraction component. The present invention combines the automatic and passive extraction of attributes from raw traffic with attribute-based policies to create semantic-aware policies to discern acceptable from malicious network activities. A single attribute-based policy can refer to several entities (all those sharing the same attributes). The attribute based policies are easy to write and understand.

In an embodiment, in case the model relating to the first host, the second host or the link cannot be matched to any of the whitelist policies, the data communication relating to the respective one of the first host, the second host and the link is stored in a quarantine. Storing the data traffic that does not comply to a whitelist policy in quarantine, allows to suspend judgement as to allowability of the data traffic. The data traffic as stored in quarantine may be applied to learn new policies and/or to update existing policies as will be described in more detail further below.

In an embodiment, the method further comprises providing a consistency rule, the consistency rule defining consistent combinations of at least two attributes of one of the model relating to the fist host, the model relating to the second host and the model relating to the link, comprising

- verifying, on the basis of the attributes derived from the monitored data traffic, if the monitored data traffic complies to the consistency rule,

- storing the data traffic in a quarantine in case the data traffic does not comply to the consistency rule.

Using the consistency rule, it can be verified if the attributes of a host are consistent with each other.

The consistency rule may accordingly be defined as a rule that defines acceptable attribute values or combination of attribute values in a host/link model. The consistency rules may on the one hand be applied to detect inconsistencies, before even applying existing attribute based policies. Thus, inconsistencies may be detected and the related data traffic may be stored in quarantine in case an inconsistency is detected. Furthermore, when updating the attribute based policies from the data traffic stored in quarantine, as will be described in more detail further below, the consistency rules may be used to avoid to introduce errors in the host/link models during the update process.

In an embodiment, the detecting, on the basis of the attributes derived from the monitored data traffic, if the monitored data traffic complies to the consistency rule, is performed before step e). Thus, data traffic may be handled accordingly, e.g. stored in quarantine, in case it appears that the consistency rules are not complied with.

In an embodiment, the consistency rule includes at least one of a time of occurrence of the data traffic and a location of occurrence of the data traffic. Hence, the consistency rules may take account of time and location thus allowing to refine the consistency rules.

In an embodiment, the consistency rule relating to the first host comprises an attribute of another host, preferably an attribute of the second host. Hence, the consistency rules relating to one host may take into account attributes of another host. For example, if the first host is a PLC and the second host is a PRINTER, then they cannot have the same value for the attribute NETWORK. In other words, PLCs and PRINTERs cannot be placed in the same network

In an embodiment, a group of hosts is defined, wherein the method comprises determining if the host to which the attributes relate, is comprised in the group, and applying the consistency rule in case the host to which the attributes relate, is comprised in the group. Thus, consistency rules may be defined locally, e.g. relating to a group of e.g. similar hosts. The group may be defined using e.g. a clustering algorithm.

In an embodiment, the consistency rules are learned using machine learning, preferably using association rules. The consistency rules may be predefined, e.g. entered by an operator based on experience, security regulations, etc.. Alternatively, the consistency rules may be learned from the data traffic.

Consistency rules can be of various types, such as:

- general rules dictated by experience to e.g., identify combinations of attributes values that are not permitted (e.g., "*a host cannot be at the same time a PLC and an Engineering workstation* ", or "*the role of a host cannot be* terminal *if the vendor is* Honeywell") ;
- local rules to the particular site to e.g., identify combinations of attributes values that are necessary, (e.g., "*all host with role equal to PLC have vendor equal to Honeywell* " implying that if there is a host with attributes role equal to *PLC* and attribute vendor equal to *Rockwell*, this would be in violation of the rule;
- may depend on security or governance policies, or on regulations;
- may change over time, e.g., when the governance policy of a system can apply;
- may depend on parameters of different kind, including time, location, and attributes of other hosts;
- may be either global, i.e., the consistency rule applies to every host in the network, or local, i.e., the consistency rule is only true for a certain neighborhood, namely, a group of similar hosts. Clearly, to define a local rules it is necessary to define a metric for assessing the similarity between hosts and thus identify neighborhood of similar hosts (e.g., by using clustering algorithms).

In many cases, a consistency rule will be a function that maps the attributes of a host into the set *consistent*, *inconsistent* . More generally, a consistency rule is a function that maps all attributes of all hosts and their environment into a set of *acceptable* values. This gives an indication of the *health* or *consistency* state of the system at hand and it may also give an indication on actions that may be taken in case an inconsistency is identified. For instance, it could be useful to have rules mapping attributes into a set in which the members could be *consistent*, *inconsistent*, *suspicious*, *problematic* etc. Consistency rules can be either manually defined by an expert or inferred from the data. In the latter case, machine learning algorithms can be used (e.g., association rules) to distinguish normal from unusual combination of attributes.

Thus, applying consistency rules, the following may be achieved:

- guarantee that the algorithm used to automatically extract attributes, does not put the system in a semantically invalid state.
- apply consistency check to detect when an attribute is 'strange' not on its own but with respects to other attributes of the same host or of the same neighborhood.
- machine learning techniques to distinguish normal from unusual values or combination of attributes values.

The terms quarantine, quarantine and detection algorithm, consistency rule, and inconsistency detection may be described as follows:

quarantine: the quarantine contains hosts and links that are in a limbo: the judgment of whether they are legit or not is suspended until there is not enough evidence to classify them. We represent the quarantine as a list of tuples containing the following elements:

- – an identifier for the tuple ;
- – a target, namely the host or the link that is in quarantine, with the associated host/link model
- – the support to the hypothesis 0 that the host/link in quarantine is legit;
- – the support to the hypothesis 1 that the host/link in quarantine is malicious;
- – the list of *events* composing the evidence used to compute the support. Events can be extracted from data traffic or can be generated by the system or provided by the user. Example of events can include: New Host, New Host Attribute Value, New Link, New Link Attribute Value, Alert, User feedback, etc.
- – the data traffic corresponding to the host/link in quarantine.

detection and quarantine algorithm: the process to detect incident and quarantining hosts and/or links. The process works as follows:

- new data traffic is available,
- parse data traffic to extract host and link attributes
- select the corresponding hosts and/or link model(s)
- if there is no existing host/link model to associate with the data traffic, initialize the model and add the hosts and/or links to the quarantine.
- otherwise, check for attribute inconsistencies
- in case of inconsistency add the correspondent hosts and/or links to the quarantine

- otherwise match the network message with the attribute-based policies
- in case of match with a blacklist policy, raise an alert
- in case of match with a whitelist policy, do nothing
- in case the match cannot be evaluated (e.g., because not all the attributes in the correspondent host/link models are yet featured) then put the host and/or link in the quarantine

consistency rule: a rule that defines acceptable attribute values or combination of attribute values in a host/link model. Consistency rules are used to avoid to introduce errors in the host/link models during the update process.
inconsistency detection: the process of identifying consistency rule violations;

Inconsistency detection to verify that the values of the attributes we use to characterize a host are not in a conflicting state. For instance, a host cannot have at the same time the operating system equal to *Linux* and the Vendor equal to *ABB*. Examples of consistency rules include the following: i) a combination of attributes values that is not possible, e.g., if Role is *Terminal* then Vendor cannot be *Honeywell* ;
ii) a combination of attributes values that is necessary, e.g., if Role is *PLC* then the Vendor has to be *Honeywell* ; iii) an attribute that has only certain values in a certain domain, e.g., Vendor is only *Dell* and *Honeywell* in a certain network; and iv) a combination of attributes values that is usual according to the attributes values of the neighborhood, e.g., all the hosts in a neighborhood only speak certain protocols. The present aspect prevents the algorithm used to extract hosts' attributes to put the model in a semantically invalid state. In addition, in case a malicious host is trying to mimic the behavior of a legit host, consistency check help to detect such misbehavior. This is because the malicious host cannot perfectly mimic another host (e.g., information about the vendor and the operating system are difficult to mimic). Hence, its mimic attempt will likely result in a consistency check violation.

In an embodiment, the method comprises learning the attribute based policy from the data traffic, the learning comprising:

- monitoring the data traffic,
- deriving host attributes and link attributes from the monitored data traffic,

- transform the data traffic into a dataset of attribute-based transactions,
- generate rules by taking into account a frequency of item sets of the host attribute values and link attribute values in the dataset, each one of the rules comprising an antecedent defining a condition and a consequent defining an action,
- determine for each rule a confidence that specifies how often the rule appears to be true, and a support that specifies how often the item set underlying the rule appears in the dataset,
- select rules based on a level of support and a level of confidence,
- translating the rules into the attribute based policy by
    - defining the attribute-based policy condition by joining the antecedent and the consequent of the selected rules, and
    - defining the attribute-based policy *action* based on the level of support and/or the level of confidence.

Policies may be learned e.g. in a learning phase. In the learning phase, (training) data traffic is provided and policies are learned from the (training) data traffic. The (training) data traffic may be formed by regular data traffic as observed during normal operation. Preferably, no stimuli are injected, i.e. the data traffic is gathered by passively listening to the communication taking place on the data communication network. Also, marking by an operator certain events as allowable or not allowable may be omitted as the policies may be learned in an automatized way.

The data traffic is monitored and attributes are extracted from the data traffic in a way as described above. The attributes may be direct and/or indirect (semantic) attributes. Host attributes as well as link attributes are extracted in order to have context available to enable the generation of meaningful policies. The attributes as extracted are stored as a dataset of attribute based transactions, in other words, instead of storing that a certain IP address sends code 0x0A to another IP address, it may be stored that a terminal of the vendor Dell using Windows as operating system sends a read from file message to another terminal of the vendor Dell using Windows as operating system, the link between the terminals using the SMB protocol. Thus, the attributes as derived from the data traffic for transactions, i.e. for network activities, are stored in the form of the dataset. The dataset is applied to generate rules. Thereto, machine learning may be applied to extract rules in the dataset. Association rules are derived from the dataset. The machine learning takes account of a frequency of item sets, i.e. a

frequency of sets of attributes stored in the dataset. Thus, relations between items in the dataset may be discovered making use of the frequency of occurrence of such sets of items, i.e. set of attributes.

Each one of the rules comprises an antecedent defining a condition and a consequent defining an action, i.e. the action associated with the condition. Both the condition and the action are defined in terms of one or more of the attributes. For example, the condition may specify a certain protocol, while the action specifies a port (as it appears, from the data traffic, that the data traffic using the specific protocol usually makes use of that particular port).

For each rule, a confidence and a support is determined. The confidence defines how often the rule appears to be true. The support defines how often an item set, i.e. a set of attributes underlying the rule, occurs in the dataset.

Then, the level of confidence and the level of support are used to select rules. In order for a rule to be translated into a policy, the rule is required to have a certain level of support. For example, only rules with a certain minimum level of support (e.g. a predetermined threshold) are translated into a policy. The condition of the policy is formed from a combination of the antecedent and consequent of the rule. The outcome of the policy (e.g. allow, deny) is formed from the level of confidence. The rules that are often found to be true, i.e. with a high level of confidence may result in an allow action, while the rules that are true only in a low number of cases, i.e. with a low level of confidence may result in a deny action.

Thus, policies may be learned from the data traffic, whereby sets of attributes providing rules that are mostly true may provide attribute based policies defining allowable traffic and sets of attributes providing rules that are mostly not true may provide attribute based policies defining traffic that is to be denied. Accordingly, in an embodiment, the selecting rules based on the level of support and the level of confidence comprises:

selecting, for whitelist policies, rules having the level of confidence above a predetermined positive level of confidence and having the level of support above a predetermined level of support.

Similarly, in an embodiment, the selecting rules based on the level of support and the level of confidence comprises:

selecting, for blacklist policies, rules having the level of confidence below a predetermined negative level of confidence and having the level of support above a predetermined level of support.

The rules may be generated from the attributes using any machine learning and/or data mining technique. In particular, association rules and frequent item extraction may be applied. Accordingly, in an embodiment, the generating rules from the host attributes and the link attributes comprises applying association rules to the host attributes and the link attributes. Furthermore, the generating rules from the host attributes and the link attributes comprises applying frequent items set extraction to the host attributes and the link attributes.

In an embodiment, the translating the rules into the attribute based policy further comprises:

- reducing the number of policies by removing redundant policies, a policy being redundant if its condition includes the whole condition of another policy.
- in case of conflict where two policies share the same condition while the two policies comprise different actions, remove the policy that has less support and confidence.

The number of policies as generated may be reduced, as redundant policies may be removed, assuming that the policies have a same action. In case of different actions, i.e. in case contradicting behaviour may occur, the policy that has less support and/or confidence may be removed..

learning of attribute-based policy may be described as the process of automatically extracting attribute-based policies from data traffic. The process consisting of the following steps:

1. **extraction**: capture data traffic from the network.
2. **pre-processing**: analyze data traffic and transform it into a dataset of attribute-based transactions;
3. **rule mining**: apply rule mining algorithms (e.g., association rules) to the dataset in order to extract association rules. Association rules are generated by taking into account the number of times that certain item set of attributes are observed together into the dataset. The support of a rule specifies how frequently an item set appears in the dataset, while the confidence (expressed in percentage) is an indication of how often the rule has been found to be true.
4. **translation to policy**: the rules generated by machine learning algorithms give an indication of what attributes are more often seen together (antecedent) and what they imply (consequent). This information needs to be translated into attribute-based policies in line with our definitions. For a rule to become a policy, in our method we have to

decide what is the minimum confidence and support and whether a certain level of confidence and support lead to a positive (i.e., allow) or negative (i.e., deny) policy. For instance, let assume positive policies will be generated exclusively for rules with confidence higher than 99% and negative rules will be generated exclusively for rules with confidence lower than 1%.

5. **policy reduction**: some of the policies generated at the previous step can be redundant. A policy Px is redundant if its condition includes the whole condition of another policy Py and the action of the policy is the same. In such a case, Px can be removed.
6. **policy conflict resolution**: a policy Px is in conflict with a policy Py if they share the same condition to which correspond two different actions. To resolve the conflict, one of the two policies need to be removed. To decide which policy to remove, the conflict resolution process will take into account the confidence and the support of the rules that led to the conflicting policies.

Learning of attribute-based policies may be described as to automatically infer new policies by passively observing network traffic. We use learning techniques to automatically extract attribute-based policies. Human intervention –to define policies for distinguishing acceptable from non acceptable behavior – is limited or not required.

It may automatically infer new policies by passively observing network traffic, e.g., by learning the relationships between certain attributes and certain behavior and automatically extract policies from this.

Attributes-based policies can be either defined by a human expert, in which case extensive knowledge about allowed and not allowed behavior is required, or inferred from the data with a learning approach. While the first approach is time consuming and error-prone, the second can be completely automatized and reduces the risks of error (e.g., duplicated or overlapping policies). The present invention proposes an automatic technique to overcome the laborious manual process of defining policies by deriving the policies from the observation of the network traffic flow. Specifically, we mine the whitelist policies from the observation of benign network messages, and the blacklist policies from the observation of network activities under attack (when possible).

ID	Confidence (%)	Antecedent	Consequent	Support (#)
1	99.9	L.*Hsrc*.Role = SCADA & L.*Hdst*.OS = Proprietary	L.MessageType = reprogram	1500
2	100	L.*Hsrc*.Role = SCADA	L.MessageType = reprogram	1600
3	50	L.Proto=SMB	L.DstPort = 139	850
4	0.7	L.Proto=HTTP	L.DstPort = 815	2

Table 1: Association Rules Example

Below, we describe an embodiment of the policy learning process:

1. **extraction**: capture traffic from the network. Network traffic (M) is a sequence of network messages $M = < m_1, m_2, \dots, m_w >$;
2. **pre-processing**: network traffic is analyzed in order to extract host, link and context attributes and it is transformed into a dataset of transactions;
3. **rule mining**: rule mining algorithms (e.g., association rules) can be applied to the dataset in order to extract rules. Table 1 shows a small examples of association rules that can be generated by analyzing the dataset of transactions. Association rules are generated by taking into account the number of times that certain *item set* of attributes are observed together into the dataset. The support of a rule specifies how frequently an item set appears in the dataset, while the confidence (expressed in percentage) is an indication of how often the rule has been found to be true.
4. **translation to policy**: the rules generated by machine learning algorithms give an indication of what attributes are more often seen together (antecedent) and what they imply (consequent). This information need to be translated into attribute-based policies in line with our definitions. For a rule to become a policy, in our method we have to decide what is the minimum *confidence* and *support* and whether a certain level of confidence and support lead to a positive (i.e., allow) or negative (i.e., deny) policy. For instance, let assume positive policies will be generated exclusively for rules with confidence higher than 99% and negative rules will be generated exclusively for rules with confidence lower than 1%. In this case, rule 1 and rule2 in Table 1 will be translated into a positive policy

(i.e., *P*1 and *P*2 below), while rule 4 will be translated into a negative policy (i.e., *P*3). Following, we present the policies obtained as translation of the rules in Table 1:

P1: <if
L:Hsrc:role ==SCADA AND
L:Hdst:os == proprietary AND
L:messageType == reprogram
then allow>

P2: < if
L:Hsrc:role == SCADA AND
L:messageType == reprogram
then allow>

P3: <if
L:Proto == HTTP AND
L:DstPort ==815
then deny >

5. **policy reduction**: some of the policies generated at the previous step can be redundant. A policy *Px* is redundant if its condition includes the whole condition of another policy *Py* and the action of the policy is the same. In such a case, *Px* can be removed. For instance, policy *P*1 in the previous example is redundant as it contains the whole condition of policy *P*2 and they share the same action. Hence, *P*1 can be removed.
6. **policy conflict resolution**: a policy *Px* is in conflict with a policy *Py* if they share the same condition to which correspond two different actions. For instance, in case we have a policy *P*1: <*if L.*Proto == HTTP AND *L.*DstPort == 815 *then* deny> and a policy *P*2:<*if L.*Proto == HTTP AND *L.*DstPort == 815 *then* allow> then *P*1 and *P*2 are in conflict. To resolve the conflict, one of the two policies need to be removed. To decide which policy to remove, the conflict resolution process will take into account the confidence and the support of the rules that led to the conflicting policies.

Accordingly, it is aimed to create policies that take context and semantic into account (e.g., not only the attributes of the link but also those of the source and destination hosts).
since policies are expressed in a policy language that is able to detect and resolve conflict, we insure the system does not fall in an inconsistency status.

association rules and frequent-item extraction algorithm cannot be applied as-is: In the present case some features need to be part of the precedent (or the antecedent). In addition, pre-cooked algorithm might have association rules with a variable number of attributes while we might want this number to be fixed. In addition, the output of association rules and frequent-item set might only serve as an input that needs to be translated into the actual policies.

In an embodiment, the set of attribute based policies comprises the whitelist policies, the outcome of the whitelist policies indicating if the selected model is allowable. In case the model relating to the first host, the second host or the link cannot be matched to any of the whitelist policies, the data communication relating to the respective one of the first host, the second host or link may be listed in a quarantine. Judgement may be suspended and further information may be gathered to decide about the host or link as listed in quarantine. For example, policies may be adapted based on the attribute values as gathered from the data traffic related to the host or link in quarantine, as described in more detail below.

In an embodiment, in case the network message carries information about a host or link for which no model is available the respective host or link is listed in a quarantine.
The quarantine may hence be used to log activities relating to yet unknown hosts or links. The quarantine may be formed by a memory or memory part, where the network messages that relate to the host or link in quarantine may be stored for future analysis.

The method may further comprise: deriving attribute values relating to the host or link listed in the quarantine.
Hence, likewise to a known host or link, protocol field values are extracted from the protocol message and attribute values are derived from the protocol field values. The attribute values may relate to direct attributes as well as to semantic (indirect) attributes. The semantic attributes may be derived in the same or similar way as described above in relation to known hosts or links. Thus, a model for the unknown host or link may be formed based on the attribute values.

The method may further comprise: calculating, from the attribute values relating to the host or link listed in the quarantine, a support to a hypothesis that the host or link listed in the quarantine is legit, and calculating, from the attribute values relating to the host or link listed in the quarantine, a support to a hypothesis that the host or link listed in the quarantine is

malicious. The support to the hypothesis may for example be formed by a probability that the hypothesis is true. The calculations may be performed using the attribute values of the host or link in quarantine, and may for example be performed by comparing a similarity of the attributes of the host or link in quarantine with the attributes of a known host or link.

The calculations may be repeated each time the attributes values of the host or link in quarantine are updated. Thus, the support to the hypotheses may be updated every time new evidence becomes available, Hence, as soon as sufficient support is derived from the attributes values to enable to declare the host or link in quarantine as malicious or legit, following steps may be taken as described below.

Accordingly, the quarantine may comprises, i.e. store, for the hosts and/or links listed in the quarantine:

- an identification of the host or link,
- a list of known attribute values of the host or link,
- a support to a hypothesis that the host or link is legit,
- a support to a hypothesis that the host or link is malicious,
- an identification of data traffic used to determine the support to the hypotheses.

In an embodiment, the method further comprises: checking using the consistency rules, the attribute values of the host or link in quarantine for consistency. The same consistency rules as described above may be applied to the hosts or links in quarantine. Using the consistency rules, potentially malicious activities may be recognized at an early stage, i.e. even before the hypothesis to the fact that the host or link stored in quarantine is malicious, has reached a sufficiently high level. Thus, malicious activities may be recognized at an early stage.

In an embodiment, the method further comprises: assessing, using the attribute based policies, if the attribute values of the host or link in quarantine comply with the set of attribute-based policies. The existing attribute based policies may hence be applied to the attribute values of the host or link in quarantine. The outcome of holding the attribute values of the host or link in quarantine against the attribute based policies may provide for evidence that the host or link in quarantine is legit or malicious. The outcome may be used as evidence in the determination of the support to the hypotheses that the host of link in quarantine is legit respectively malicious. Thus, when assessing the new host or link in quarantine, existing attribute based policies may

be taken into account. In case a blacklist policy is violated, a corresponding alert may be raised.

In an embodiment the method further comprises: deriving attribute based policies from the host attributes and link attributes of the data traffic stored in the quarantine. Once the hypothesis that the host or link stored in quarantine is legit respectively malicious has been found to exceed a threshold, the attribute values of the host or link in quarantine may be used to learn new polices and/or to update existing policies. Thereto, the same or similar learning mechanisms may be applied as described above in relation to the learning phase. Hence, a self-updating intrusion detection may be provided which learns new polices from the data stored in quarantine. Thus, the intrusion detection system may on the one hand avoid unnecessary alarms in that legitimate updates/changes relating to the data network may result in new or updated whitelist policies. On the other hand, malicious activities that have not been noted before, i.e. for which no specific blacklist policy is available yet, may be lead to new or updated blacklist policies. As described above, white the potentially malicious activity is still in quarantine for collection of attributes, evidence and assessment, an alarm may already be raised based on either an inconsistency rule or based on an existing blacklist policy. Thus, during the time that the malicious activity is in quarantine, actions may already be taken (e.g. by a human operator triggered by an alarm signal or by the intrusion detection system autonomously blocking related data traffic) to mitigate a risk of potential malicious actions taking place.

According to an aspect of the invention, frequently occurring behaviour may result in whitelist policies. Thus, when behaviour of a host or link in quarantine occurs frequently, this may, according to an aspect of the invention, be considered as indication that the behaviour is legitimate. Accordingly, a new whitelist policy may be derived from attribute values derived from protocol messages relating to the host or link in quarantine, wherein a frequency of occurrence of the protocol messages relating to the host or link in quarantine exceeds a whitelisting threshold. Hence, the intrusion detection may relatively easy adapt to legitimate changes, upgrades, etc., based on the insight that legitimate changes, upgrades, etc., may result in new behaviour that occurs relatively frequently. A detection threshold may be applied, i.e. an e.g. predetermined whitelisting threshold that expresses a frequency of occurrence above which legitimate behaviour may be assumed. Frequently occurring behaviour may be qualified as whitelist if a similarity with whitelist models or whitelist policies exceeds a similarity threshold.

In particular, the deriving attribute based polices from the host attributes and link attributes of the data traffic stored in the quarantine may comprise:

- (re)compute the support to the hypothesis that the host or link in the quarantine is legit or malicious every time data traffic related to host and/or links in quarantine is observed
- If the support to the hypothesis that the host or link in quarantine is legit is bigger than a whitelist threshold, remove the host or link from the quarantine and use the data traffic related to the host or link to extract new whitelist policies
- If the support to the hypothesis that a host/link in quarantine is malicious is bigger than a blacklist threshold, raise an alert and use the data traffic related to the host or link to extract new blacklist policies
- update the current policies using the extracted new whitelist or blacklist policies.

Once it has been found that the support to the hypothesis that the host or link is legit exceeds a whitelisting threshold, i.e. a threshold indicating that the hypothesis that the host is link is legit exceeds an (e.g. predetermined) threshold level, the host or link may be removed from quarantine and new (whitelist) policies may be learned and./or existing policies updated using the techniques described above.

Similarly, once it has been found that the support to the hypothesis that the host or link is malicious exceeds a blacklisting threshold, i.e. a threshold indicating that the hypothesis that the host is link is malicious exceeds an (e.g. predetermined) threshold level, an alert may be raised and new (blacklist) policies may be learned and./or existing policies updated using the techniques described above.

The (re)computing the support to the hypothesis that the host or link in the quarantine is legit or malicious may comprise:

- computing a similarity of the host or link in quarantine with another host or link.

As explained above, the support to the hypothesis may for example be formed by a probability that the hypothesis is true. The calculations may be performed using the attribute values of the host or link in quarantine, and may for example be performed by comparing a similarity of the attributes of the host or link in quarantine with the attributes of a known host or link. Furthermore, the attributes values of the host or link in quarantine may be held against the existing polices, and the outcome thereof may be used as evidence in the computation of the support to the hypotheses. For instance, if the host or link in quarantine has most (e.g. higher

than 90%) of attribute values equal to a known host or link, then we increase the support to the legit hypothesis. Similarly, if the host or link has attribute values rarely or never seen (e.g., less than 1% of known host or link models have such values) then we increase the support to the malicious hypothesis.

As mentioned, the using the data traffic related to the host or link to extract new whitelist policies respectively the using the data traffic related to the host or link to extract new blacklist policies may comprise learning the whitelist policies respectively the blacklist policies using the above described learning techniques.

Hence, the learning of the attribute based policies may not only be performed at an initial stage, e.g. in a learning phase, but may be continued during operation, so as to provide that the intrusion detection system keeps learning to adapt to legitimate changes, new malicious activities etc.

The process of self-updating of attribute-based policy may be described as follows:
the process to maintain and analyze the information contained in the quarantine in order to extract new black/white list attribute-based policies from new emerging behavior. The algorithm works in the following way:

1. Let qi be a host or a link in the quarantine.
2. Anytime new data traffic related to qi is observed:
   - add the data traffic to the evidence related to qi
   - (re)compute the support to the hypothesis that the quarantine item qi is legit (hypothesis 0)
   - (re)compute the support to the hypothesis that the quarantine item qi is malicious (hypothesis 1)
3. For each item in quarantine, the user can provide feedback that will directly impact the related value of hypothesis 0 or hypothesis 1.
4. When a trigger is fired (e.g., a certain period has passed or the user has provided feedback), for each item q in quarantine and for a given threshold τ :
   (a) if hypothesis 0 > τ , remove q from quarantine and add the network messages to the set of legit messages;
   (b) if hypothesis 1> τ , remove q from quarantine and add the network messages to the set of malicious messages;

5. Give the set of legit messages as input to the attribute-based policy learning algorithm to learn new whitelist policies;
6. Give the set of malicious messages as input to the attribute-based policy learning algorithm to learn new blacklist policies;
7. Update the current whitelist and blacklist policies.

Self-updating policies may provide to keep policies and consistency rules up-to-date. This is obtained by suspending the judgment of network events that do not explicitly violate existing policies or consistency rules until 'enough' information is available. In addition, the information collected during the '*suspended judgment*', is continuously analyzed in order to verify whether, considered as a whole, such information does suggest the adherence to an acceptable or a not-acceptable pattern. Finally, when the time comes, the information collected is also aggregated and semantically interpreted for the presentation to the end user. In this way, rather than presenting the user with a detailed lists of events (e.g., '*New host 192.168.1.5*', '*New communication of host 192.168.1.5 with host 192.168.1.4*', '*New communication with protocol Modbus*',
...), we can reason about the events and present a single meaningful message (e.g., '*New host that looks like a legit RTU has been identified*'). The system allows the reduction of false positive caused by legitimate changes in the network behavior. This is achieved by automatically recognizing new emerging legitimate behavior and by automatically embedding it in new policies derived from the new information collected. By suspending judgment, collecting additional evidence to gain more support before making a decision and using semantic enrichment, we go beyond the general approach to anomaly detection. In fact, generally a single anomaly is considered bad and many anomalies are considered worse. On the contrary, in our case, many anomalies might be a good thing, suggesting that what we are observing is not bad, but just the emerging of new legit behavior.

The inventors aim at addressing the problem that static, 'defined once, valid forever' security policies are not able to deal with changes in dynamic environments. In addition, policies tend to become suboptimal with time, as the operational requirements change. A goal of this component is to keep policies and consistency rules in line with the current state of the system. This means that if new behavior that does not explicitly violates existing policies and constraints is observed and it is considered acceptable, then it needs to be embedded in the policy system.

Policies can be either defined by a human operator or generated by a learning algorithm. Both approaches suffer from the problem of producing incomplete policies set. In case of a human operator, incomplete policies are due to the fact that is difficult to list all acceptable or non-acceptable behavior (e.g., 0-days attacks). In case of a learning algorithm, incomplete policies are due to the fact that, during learning, it is possible that part of the acceptable activities does not emerge (e.g., not all the operation scenarios are executed) or that the system has gone under changes over time. This can eventually cause false positives, namely the misclassification of legit behavior as malicious.

Generally, every system, including computer networks, evolves over time and such change is known as concept drift. In some systems changes can occur faster (e.g., in IT networks) than in other (e.g., in OT networks). In an OT network, examples of concept drift include updates to the hardware (e.g., an old PLC is replaced) or to the underlining physical process. Most of the available solutions for network monitoring, independently from the model they use to perform detection, typically do not update such model or, in case they do, they require human intervention to define the new legit emerging behavior. However, to maintain the optimality of a detection model, it should be updated continuously. To this end, existing machine learning algorithms need to be revised in order to account for continuous update, change or pruning and, especially in network flows, for possible infinite stream of data. Our goal is to automatically identify new legit emerging behavior and automatically update existing policies and consistency rules. A first step towards this goal is the detection of malicious activities and the quarantining of unknown emerging behavior, as described below.

A quarantine Q is a list of records q in the form $q: <ID, Target, Hyp0, Hyp1, E, M>$ where:

ID is the identifier of the record q;

$Target \in \{Host, Link\}$ is the host H or the link L that is in quarantine.

$Hyp0 \in \{0, 1\}$ is the support (e.g., a probability) to the hypothesis that the host/link in quarantine is legit;

$Hyp1 \in \{0, 1\}$ is the support (e.g., a probability) to the hypothesis that the host/link in quarantine is malicious;

$E: <e1, e2, ..., en>$ is the list of events composing the evidence used to compute the support. Example of events can include: New Host; New Host Attribute Value (role =

PLC); New Link; New Link Attribute Value (L.MessageType=reprogram); Alert; User feedback, ...

M :<*m*1, *m*2, ..., *mn*> is the set of network messages related to the host/link in quarantine.

The detection and quarantining algorithm works as follows. Whenever a new network message is available, host and link attributes are extracted and if the network messages carries information about unknown hosts or links, they are added to the quarantine. Then, the algorithm proceeds to check for attribute inconsistencies and to match the network message with the attribute-based policies. In case a blacklist policy is matched, that means the network activity is related to a well-known attack, hence a 'blacklist alert' is raised. Otherwise, the activity is matched against the whitelist policies. In this case, the result of the matching can be threefold: 1) the activity matches a policy in the whitelist, hence we can go back to observe new network activities; 2) the activity does not match any policy, hence an alert is raised; or 3) the activity cannot be matched against the policies, e.g., because not all the attributes necessary to evaluate the policy are available. In the last two cases, the activity is new to the system, hence it is necessary to establish whether it is legitimate or not. To make such decision, a single network activity might not be sufficient, hence we adopt a quarantine approach that logs activities until enough information for emerging behavior is collected. The quarantine contains hosts and links that are in a limbo: the judgment of whether they are legit or not is suspended until there is not enough evidence to classify them.

This approach relies on the intuition that (too often) policies evaluation is made with a pessimistic approach, namely alerts are raised even when that is not necessary (i.e., not everything new is dangerous). Only in case evidence suggests the emerging of new (il)legitimate activities, it is used to infer new policies.

The detection and quarantining algorithm may work as follows.

1. Let P be the set of attribute-based policies where $p_i \in P$ is referred to as blacklist if p_i.*action* == deny, while $p_j \in P$ is referred to as whitelist if p_j .*action*== permit;
2. Let $Q = < q_1, q_2, \ldots, q_n, >$ be the quarantine with $q_1 \in Q$ is in the form $q_i = < ID_i, Target_i, Hyp0_1, Hyp1_i, E_i, M_i >$
3. For every new network message *m*:

(a) extract the source host *Hsrc*, the destination host *Hdst* and the link *L* from *m*;

i. if *Hsrc* (or *Hdst* or *L*) is not in the database of known hosts/links, add it to the quarantine, namely create a quarantine item $q_{n+1} = < (n+1), Hsrc, 0,0, \emptyset, m >$ and add q_{n+1} to *Q* (do the same for *Hdst* or *L*);

(b) extract explicit and implicit attributes for *Hsrc*, *Hdst* and *L*;

(c) if extracted attributes are not compliant with consistency rules, then create an alert about a 'consistency violation';

i. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

ii. otherwise, add the alert to the list of alerts;

(d) else, update *Hsrc*, *Hdst* and *L* with the new attributes values;

i. if *Hsrc*, *Hdst* or *L* are in quarantine, update the respective quarantine items by adding the new attribute values as evidence;

(e) match *m* against each attribute-based policy $p_i \in P$;

i. if *m* matches a **blacklist** policy, create an alert about a 'blacklist violation';

A. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

B. otherwise, add the alert to the list of alerts;

ii. else if *m* matches a **whitelist** policy, then go to analyze the next network message;

iii. else if *m* **does not match any whitelist policy**, then create an alert about a 'not white-listed violation';

A. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

B. otherwise, add the alert to the list of alerts;

iv. else if *m* **match is undefined** (namely no policy is matched e.g., because some of the attributes are not defined) then update the respective quarantine items by adding the candidate policies (e.g., policies that could in future be matched) as evidence;

Human operators can view the quarantine and eventually provide feedback or add missing information. This could help to collect more evidence about the (il)legitimacy of certain hosts/links. When the right time arrives, the elements in the quarantine can be analyzed by our self-updating algorithm to: extract new black- or white-list policies. Events that trigger the

analysis of the quarantine include but are not limited to: i) a user's request, ii) the passing of a time period; iii) the fact that too many entries have been recently added to the quarantine; or iv) the observation of a specific event that, together with other events in the quarantine, permit to confirm or reject a hypothesis (e.g., the recognition of an acceptable or a not-acceptable pattern). For instance, let assume the system identifies a new host on the network. The new host might either be a new legit device added to the network (hypothesis *Hyp*0) or an intruder (hypothesis *Hyp*1). With a conservative detection approach, in this case one would immediately raise an alert. On the contrary, with our self-learning approach, we decide to quarantine the host and to wait for additional information before making any decision. For instance, if we observe activities that show the new host is similar to existing profiles and it does not violate any consistency rule nor blacklist policies, then the support to *Hyp*0 increases. On the other hand, if activities that show the host does not match any profile (e.g., it seems to belong to a certain role but it acts not in line with such a role) then the support to *Hyp*1 increases. One could then define a threshold for the support necessary to make a decision and to notify the end-user with a semantically valid message. Clearly, the system has to be tuned in such a way that, in case the pessimistic hypothesis *Hyp*1 is the correct one, the malicious host cannot stay in quarantine for long enough to make any damage. To this end, we rely on the fact that blacklist policies and consistency rules would immediately spot a condition of danger and raise an alert.

The self-updating algorithm may be defined as follows. The self-updating algorithm maintains and analyze the information contained in the quarantine in order to extract new black/white list policies from new emerging behavior. The algorithm works in the following way:

1. Let $Q = < q_1, q_2, \ldots, q_n, >$ be the quarantine with $q_1 \in Q$ is in the form $q_i = < ID_i, Target_i, Hyp0_1, Hyp1_i, E_i, M_i >$
2. Anytime new evidence *ei* is added to the evidence *Ei* of a quarantine item *qi*:
   (a) (re)compute the support to the hypothesis that the quarantine item *qi* is legit (*Hyp*0*i*) or malicious (*Hyp*1*i*) by accounting for the new evidence *ei*. In a possible embodiment, the support can be computed based on the evidence type. For instance, in case *ei*==New Host, we can compute the similarity of the new host with known host and increase *Hyp*0*i* if the similarity is high while increasing *Hyp*1*i* if the similarity is low. Similar approaches can be taken in case of New Attribute Value or New Link. **In another embodiment, techniques such as First Order Logic,**

Bayesian Inference or Dempster-Shafer Evidence Theory can be used to compute the support.

2. For each item $q \in Q$, the user can provide feedback that will directly impact the related value of *Hyp*0 and/or *Hyp*1.
3. When a trigger is fired (e.g., a certain period has passed or the user has provided feedback), for each item $q \in Q$ and for a given threshold τ:
   (a) if $q.Hyp0 > \tau$ then remove q from Q and add the network messages $q.M$ to the set of legit messages *Mlegit*;
   (b) if $q.Hyp1 \longrightarrow \tau$ then remove q from Q and add the network messages $q.M$ to the set of malicious messages *Mmalicious*;
4. Give *Mlegit* as input to the learning algorithm to learn new whitelist policies;
5. Give *Mmalicious* as input to the learning algorithm to learn new blacklist policies;
6. Update the current whitelist and blacklist policies.

This approach amongst others differs from existing solutions in the sense that while, typically, more anomalies means more 'alerts', hence more 'risk', in our case more anomalies can actually lead to less alerts.
this approach has the advantage of coping with changes in dynamic operational environments by continuously updating and refining policies.
adopt a quarantine algorithm that suspend the judgment about a new host attributes or behavior until there is not enough evidence. In case evidence suggest the emerging of new legitimate activities, it is used to derive new policies.
the system allows to automatically capture changes in the monitored network behavior with reduced false alerts.
changes are automatically embedded in new policies derived from the new information collected.

According to another aspect of the invention, there is provided an intrusion detection system configured to perform the method according to the invention. The intrusion detection system may comprise a data processing device such as a microprocessor, a memory in which program instructions are stored to make the microprocessor perform the method, a network connection for connecting to the data network in order to observe the data traffic. The memory may further store models, policies, alarm messages, etc. Optionally, a user interface may be provided, e.g. for alerting a user, etc.

The invention further provides a data communication network comprising the intrusion detection system according to the invention.

Still further, the invention provides an apparatus comprising the intrusion detection system according to the invention. Applications may for example include a host monitoring system, a host characterization and classification, a standard compliance checking, a policy compliance checking.

Further features, effects and advantages will follow from the appended drawing, illustrating a non-limiting embodiment, wherein:

Fig. 1 illustrates an operating principle of a prior art intrusion detection system;

Fig. 2 illustrates an operating principle of an intrusion detection system according to an embodiment of the invention;

Fig. 3 illustrates a possible attribute extraction according to an embodiment of the invention

Fig. 4 illustrates an overview of an intrusion detection according to an embodiment of the invention;

Fig. 5 illustrates examples of host and link attributes according to an embodiment of the invention;

Fig. 6 illustrates a model for a link attribute according to an embodiment of the invention;

Fig. 7 illustrates an example of inconsistency according to an embodiment of the invention;

Fig. 8 illustrates an example of a network traffic dataset according to an embodiment of the invention;

Fig. 9 illustrates a process of policy learning according to an embodiment of the invention; and

Fig. 10 illustrates an example of a data structure as applied for policy updating according to an embodiment of the invention.

Figure 1 depicts a schematic view of a prior art intrusion detection system. The intrusion detection system is learned in a learning phase, and after the learning phase, the intrusion detection system is applied for intrusion detection. Whitelisting and /or blacklisting models may be learned.

All in all, an intuition behind the present invention is that a whitelisting *model* cannot be learned once and remain static ever after anymore. In addition, relying on the operators for the manual update and maintaining of the *model* is unrealistic since operators often lack the time and the

knowledge needed for the task. An idea underlying the present invention is to create a NIDS that automatically updates the *model* as new traffic is observed. A key challenge for the system is the capability of distinguishing what changes are legitimate (and can be added to the model) and what changes are not (and therefore should still generate alerts). The solution as proposed may form a radical change to the general approach to whitelisting NIDS by substituting the two-phases (learn/detect) approach with a continuous process that is able to deal with legitimate changes in the monitored traffic. The approach as proposed may include the following phases, as schematically depicted in Figure 2:

1. **learn**: monitor the network for a short period of time and use the data collected to create initial *models*, *policies* and *constraints* ;
2. **observe**: passively collect data from the network;
3. **identify**: use new available information to improve existing elements characterization or to create an initial characterization of new elements;
4. **reason**: verify if new evidence is in line with existing *models* and *constraints* and if it does suggest the emerging of new (legitimate) patterns;
5. **react**: if a mismatch with models and constraints is found generate an alert while in case there is enough evidence of (legitimate) emerging patterns update the models.

To implement this model in a successful way, semantic enrichment is made use of. With semantic enrichment, a NIDS is not only capable of communicating punctual facts about network activities (e.g., '*IP 192.168.1.4 sends Step7 command 240 to IP 192.168.1.5* ') but also of interpreting such facts to make them more understandable and increase their information gain (e.g., '*A PLC is sending a reprogramming message to another PLC* '). The semantic enrichment reduces the distance between technology and human operators: if operators get more valuable information from the automatic system (e.g., more meaningful alerts) they are also more inclined to provide feedback to tune the models that in turn improve the overall system.

Overview

Fig 4 depicts an overview of the method as described in the present document. An initial set of whitelist and blacklist policies, together with consistency rules, is given as input to the system as soon as the network monitoring starts. Note that the whitelist and blacklist policy set, together with the consistency rule set could be initially empty. For every new network activity

(A), the system extract hosts' and links' attributes (B). It is important to distinguish between hosts' and links' attributes: while hosts' attributes refer to characteristics of a host, links' attributes refers to the communication taking place between hosts. Once attributes are available, a consistency check (by using the available consistency rules) is performed (C). In case the current attributes do not comply with the consistency rules, an alert is stored in the alert list (L), otherwise the process continues. At this point the network action is matched against the blacklist (D): if a match occurs an alert is generated (L), otherwise we can distinguish two cases: if the system is still in learning mode (E) then the network activity is given in input to the learning module to updated the whitelist policies and the consistency rules; on the other hand, in case the learning phase is over, the network activity is matched against the whitelist policies (G). In case a match occurs at this point, it means the network activity has been already seen in the past (known host/behavior), otherwise we are in the situation where a unknown hosts or behavior is emerging. Typical NIDS at this point would raise an alert. On the contrary, we decide to use a quarantine system that logs network activities (I) until there is not enough evidence to judge the activity legitimate or illegitimate (H). Note that, we also log alerts in the quarantine (O) in case the alert is related to hosts or links that are in the quarantine. In case enough information to make a decision is available, the system will use the events in quarantine for self-updating, namely dynamic extraction of new white- or black-list policies (J). The creation of new policies also produces an event (K) that can be added to the quarantine as additional evidence. Finally, the alert list is analyzed in order to aggregate, correlate and interpret (M) its content and to generate semantic-enriched and context-aware alerts addressed to the end-users (N). The end-users can visualize the alerts and the quarantine items and they can also provide feedback to the quarantine such as flag hosts and links as legitimate or malicious.

In the following, we explain in details of the main components that help to implement the system described above, namely Attributes extraction, Inconsistency detection, Attribute-based policies, Learning of attribute- based policies and Self-updating policies.

Attribute extraction

Figure 3 shows a possible embodiment for the extraction of hosts and links attributes by capturing network traffic. The first step is to extract explicit attributes. Once explicit attributes are available, we can proceed to derive the implicit attributes, for which there is no trivial map with message fields. Each implicit attribute can be derived by applying either heuristics or classifiers. Typically, the results obtained by heuristics have priority over classifiers. An

example of heuristic used to infer the attribute Role is as follows: *"If a conversation using the DNS protocol is observed, then the target host of such link has role DNS server"*. Clearly, several such heuristics can be put in place. In addition, heuristics can be global or local to a specific deployment (e.g., heuristics that are true only in a given sector). On the other hand, a classifier is a model that given a set of features as input is able to associate a value (or class) to an attribute as output. Classifiers are typically generated by applying machine learning algorithms to labeled datasets, namely datasets where the association between features and class is known. Later, these classifiers can be used to infer the class on un-labeled data. Generally, classifiers associate a confidence level to their guessing: to keep our attribute extraction component highly reliable, we only resolve the attribute if the class guessed by the classifier has a high level of confidence (e.g., more than 90%). Note that, it is possible that certain implicit attributes can stay unresolved for a certain time, e.g., until enough information to assess their value is available (see the vendor attribute in the first row in Figure 5).

Example of attribute modelling

Figure 6 shows how the *implicit* link attribute message type can be modeled. The attribute message type expresses the semantic meaning that a specific network message can have in a certain domain. For instance, if host A sends a Modbus message to host B with function code 23, it means that host A is *reading* a data value from host B. The *reading* operation, as shown in figure, can be encoded differently for different protocols: e.g, in IEC-104 a read is encoded with the number 45 and in STEP 7 with the number 4. By applying reasoning and semantic labeling, one can enrich the raw traffic coming from the network. This enrichment can then be used to express policies that are generic (not specific to a given protocol) and simpler to understand. For instance, one could express the generic policy *"Hosts with role Terminal do not reprogram a PLC"* and such policy can be monitored without the policy maker knowing how the reprogramming operation is actually encoded at the network level.

Example 1

Let assume we have the following host-related attributes A_H, link-related attributes A_L, context-related attributes A_C where the symbol ? means the attribute value is unknown, and a network message m_1 where:

- $s = < ip = ?, mac = ?, os = ?, role = ? >$
- $A_L = < proto = ?, srcPort = ?, dstPort = ?, messageType = ?, linkType = ? >$

- $A_C = < location = ?, netwrokType = ?, time = ? msgFrequency = ? >$
- $m_1 = < protoId = Modbus, srcIp = 10.1.1.1,\ dstIp = 10.1.1.2,\ srcPort = 22;\ dstPort = 44;\ functionCode = 16 >$ where the fields are in the order: the identifier of the protocol used (e.g., Modbus, HTTP, IOEC104), the IP address and the port for the sender and the destination, and the function code of the message.

Given the message m_1, we can extract the following host and link information:

- $H_1 = < id = 1,\ A_H = < ip = 10.1.1.1, mac = ?, os = ?, role = ? >>$
- $H_2 = < id = 2,\ A_H = < ip = 10.1.1.2, mac = ?, os = ?, role = ? >>$
- $L_1 = < id = 1,\ H_{src} = H_1, H_{dst} = H_2,\ A_L = < proto = Modbus, srcPort = 22, dstPort = 44, messageType = ?, linkType = ? >>$

In this example, we can see that the link-related attribute *L*.proto is an **explicit attribute** since there is a direct mapping between its value (Modbus) and the field *protoId* of *m*1. On the other hand, the host-related attribute *H*.role is an example of **implicit attribute** since to derive its value one should look at multiple messages and multiple message fields. In fact, given a single message, this attribute stays undefined. For instance, to be able to say that the role of a host is *server* or *client*, one has to look at the *functionCode* and the *protoId* fields of different messages and at the number of messages with a certain *functionCode* (e.g., server typically are the destination of a high number of messages on certain ports with certain message codes). Under these conditions, an example of attribute-policy P is as follows:

<if

L.Hsrc.role == Engineering Workstation AND

L.Hdst.role == PLC AND

L.messageType == reprogram AND

C.networkType == industrial

then

allow,

send email = name@domain.com, set priority = *high*>

Inconsistency detection

Figure 7 shows some example of inconsistency detection. The upper part of the figure shows the 'normal' situation of an example network composed of 4 hosts: 2 DCSs and two PLCs. The

table shows examples of consistency rules. Specifically, rule number 1 says that if a host has role equal to *PLC* then its protocols can only be *Modbus*, while rule number 2 says that if a host has role that includes the value *PLC*, then it can only have *Master and RTU* as additional role (recall that a host can have multiple roles). Finally rule number 3 says that if a host has OS equal to *Windows* then its vendor can only be *Dell*. Under these conditions, in case there is evidence that a certain device (e.g., host F in figure) having role equals to *PLC* is behaving as a host with role *DNS server*, there is a violation of consistency rule number 2. This might suggest that a PLC is corrupted and malicious code is generating the anomalous change of role. Likewise, if the protocols associated to a host contain values *DNS and FTP client* while the role for the same host is *PLC*, there is a violation of consistency rule number 1, also suggesting a corruption of the PLC. On the other hand, in case the vendor of a host is detected to be HP, then rule number 3 is violated. Generally, consistency violations might either indicate that there is a legit change in the system (e.g., the management decided to change its hardware provider) or that a host is compromised hence the consistency rule violation is a signal that something is wrong. Understanding which of the two situations is true is a task for the self-updating policies component.

Learning of attribute based policies

Figure 8 shows an example of how legitimate network messages can be structured after the attributes extraction had took place. As we can see, every row represents a network activity and the associated at- tributes (or features). To this data structure, machine learning and data mining techniques can be applied to extract attribute-based policies. For instance, one could apply association rules learning, a rule-based machine learning method for discovering interesting relations between elements in large datasets. Association rules are if condition then action statements (as our policies) that help uncover relationships between seemingly unrelated data.

However, association rules and frequent items set extraction are approaches that can aid our policies generation but do not cover the entire process as shown in Figure 9. In fact, after an association rules algorithm has been applied to network data, the information extracted from the data needs to be translated in policies. In addition, since association rules tend to generate a high number of rules, an algorithm for policy reduction and conflict resolution needs to be put in place as well.

ID	Confidence (%)	Antecedent	Consequent	Support (#)
1	99.9	L.*Hsrc*.Role = SCADA & L.*Hdst*.OS = Proprietary	L.MessageType = reprogram	1500
2	100	L.*Hsrc*.Role = SCADA	L.MessageType = reprogram	1600
3	50	L.Proto=SMB	L.DstPort = 139	850
4	0.7	L.Proto=HTTP	L.DstPort = 815	2

Table 1: Association Rules Example

Quarantine

Figure 10 shows an example of how the quarantine could look like.

The detection and quarantining algorithm works as follows. Whenever a new network message is available, host and link attributes are extracted and if the network messages carries information about unknown hosts or links, they are added to the quarantine. Then, the algorithm proceeds to check for attribute inconsistencies and to match the network message with the attribute-based policies. In case a blacklist policy is matched, that means the network activity is related to a well-known attack, hence a 'blacklist alert' is raised. Otherwise, the activity is matched against the whitelist policies. In this case, the result of the matching can be threefold:

1) the activity matches a policy in the whitelist, hence we can go back to observe new network activities; 2) the activity does not match any policy, hence an alert is raised; or 3) the activity cannot be matched against the policies, e.g., because not all the attributes necessary to evaluate the policy are available. In the last two cases, the activity is new to the system, hence it is necessary to establish whether it is legitimate or not. To make such decision, a single network activity might not be sufficient, hence we adopt a quarantine approach that logs activities until enough information for emerging behavior is collected. The quarantine contains hosts and links that are in a limbo: the judgment of whether they are legit or not is suspended until there is not enough evidence to classify them.

This approach relies on the intuition that (too often) policies evaluation is made with a pessimistic approach, namely alerts are raised even when that is not necessary (i.e., not

everything new is dangerous). Only in case evidence suggests the emerging of new (il)legitimate activities, it is used to infer new policies.

Self-updating policies

Let consider the quarantine example in Figure 10. The first row contains a host (id=1) and the relative evidence to support the fact that it is legit (*Hyp*0) or malicious (*Hyp*1). Specifically, the host was first unknown to the system with no attributes associated yet (first line in column evidence). Additional network activities added evidence about its role equal to *PLC* (line 2 of column evidence) and later also about its vendor being equal to *ABB* (line 3 of column evidence). The quarantine also contains a link (quarantine id 2) going from host with id=2 to host with id=1. The evidence says that the link goes from a host id=2 with role Terminal to the host id=1 with role *PLC*. In addition, a *reprogramming* messageType is observed. Every time new evidence for a host or a link in quarantine is collected, the support to *Hyp*0 and *Hyp*1 is updated. Let assume an attribute-based policy says that hosts with role *Terminal* cannot *reprogram* hosts with role *PLCs*. This means that this policy can only be matched (and hence an alert raised) when evidence that the attribute role for the host id=1 is *PLC* is collected. Hence, the quarantine helps to detect cases where past activities have violated policies currently in places.

In addition, at any time, the user is allowed to provide feedback about the quarantine. For instance, the user could say that host with id=1 is legit. In this case, the self-learning module needs to account for all the activities performed by host with id=1 while in quarantine, as input for the extraction of possibly new white-listing policies.

Possible definitions as may be applied to define various terms applied in the present patent application are provided below.

Definition 1 (Attributes *A*)

A sequence of attributes A is defined as $A{:}< a_1 = v_{a1},\ a_2 = v_{a2},..,\ a_n = v_{an},>$ where $v_{ai} \in V_i$ with $i \in [1,n]$ is the value taken by attribute $a_i$in the co-domain V_i.

Definition 2 (Host *H* and host attributes *AH*)

A network host *H*, is defined as *H* = <*id, AH* > where *id* is an unique identifier for the host, and $A_H{:}< ah_1 = v_{h1},\ ah_2 = v_{h2},..,\ ah_n = v_{hn},>$ is a sequence of host-related attributes. Examples

of host-related attributes include the IP address, the MAC address, the operative system, the role, etc. We write *Hj .ahi* to refer to a specific attribute of a specific host, e.g., *Hj* .role to refer to the attribute role of the host *Hj* .

Definition 3 (Link *L* and link attributes *AL*)

A network link is a direct communication between a source and a destination hosts. We define a link *L=<id, Hsrc, Hdst, AL>* where *id* is an unique identifier for the link, *Hsrc* is the host source of the link, *Hdst* is the host destination of the link and $A_L: < al_1 = v_{l1},\ al_2 = v_{l2}, ..,\ al_m = v_{lm}, >$ is the sequence of link-related attributes. Example of link-related attributes include the protocol, the src port, the dst port, the message type, the link type, etc. We write *Lj .ali* to refer to a specific attribute of a specific link, e.g., *Lj* .proto to indicate the attribute proto of the link *Lj* . We also write *Lj .Hsrc.ahi* to refer to a specific attribute of the source (respectively the destination) host of the link, e.g., *L.Hsrc*.ip to refer to the IP address of the source host of the link (respectively *L.Hdst.ahi* to refer to an attribute of the host destination, e.g., *L.Hdst*.ip).

Definition 4 (Context attributes *AC*)

A context *C* is a set of circumstances surrounding network activities, and it can be represented by a set of attributes related to the network or the system. A context is denoted as $A_C: < ac_1 = v_{c1},\ ac_2 = v_{c2}, ..,\ ac_n = v_{ck}, >$. Examples of context-related attributes include the time, the geographical location, the type of monitored network (e.g., utility, manufacturing, automation), the number of similar messages over a certain period, etc.

Definition 5 (Network message *m*)

A communication protocol is a system of rules that allows two or more hosts to exchange information via the usage of elements with a well-defined structure. The structure of these elements changes greatly from one protocol to another and their terminology changes too: they can be referred to as packets, frames or messages. Independently from the protocol, we use the term **message** to refer to the basic elements of network communications. Given a sequence of fields $< f_1,\ f_2, ... f_z, >$, we define a field value $v_{fj} \in V_j$ with $j \in [1, z]$ as the value taken by field f_j in the co-domain V_j and we define a message *m* as a sequence of field values $m = < f_1 = v_{f1},\ f_2 = v_{f2},\ .., f_z = v_{fz} >$.

Definition 6 (Explicit attributes)

Given a set of host- and link-related attributes $A = A_H \cup A_L =< a_1 = v_{a1},\ a_2 = v_{a2},.., a_n = v_{an} >$ where $a_1 \in A$ has value $v_{a1} \in V_1$ with $i \in [1,n]$ and a sequence of messages $M =< m_1, m_2, \ldots, m_w >$ where $m_j \in M$ is in the form $m_j =< f_{1j} = v_{f1j}, f_{2j} = v_{f2j}, \ldots, f_{zj} = v_{fzj} >$ with $j \in [1,w]$, we say that the attribute a_1 is **explicit** if its value is equal **to a single field value** of any of the messages in *M* , namely $v_{ai} = v_{fkj}$ with *i* index of the attribute, *k* index of the message field, and *j* index of the message. We will refer to an explicit attribute as a_e.

Definition 7 (Implicit (or semantic) attributes)

Given a set of host- and link-related attributes $A = A_H \cup A_L =< a_1 = v_{a1},\ a_2 = v_{a2},.., a_n = v_{an} >$ where $a_1 \in A$ has value $v_{a1} \in V_1$ with $i \in [1,n]$ and a sequence of messages $M =< m_1, m_2, \ldots, m_w >$ where $m_j \in M$ is in the form $m_j =< f_{1j} = v_{f1j}, f_{2j} = v_{f2j}, \ldots, f_{zj} = v_{fzj} >$ with $j \in [1,w]$, we say that: the attribute a_1 is **implicit** or **semantic** if its value **is a function of multiple field values** and context-related attributes A_C over (possibly) multiple messages, namely if it exists a function *g*() such that $g(M') = v_{ai}$ where $M' \subseteq M\ x\ A_C$

We will refer to an implicit attribute as a_i.

Definition 8 (Attribute based policy)

The Attribute-based policy P may be defined as follows:

Given a set of host-, link- and context-related attributes $A = A_H \cup A_L \cup A_C =< a_1 = v_{a1},\ a_2 = v_{a2},.., a_n = v_{an} >$ where $a_1 \in A$ has value $v_{a1} \in V_1$ with $i \in [1,n]$, we define an attribute-based policy as:

P : if < ATTRIBUTE OP VALUE [{LOGICOP ATTRIBUTE OP VALUE}]> *then* ACTION [{, OBLIGATION}]>

where:

- ATTRIBUTE can be any host-, link- or context-based attribute *ai* P *A*;
- OP can be any comparison operation (e.g., P t→, →, ➔, ↕, ✗, P equal✉, etc.);
- VALUE is any value *vai E Vi* the attribute *ai* can take;
- LOGICOP is any logical operator (e.g., t{,), ✣✉)
- ACTION defines what has to be done in case of a positive match of the policy (e.g., deny or permit)
- OBLIGATION defines additional actions to be taken in case of a match (e.g., send email= name@domain.com, set priority=high, set criticality = medium).

Definition 9 (Quarantine)

A quarantine *Q* is a list of records *q* in the form *q*: <*ID, Target, Hyp*0*, Hyp*1*, E, M* > where:

ID is the identifier of the record *q*;

Target ∈ {*Host, Link*} is the host *H* or the link *L* that is in quarantine.

*Hyp*0 ∈ {0, 1} is the support (e.g., a probability) to the hypothesis that the host/link in quarantine is legit;

*Hyp*1 ∈ {0, 1} is the support (e.g., a probability) to the hypothesis that the host/link in quarantine is malicious;

E: <*e*1, *e*2, ..., *en*> is the list of events composing the evidence used to compute the support. Example of events can include: New Host; New Host Attribute Value (role = PLC); New Link; New Link Attribute Value (L.MessageType=reprogram); Alert; User feedback, ...

M :<*m*1, *m*2, ..., *mn*> is the set of network messages related to the host/link in quarantine.

Definition 10 (Detection and quarantining algorithm)

The detection and quarantining algorithm may work as follows.

1. Let P be the set of attribute-based policies where $p_i \in P$ is referred to as blacklist if p_i.*action* == deny, while $p_j \in P$ is referred to as whitelist if p_j .*action*== permit;
2. Let $Q = < q_1, q_2, ..., q_n, >$ be the quarantine with $q_1 \in Q$ is in the form $q_i = < ID_i, Target_i, Hyp0_1, Hyp1_i, E_i, M_i >$
3. For every new network message *m*:

(f) extract the source host *Hsrc*, the destination host *Hdst* and the link *L* from *m*;

i. if *Hsrc* (or *Hdst* or *L*) is not in the database of known hosts/links, add it to the quarantine, namely create a quarantine item $q_{n+1} = < (n+1), Hsrc, 0,0, \emptyset, m >$ and add q_{n+1} to *Q* (do the same for *Hdst* or *L*);

(g) extract explicit and implicit attributes for *Hsrc*, *Hdst* and *L*;

(h) if extracted attributes are not compliant with consistency rules, then create an alert about a 'consistency violation';

i. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

ii. otherwise, add the alert to the list of alerts;

(i) else, update *Hsrc*, *Hdst* and *L* with the new attributes values;

i. if *Hsrc*, *Hdst* or *L* are in quarantine, update the respective quarantine items by adding the new attribute values as evidence;

(j) match *m* against each attribute-based policy $p_i \in P$;

i. if *m* matches a **blacklist** policy, create an alert about a 'blacklist violation';

A. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

B. otherwise, add the alert to the list of alerts;

ii. else if *m* matches a **whitelist** policy, then go to analyze the next network message;

iii. else if *m* **does not matches any whitelist policy**, then create an alert about a 'not white-listed violation';

A. if the alert involves hosts or links that are in quarantine, update the respective quarantine items by adding the alert as evidence;

B. otherwise, add the alert to the list of alerts;

iv. else if *m* **match is undefined** (namely no policy is matched e.g., because some of the attributes are not defined) then update the respective quarantine items by adding the candidate policies (e.g., policies that could in future be matched) as evidence;

Definition 11 (Self-updating algorithm)

The self-updating algorithm maintains and analyze the information contained in the quarantine in order to extract new black/white list policies from new emerging behavior. The algorithm works in the following way:

1. Let $Q = < q_1, q_2, \ldots, q_n, >$ be the quarantine with $q_1 \in Q$ is in the form $q_i = < ID_i, Target_i, Hyp0_1, Hyp1_i, E_i, M_i >$
2. Anytime new evidence *ei* is added to the evidence *Ei* of a quarantine item *qi*:
   (a) (re)compute the support to the hypothesis that the the quarantine item *qi* is legit (*Hyp*0*i*) or malicious (*Hyp*1*i*) by accounting for the new evidence *ei*. In a possible embodiment, the support can be computed based on the evidence type. For instance, in case *ei*==New Host, we can compute the similarity of the new host with known host and increase *Hyp*0*i* if the similarity is high while increasing *Hyp*1*i* if the similarity is low. Similar approaches can be taken in case of New Attribute Value or New Link. **In another embodiment, techniques such as First Order Logic,**

Bayesian Inference or Dempster-Shafer Evidence Theory can be used to compute the support.

3. For each item $q \in Q$, the user can provide feedback that will directly impact the related value of *Hyp*0 and/or *Hyp*1.
4. When a trigger is fired (e.g., a certain period has passed or the user has provided feedback), for each item $q \in Q$ and for a given threshold τ:
    a. if *q.Hyp*0 > τ then remove *q* from *Q* and add the network messages *q.M* to the set of legit messages *Mlegit*;
    b. if *q.Hyp*1 $\rightarrow \tau$ then remove *q* from *Q* and add the network messages *q.M* to the set of malicious messages *Mmalicious*;
5. Give *Mlegit* as input to the learning algorithm to learn new whitelist policies;
6. Give *Mmalicious* as input to the learning algorithm to learn new blacklist policies;
7. Update the current whitelist and blacklist policies.

P33194NL00

CONCLUSIES

1. Werkwijze voor het detecteren van abnormaal gedrag in dataverkeer op een datacommunicatienetwerk, waarbij een eerste host en een tweede host zijn verbonden met het datacommunicatienetwerk, waarbij het dataverkeer op het datacommunicatienetwerk een link verschaft tussen de eerste host en de tweede host, waarbij de werkwijze omvat:

a) het parsen van het dataverkeer voor het extraheren van protocolveldwaarden van een protocolbericht van het dataverkeer;

b) het afleiden, uit de geëxtraheerde protocolveldwaarden, van attribuutwaarden van attributen van één van de eerste host, de tweede host, en de link;

c) het selecteren uit een set met modellen, van een model dat betrekking heeft op de ene van de eerste host, de tweede host, en de link, waarbij de geselecteerde host meerdere attributen omvat voor het beschrijven van ene van de eerste host, de tweede host, en de link, waarbij ten minste één van de attributen een semantisch attribuut is, waarbij het semantische attribuut een semantische betekenis uitdrukt voor de ene van de eerste host, de tweede host, en de link,

d) het updaten van het geselecteerde model met de afgeleide attribuutwaarden, wanneer de afgeleide attribuutwaarden niet zijn opgenomen in het geselecteerde model bij selectie;

e) het beoordelen of het ge-update, geselecteerde model voldoet aan een set attribuut gebaseerde policies, waarbij elke attribuut gebaseerde policy een veiligheidsrandvoorwaarde definieert van het datacommunicatienetwerk gebaseerd op ten minste één van de attributen van de eerste host, de tweede host of de link, en

f) het genereren van een alert signaal in het geval dat de attribuut gebaseerde policies aangeven dat het ge-update geselecteerde model ten minste één van de attributen gebaseerde policies schendt.

2. Werkwijze volgens conclusie 1, waarbij de ten minste ene semantische attribuutwaarde is afgeleid uit een combinatie van protocolveldwaarden die zijn verkregen van ten minste twee protocolberichten die over het datacommunicatienetwerk op verschillende punten in tijd zijn verzonden.

3. Werkwijze volgens conclusie 1 of 2, waarbij de set met modellen een model omvat voor de eerste host, een model voor de tweede host en een model voor de link, waarbij elke van de modellen ten minste één semantisch attribuut omvat.

4. Werkwijze volgens een van de voorgaande conclusies, waarbij de policies elk een uitkomst definiëren in het geval aan een conditie is voldaan, waarbij de conditie is gedefinieerd in termen dat een respectieve ten minste ene van de attributen een gedefinieerde attribuutwaarde heeft, waarbij de uitkomst van de attribuut gebaseerde policy aangeeft of het geselecteerde model toelaatbaar of niet toelaatbaar is.

5. Werkwijze volgens een van de voorgaande conclusies, waarbij de conditie van elke policy ten minste één semantische attribuutwaarde omvat.

6. Werkwijze volgens een van de voorgaande conclusies, waarbij b) het toepassen van regels omvat op de protocolveldwaarden, waarbij de regels attribuutwaarden toekennen aan attributen gebaseerd op de protocolveldwaarden.

7. Werkwijze volgens een van de voorgaande conclusies, waarbij b) het direct meppen van protocolvelden op attribuutwaarden omvat.

8. Werkwijze volgens een van de voorgaande conclusies, waarbij b) het toepassen omvat van een heuristic op het dataverkeer en het afleiden van de semantische attribuutwaarde gebruikmakend van de heuristic.

9. Werkwijze volgens een van de voorgaande conclusies, waarbij b) het toepassen van een classifier op het dataverkeer omvat en het afleiden van de semantische attribuutwaarde gebruikmakend van classifier.

10. Werkwijze volgens conclusie 9, verder omvattende het bepalen van een niveau van vertrouwen van de classifier en waarbij de attribuutwaarde alleen uit de classifier wordt afgeleid wanneer het niveau van vertrouwen boven een tevoren bepaald vertrouwensniveau is.

11. Werkwijze volgens een van conclusies 8-10, waarbij de attribuutwaarde die is verkregen gebruikmakend van heuristic prioriteit heeft boven de attribuutwaarde die is verkregen gebruikmakend van de classifier.

12. Werkwijze volgens een van de voorgaande conclusies, waarbij geen stimulus wordt geïnjecteerd in het datacommunicatienetwerk.

13. Werkwijze volgens een van de voorgaande conclusies, waarbij stappen b), c), d) en e) worden uitgevoerd voor de eerste host, voor de tweede host en voor de link, waarbij de set met modellen een model omvat dat gerelateerd is aan eerste host, een model dat gerelateerd is aan de tweede host, en een model dat gerelateerd is aan de link, waarbij de attribuut gebaseerde policies van de set met attribuut gebaseerde policies condities definiëren in termen van de attributen van de eerste host, de attributen van de tweede host en de attributen van de link.

14. Werkwijze volgens een van de voorgaande conclusies, waarbij de set met attribuut gebaseerde policies witte lijst policies omvat, waarbij de uitkomst van de witte lijst policies aangeeft of het geselecteerde model toelaatbaar is.

15. Werkwijze volgens een van de voorgaande conclusies, waarbij de set met attribuut gebaseerde policies zwarte lijst policies omvat, waarbij de uitkomst van de zwarte lijst policies aangeeft of het geselecteerde model niet toelaatbaar is.

16. Werkwijze volgens een van de voorgaande conclusies, verder omvattende het verschaffen van een consistentieregel, waarbij de consistentieregel consistente combinaties definieert van ten minste twee attributen van één van het model gerelateerd aan de eerste host, het model gerelateerd aan de tweede host en het model gerelateerd aan de link, omvattende

- het verifiëren, op basis van de attributen die zijn afgeleid uit het gemonitorde dataverkeer, of het gemonitorde dataverkeer voldoet aan de consistentieregel,
- het opslaan van de dataverkeer in een quarantaine in het geval het dataverkeer niet aan de consistentieregel voldoet.

17. Werkwijze volgens conclusie 16, waarbij het detecteren, op basis van de attributen die zijn afgeleid uit het gemonitorde dataverkeer, of het gemonitorde dataverkeer voldoet aan de consistentieregel, wordt uitgevoerd voorafgaand aan stap e).

18. Werkwijze volgens conclusie 16 of 17, waarbij de consistentieregel ten minste één omvat van een tijd van optreden van het dataverkeer en een locatie van optreden van het dataverkeer.

19. Werkwijze volgens een van conclusie 16-18 waarbij de consistentieregel die is gerelateerd aan de eerste host een attribuut van een andere host, bij voorkeur een attribuut van de tweede host, omvat.

20. Werkwijze volgens een van conclusies 16-19, waarbij een groep met hosts is gedefinieerd, waarbij de werkwijze het bepalen omvat of de host waaraan de attributen relateren is opgenomen in de groep, en het toepassen van de consistentieregel in het geval de host waaraan de attributen relateren, in de groep is opgenomen.

21. Werkwijze volgens een van conclusies 16-20, waarbij de consistentieregels worden geleerd gebruikmakend van machine-leren, bij voorkeur gebruikmakend van associatieregels.

22. Werkwijze volgens een van de voorgaande conclusies, waarbij de attribuut gebaseerde policy verder ten minste één van een tijd voor het uitvoeren van de actie, en een link via welke de actie wordt uitgevoerd, uitdrukt.

23. Werkwijze volgens een van de voorgaande conclusies, omvattende het leren van de attribuut gebaseerde policy uit het dataverkeer, waarbij het leren omvat:

- het monitoren van het dataverkeer,
- het afleiden van het host-attributen en linkattributen uit het gemonitorde dataverkeer,
- het transformeren van het dataverkeer in een dataset met attribuut gebaseerde transacties,
- het genereren van regels door het rekening houden met een frequentie van itemsets van de host-attribuutwaarden en link-attribuutwaarden in de dataset, waarbij elk van de regels een antecedent omvat die een conditie definieert en een consequent die een actie definieert, een vertrouwen en een supportniveau,
- het bepalen voor elke regel van een vertrouwen dat specificeert hoe vaak de regel waar blijkt te zijn, en een support die specificeert hoe vaak de itemset die onderliggend is aan de regel in de dataset voorkomt,
- het selecteren van regels gebaseerd op een niveau van support en een niveau van vertrouwen,
- het vertalen van de regels in de attribuut gebaseerde policy door

- het definiëren van de attribuut gebaseerde policy-conditie door het samenvoegen van de antecedent en de consequent van de geselecteerde regels, en
- het definiëren van de attribuut gebaseerde policy-actie gebaseerd op het niveau van support en/of het niveau van vertrouwen.

24. Werkwijze volgens conclusie 23, waarbij het selecteren van regels gebaseerd op het niveau van support en het niveau van vertrouwen omvat:
het selecteren, voor witte lijst policies, van regels waarvan het niveau van vertrouwen boven een tevoren bepaald positief niveau van vertrouwen is en met een niveau van support boven een tevoren bepaald niveau van support.

25. Werkwijze volgens conclusie 23 of 24, waarbij het selecteren van regels gebaseerd op het niveau van support en het niveau van vertrouwen omvat:
het selecteren, voor zwarte lijst policies, van regels waarvan het niveau van vertrouwen onder een tevoren bepaald negatief niveau van vertrouwen is en met een niveau van support boven een tevoren bepaald niveau van support.

26. Werkwijze volgens een van conclusies 23-25, waarbij het genereren van regels uit de host-attributen en de linkattributen het toepassing van associatieregels op de host-attributen en de link-attributen omvat.

27. Werkwijze volgens een van conclusies 23-26, waarbij het genereren van regels uit de host-attributen en de link-attributen het toepassen van frequente items-set-extractie op de host-attributen en de link-attributen omvat.

28. Werkwijze volgens een van conclusies 23-27, waarbij het vertalen van de regels in de attributen gebaseerde policy verder omvat:
het reduceren van het aantal policies door het verwijderen van redundante policies, waarbij een policy redundant is wanneer de conditie daarvan de gehele conditie van een andere policy omvat,
in het geval van conflict waarbij twee policies dezelfde conditie delen terwijl de twee policies verschillende acties omvatten, verwijderen van de policy die minder support en vertrouwen heeft.

29. Werkwijze volgens een van de voorgaande conclusies, waarbij de set met attribuut gebaseerde policies de witte lijst policies omvat, waarbij de uitkomst van de witte lijst policies

aangeven of het geselecteerde model toelaatbaar is, en waarbij, in het geval het model dat betrekking heeft op de eerste host, de tweede host of de link niet kan worden gematched met enige van de witte lijst policies, de datacommunicatie die betrekking heeft op de respectieve ene van de eerste host, de tweede host of de link wordt opgenoemd in een quarantaine.

30. Werkwijze volgens een van de voorgaande conclusies, waarbij, in het geval het netwerkbericht informatie draagt over een host of link waarvoor geen model beschikbaar is, de respectieve host voor link in een quarantaine wordt opgenoemd.

31. Werkwijze volgens conclusie 30, verder omvattende:
het afleiden van attribuutwaarden die gerelateerd zijn aan de host of link die in de quarantaine is opgenoemd.

32. Werkwijze volgens conclusie 31, verder omvattende:
het berekenen, uit de attribuutwaarden die gerelateerd zijn aan de host of link die in de quarantaine is opgenoemd, van een support voor een hypothese dat de host of link die in de quarantaine is opgenoemd legaal is, en
het berekenen, uit de attribuutwaarden die gerelateerd zijn aan de host of link die in de quarantaine is genoemd, van een support voor een hypothese dat de host of link die in de quarantaine is opgenoemd kwaadaardig is.

33. Werkwijze volgens conclusie 32, waarbij het berekenen van de supports voor de hypothesen wordt herhaald wanneer de attribuutwaarden die gerelateerd zijn aan de host of link die in de quarantaine is genoemd geupdated zijn.

34. Werkwijze volgens een van conclusies 30- 33, waarbij de quarantaine omvat, voor de host en/of links die in de quarantaine zijn opgenoemd:
een identificatie van de host of link,
een lijst met bekende attribuutwaarden van de host of link,
een support voor een hypothese dat de host of link legaal is,
een support voor een hypothese dat de host of link kwaadaardig is,
een identificatie van dataverkeer dat is gebruikt voor het bepalen van de support voor de hypotheses.

35. Werkwijze volgens een van conclusies 31-34, waarbij de werkwijze verder omvat:

het checken, gebruikmakend van de consistentieregels, van de attribuutwaarden van de host of link in quarantaine op consistentie.

36. Werkwijze volgens een van conclusies 31-35, waarbij de werkwijze verder omvat: het beoordelen, gebruikmakend van de attribuut gebaseerde policies, of de attribuutwaarden van de host of link in quarantaine voldoen aan de set met attribuut gebaseerde policies.

37. Werkwijze volgens een van conclusies 31-36, verder omvattende:
het afleiden van attribuut gebaseerde policies uit de host-attributen en de linkattributen van het dataverkeer dat in de quarantaine is opgeslagen.

38. Werkwijze volgens conclusie 37, waarbij een nieuwe witte lijst policy wordt afgeleid uit attribuutwaarden die zijn afgeleid uit protocolberichten die gerelateerd zijn aan de host of link in quarantaine, waarbij een frequentie van voorkomen van de protocolberichten die gerelateerd zijn aan de host of link in quarantaine, een witte lijst drempel overschrijdt.

39. Werkwijze volgens conclusie 37 of 38, waarbij het afleiden van attribuut gebaseerde policies uit de host-attributen en linkattributen van het dataverkeer dat in de quarantaine is opgeslagen omvat:

- het (her) berekenen van de support voor de hypothese dat de host of link in de quarantaine legaal of kwaadaardig is elke keer dat dataverkeer dat gerelateerd is aan de host en/of links in quarantaine wordt waargenomen,
- wanneer de support voor de hypothese dat de host of link in quarantaine legaal is groter is dan een witte lijst drempel, verwijderen van de host of link uit de quarantaine en gebruiken van het dataverkeer dat gerelateerd is aan de host of link voor het extraheren van nieuwe witte lijst policies,
- wanneer de support voor de hypothese dat een host/link in quarantaine kwaadaardig is groter is dan een zwarte lijst drempel, genereren van een alert en gebruik maken van het dataverkeer dat gerelateerd is aan de host of link voor het extraheren van nieuwe zwarte lijst policies,
- updaten van de huidige policies gebruikmakend van de geëxtraheerde nieuwe witte lijst of zwarte lijst policies.

40. Werkwijze volgens conclusie 39, waarbij het (her)berekenen van de support voor de hypothese dat de host of link in de quarantaine legaal of kwaadaardig is omvat:

- het berekenen van een soortgelijkheid van de host of link in quarantaine met een andere host of link.

41. Werkwijze volgens conclusie 39 of 40, waarbij het gebruiken van het dataverkeer dat gerelateerd is aan de host of link voor het extraheren van nieuwe witte lijst policies respectievelijk het gebruiken van het dataverkeer dat gerelateerd is aan de host of link voor het extraheren van nieuwe zwarte lijst policies omvat:

het leren van de witte lijst policies respectievelijk de zwarte lijst policies volgens een van conclusies 23-28.

42. Een intrusie-detectiesysteem dat ingericht is voor het uitvoeren van de werkwijze volgens een van de voorgaande conclusies.

43. Een datacommunicatienetwerk omvattende het intrusie-detectiesysteem volgens conclusie 42.

44. Een inrichting omvattende het intrusie-detectiesysteem volgens conclusie 42.

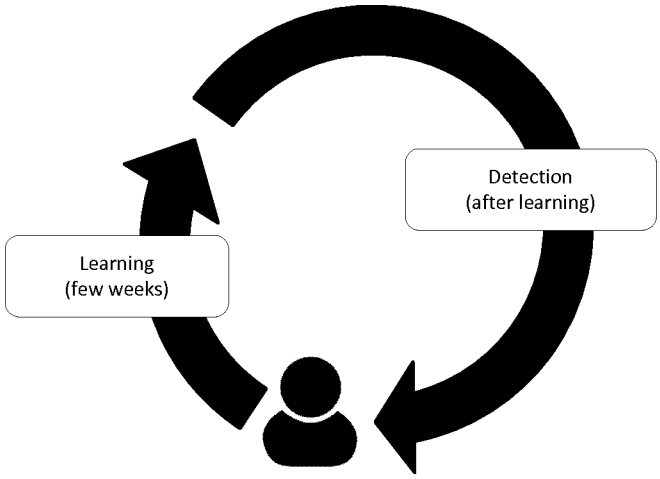


Figure 1

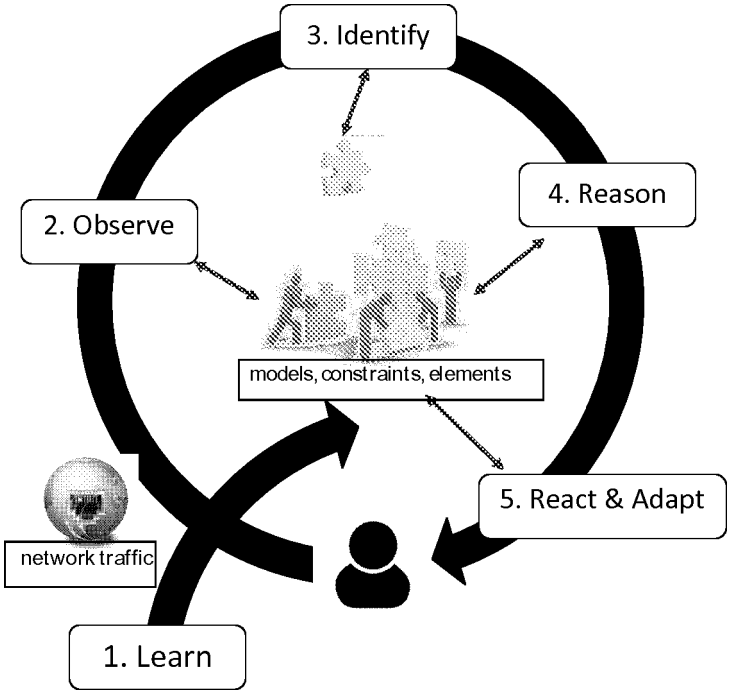

3. Identify
4. Reason
2. Observe
models, constraints, elements
5. React & Adapt
network traffic
1. Learn

Figure 2

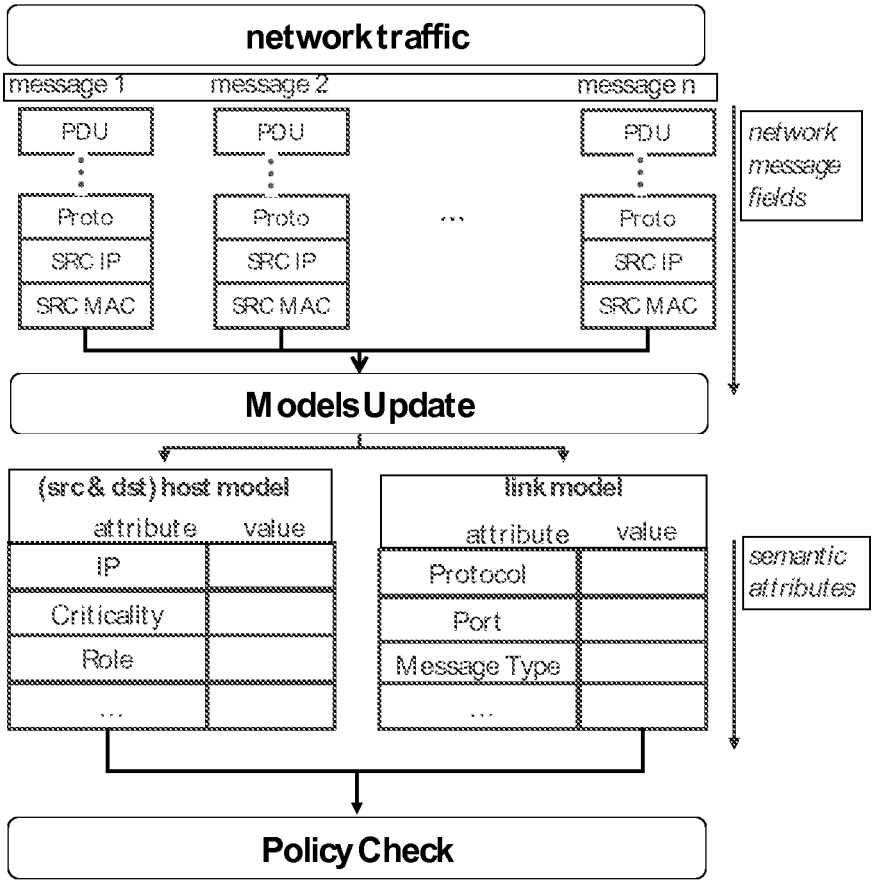

network traffic
message 1
message 2
message n
PDU
PDU
PDU
Proto
SRC IP
SRC MAC
Proto
SRC IP
SRC MAC
...
Proto
SRC IP
SRC MAC
network message fields
Models Update
(src & dst) host model
attribute
value
IP
Criticality
Role
...
link model
attribute
value
Protocol
Port
Message Type
...
semantic attributes
Policy Check

Figure 3

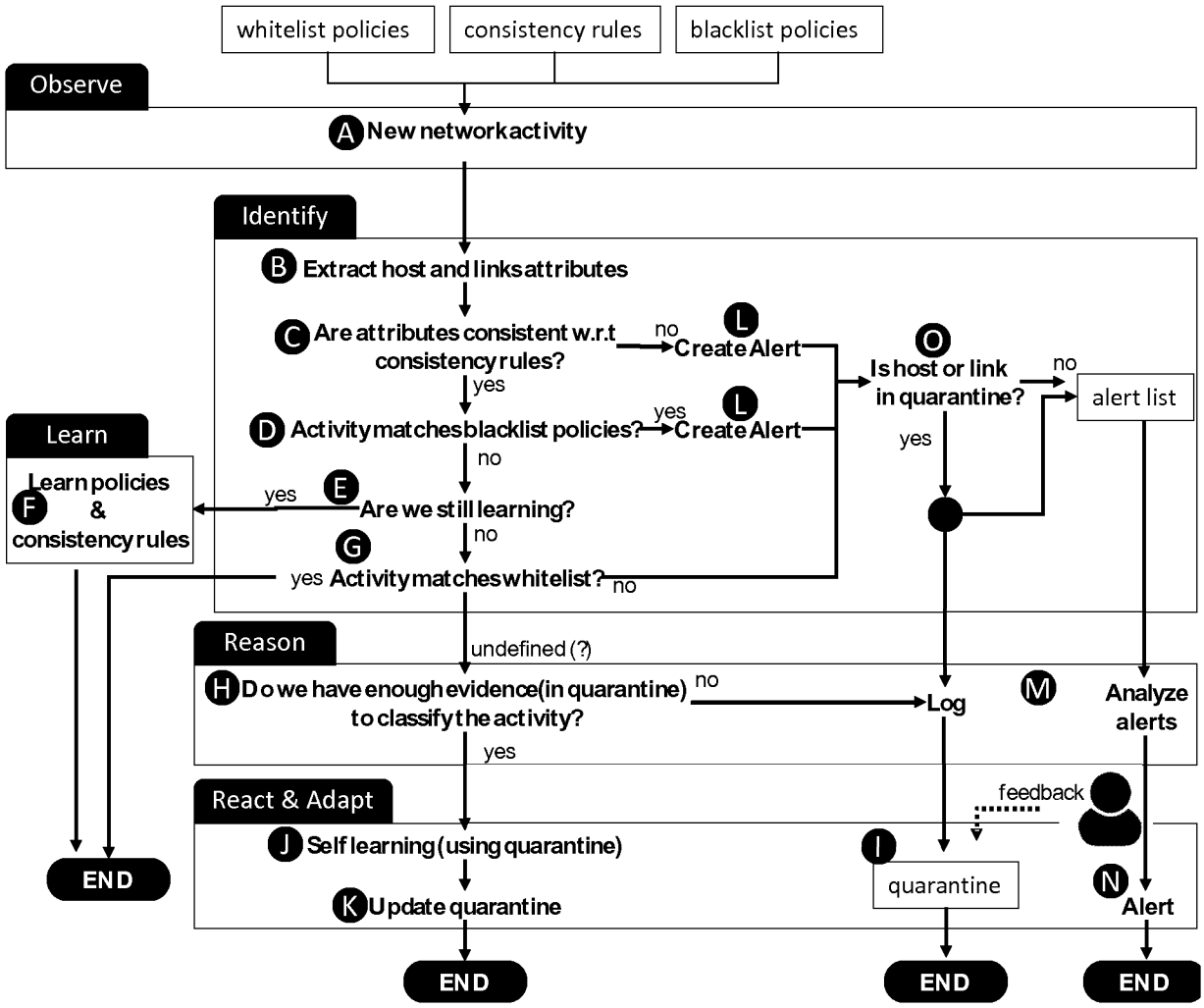

whitelist policies
consistency rules
blacklist policies
Observe
A New network activity
Identify
B Extract host and links attributes
C Are attributes consistent w.r.t consistency rules?
no
L Create Alert
yes
D Activity matches blacklist policies?
yes
L Create Alert
no
Learn
F Learn policies & consistency rules
yes
E Are we still learning?
no
G
yes Activity matches whitelist?
no
O Is host or link in quarantine?
no
alert list
yes
Reason
undefined (?)
H Do we have enough evidence (in quarantine) to classify the activity?
no
Log
M
Analyze alerts
yes
React & Adapt
feedback
J Self learning (using quarantine)
I
quarantine
N
Alert
K Update quarantine
END
END
END
END

Figure 4

Hosts DB					
Host	OS	IP	vendor	role	...
A	windows	10.10.1.1	?	SCADA	
B	windows	10.10.1.2	Dell	Terminal	
C	linux	10.10.2.3	Dell	DNS	
D	proprietary	172.1.41.5	Siemens	PLC	

Link DB						
Source	Target	Proto	Port	Message type	Link type	...
A	B	SMB	139	read file	cross-network	
B	A	SMB	139	open file	cross-network	
A	D	Modbus	502	read value	local	
A	D	STEP7	53	reprogram	local	
A	C	DNS	60	solve name	local	

Figure 5

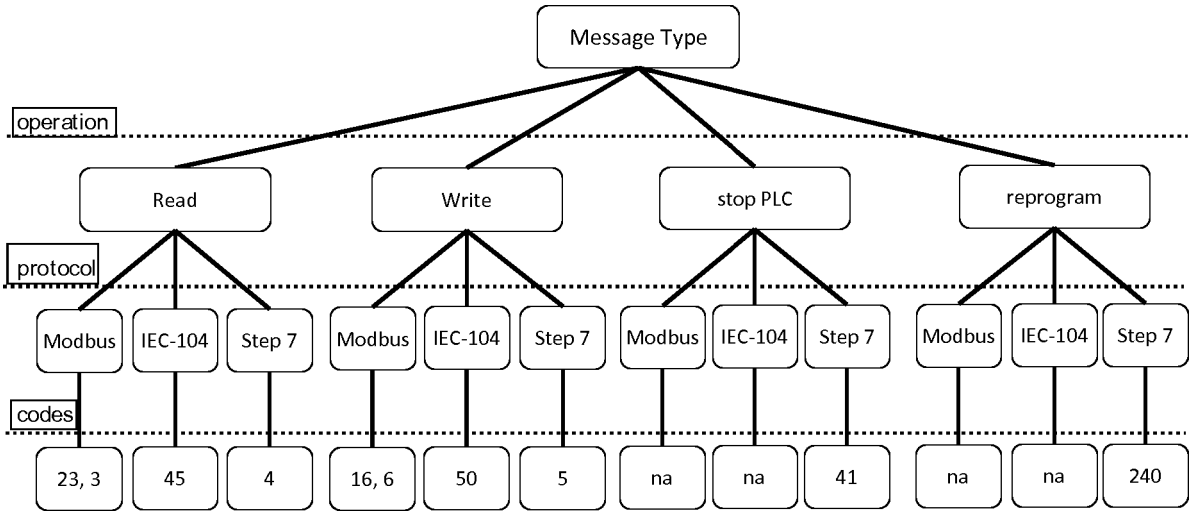
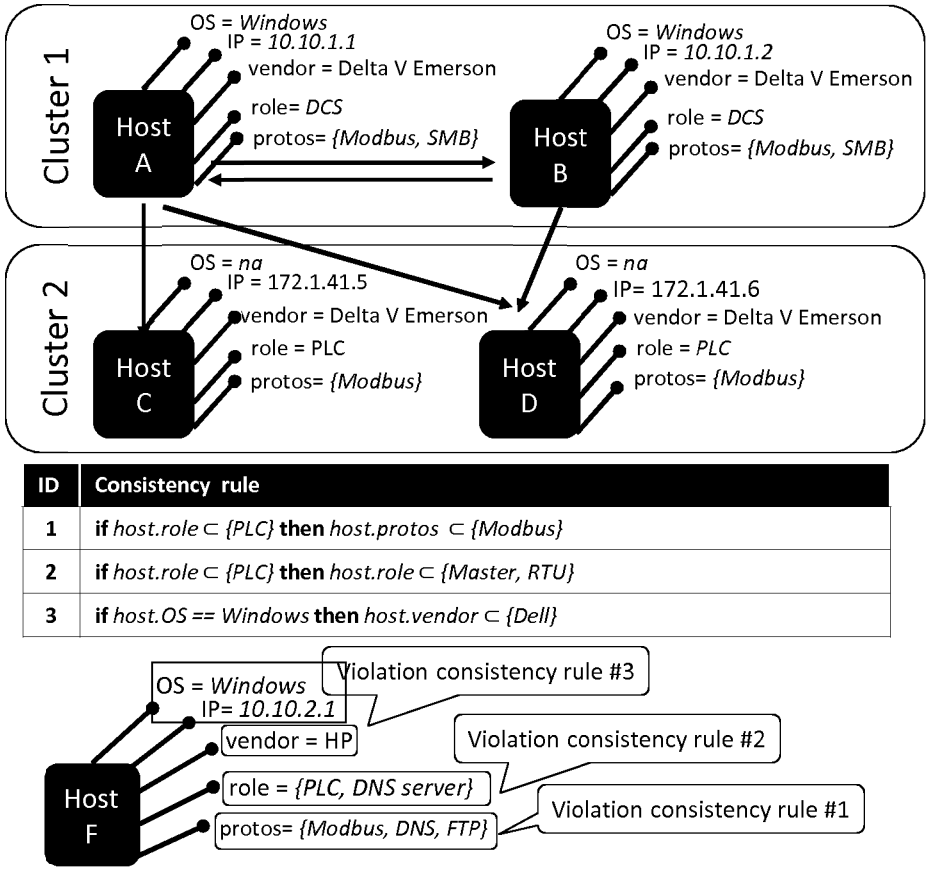

Message Type
operation
Read
Write
stop PLC
reprogram
protocol
Modbus
IEC-104
Step 7
Modbus
IEC-104
Step 7
Modbus
IEC-104
Step 7
Modbus
IEC-104
Step 7
codes
23, 3
45
4
16, 6
50
5
na
na
41
na
na
240

Figure 6



ID	Consistency rule
1	**if** *host.role* ⊂ *{PLC}* **then** *host.protos* ⊂ *{Modbus}*
2	**if** *host.role* ⊂ *{PLC}* **then** *host.role* ⊂ *{Master, RTU}*
3	**if** *host.OS == Windows* **then** *host.vendor* ⊂ *{Dell}*

Figure 7

Network traffic dataset										
ID	$L.H_{Src}$.OS	$L.H_{Src}$. vendor	$L.H_{Src}$.. role	$L.H_{dst}$.OS	$L.H_{dst}$. vendor	$L.H_{dst}$.role	L.Proto	L.DstPort	L.Message Type	...
1	windows	Dell	Terminal	windows	Dell	Terminal	SMB	139	read file	...
2	windows	Dell	Terminal	windows	Dell	Terminal	SMB	139	open file	...
3	windows	Dell	SCADA	proprietary	Siemens	PLC	Modbus	502	read value	...
4	windows	Dell	SCADA	proprietary	Siemens	PLC	STEP7	53	reprogram	...
...	...	...	...	...	...	...	...	...	...	...

Figure 8

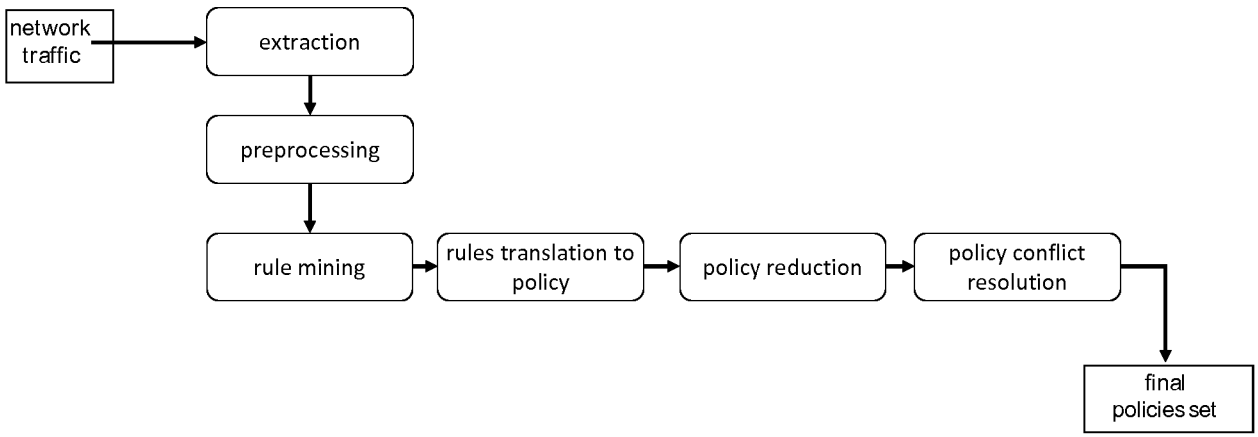

network traffic
extraction
preprocessing
rule mining
rules translation to policy
policy reduction
policy conflict resolution
final policies set

Figure 9

Quarantine					
ID	Target (G)	Support legit (Hyp_0)	Support malicious (Hyp_1)	Evidence (E)	Traffic (M)
1	**Host={**id = 1,..}	0.60 0.80 0.81 1.00	0.00 0.00 0.00 0.00	New Host New Host Attribute Value (role = **PLC**) New Host Attribute Value (vendor= **ABB**) User feedback (host is legit)	…
2	Link={ id=3, L.H.src.id = 2 L.H.dst.id =1,..}	0.00 0.00 0.00 1.00	0.25 0.70 0.90 0.00	New Link (from host 2 (role=terminal) to host 1 (role =PLC)) New Link Attribute Value (L.MessageType = **reprogram)** Alert (terminal reprograms PLC) User feedback (Terminal can reprogram PLC)	…
3	Link= {id=2, H.src.id = 1 H.dst.id =15, …}	0.50	0.10	New Link Attribute Value (L.MessageType=**read**)	…
…	…	…	…	…	…

Figure 10

ABSTRACT

A method of detecting anomalous behaviour in data traffic on a data communication network, a first host and a second host being connected to the data communication network, the data traffic on the data communication network forming a link between the first host and the second host, the method comprising:

a) parsing the data traffic to extract protocol field values of a protocol message of the data traffic;

b) deriving, from the extracted protocol field values, attribute values of attributes of one of the first host, the second host, and the link;

c) selecting from a set of models, a model relating to the one of the first host, the second host, and the link, wherein the selected model comprises a plurality of attributes to describe the one of the first host, the second host, and the link, wherein at least one of the attributes is a semantic attribute, the semantic attribute expressing a semantic meaning for the one of the first host, the second host, and the link,

d) updating the selected model with the derived attribute values, if the derived attribute values are not featured in the selected model upon selection;

e) assessing if the updated, selected model complies with a set of attribute based policies, each attribute based policy defining a security constraint of the data communication network based on at least one of the attributes of the first host, the second host or the link, and

f) generating an alert signal in case the attribute based policies indicate that the updated selected model violates at least one of the attribute based policies.

SAMENWERKINGSVERDRAG (PCT)

RAPPORT BETREFFENDE NIEUWHEIDSONDERZOEK VAN INTERNATIONAAL TYPE

IDENTIFICATIE VAN DE NATIONALE AANVRAGE	KENMERK VAN DE AANVRAGER OF VAN DE GEMACHTIGDE
	P33194NL00/HSE
Nederlands aanvraag nr. **2020552**	Indieningsdatum **08-03-2018**
	Ingeroepen voorrangsdatum
Aanvrager (Naam) **Security Matters B.V.**	
Datum van het verzoek voor een onderzoek van internationaal type **26-05-2018**	Door de Instantie voor Internationaal Onderzoek aan het verzoek voor een onderzoek van internationaal type toegekend nr. **SN71303**

I. CLASSIFICATIE VAN HET ONDERWERP (bij toepassing van verschillende classificaties, alle classificatiesymbolen opgeven)

Volgens de internationale classificatie (IPC)

H04L29/06

II. ONDERZOCHTE GEBIEDEN VAN DE TECHNIEK

Onderzochte minimumdocumentatie

Classificatiesysteem	Classificatiesymbolen
IPC	**H04L**

Onderzochte andere documentatie dan de minimum documentatie, voor zover dergelijke documenten in de onderzochte gebieden zijn opgenomen

III.	☐ **GEEN ONDERZOEK MOGELIJK VOOR BEPAALDE CONCLUSIES**	(opmerkingen op aanvullingsblad)
IV.	☐ **GEBREK AAN EENHEID VAN UITVINDING**	(opmerkingen op aanvullingsblad)

ONDERZOEKSRAPPORT BETREFFENDE HET RESULTAAT VAN HET ONDERZOEK NAAR DE STAND VAN DE TECHNIEK VAN HET INTERNATIONALE TYPE

Nummer van het verzoek om een onderzoek naar de stand van de techniek
NL 2020552

A. CLASSIFICATIE VAN HET ONDERWERP
INV. H04L29/06
ADD.

Volgens de Internationale Classificatie van octrooien (IPC) of zowel volgens de nationale classificatie als volgens de IPC

B. ONDERZOCHTE GEBIEDEN VAN DE TECHNIEK

Onderzochte minimum documentatie (classificatie gevolgd door classificatiesymbolen)
H04L

Onderzochte andere documentatie dan de minimum documentatie, voor dergelijke documenten, voor zover dergelijke documenten in de onderzochte gebieden zijn opgenomen

Tijdens het onderzoek geraadpleegde elektronische gegevensbestanden (naam van de gegevensbestanden en, waar uitvoerbaar, gebruikte trefwoorden)
EPO-Internal, COMPENDEX, INSPEC, IBM-TDB, WPI Data

C. VAN BELANG GEACHTE DOCUMENTEN

Categorie °	Geciteerde documenten, eventueel met aanduiding van speciaal van belang zijnde passages	Van belang voor conclusie nr.
X	EP 1 986 391 A1 (MITSUBISHI ELECTRIC CORP [JP]; MITSUBISHI ELECTRIC INF TECH [NL]) 29 oktober 2008 (2008-10-29) * alineas [0035] - [0053]; figuur 3b *	1-44
A	US 2017/195197 A1 (ZAMBON EMMANUELE [NL]) 6 juli 2017 (2017-07-06) * alineas [0015] - [0033], [0044], [0045], [0056], [0059], [0069], [0070] *	1-44
A	US 2011/167493 A1 (SONG YINGBO [US] ET AL) 7 juli 2011 (2011-07-07) * alineas [0053] - [0056], [0069], [0071] *	1-44

-/--

[X] Verdere documenten worden vermeld in het vervolg van vak C.

[X] Leden van dezelfde octrooifamilie zijn vermeld in een bijlage

° Speciale categorieën van aangehaalde documenten

"A" niet tot de categorie X of Y behorende literatuur die de stand van de techniek beschrijft

"D" in de octrooiaanvrage vermeld

"E" eerdere octrooi(aanvrage), gepubliceerd op of na de indieningsdatum, waarin dezelfde uitvinding wordt beschreven

"L" om andere redenen vermelde literatuur

"O" niet-schriftelijke stand van de techniek

"P" tussen de voorrangsdatum en de indieningsdatum gepubliceerde literatuur

"T" na de indieningsdatum of de voorrangsdatum gepubliceerde literatuur die niet bezwarend is voor de octrooiaanvrage, maar wordt vermeld ter verheldering van de theorie of het principe dat ten grondslag ligt aan de uitvinding

"X" de conclusie wordt als niet nieuw of niet inventief beschouwd ten opzichte van deze literatuur

"Y" de conclusie wordt als niet inventief beschouwd ten opzichte van de combinatie van deze literatuur met andere geciteerde literatuur van dezelfde categorie, waarbij de combinatie voor de vakman voor de hand liggend wordt geacht

"&" lid van dezelfde octrooifamilie of overeenkomstige octrooipublicatie

Datum waarop het onderzoek naar de stand van de techniek van internationaal type werd voltooid	Verzenddatum van het rapport van het onderzoek naar de stand van de techniek van internationaal type
23 oktober 2018	

Naam en adres van de instantie	De bevoegde ambtenaar
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Agudo Cortada, E

1

ONDERZOEKSRAPPORT BETREFFENDE HET RESULTAAT VAN HET ONDERZOEK NAAR DE STAND VAN DE TECHNIEK VAN HET INTERNATIONALE TYPE

Nummer van het verzoek om een onderzoek naar de stand van de techniek
NL 2020552

C.(Vervolg) VAN BELANG GEACHTE DOCUMENTEN

Categorie *	Geciteerde documenten, eventueel met aanduiding van speciaal van belang zijnde passages	Van belang voor conclusie nr.
A	US 2017/163666 A1 (VENKATRAMANI ANJAN [US] ET AL) 8 juni 2017 (2017-06-08) * samenvatting * * alineas [0038], [0039], [0048], [0070] - [0092], [0144] * -----	1-44
A	GARCIA-TEODORO P ET AL: "Anomaly-based network intrusion detection: Techniques, systems and challenges", COMPUTERS & SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, deel 28, nr. 1-2, 1 februari 2009 (2009-02-01), bladzijden 18-28, XP025839371, ISSN: 0167-4048, DOI: 10.1016/J.COSE.2008.08.003 [gevonden op 2008-08-27] * section 2.2 * -----	1-44

Formulier PCT/ISA/201 (vervolg tweede blad) (Januari 2004)

ONDERZOEKSRAPPORT BETREFFENDE HET RESULTAAT VAN HET ONDERZOEK NAAR DE STAND VAN DE TECHNIEK VAN HET INTERNATIONALE TYPE

Informatie over leden van dezelfde octrooifamilie

Nummer van het verzoek om een onderzoek naar de stand van de techniek: NL 2020552

In het rapport genoemd octrooigeschrift	Datum van publicatie	Overeenkomend(e) geschrift(en)	Datum van publicatie
EP 1986391 A1	29-10-2008	EP 1986391 A1	29-10-2008
		JP 2008306706 A	18-12-2008
		US 2008263661 A1	23-10-2008
US 2017195197 A1	06-07-2017	BR 112014001691 A2	13-06-2017
		CA 2842465 A1	31-01-2013
		CN 103748853 A	23-04-2014
		EA 201490333 A1	30-06-2014
		EP 2737683 A1	04-06-2014
		ES 2581053 T3	31-08-2016
		IL 230440 A	31-10-2017
		IL 254829 A	28-06-2018
		JP 6117202 B2	19-04-2017
		JP 2014522167 A	28-08-2014
		NL 2007180 C	29-01-2013
		US 2014090054 A1	27-03-2014
		US 2014297572 A1	02-10-2014
		US 2017195197 A1	06-07-2017
		WO 2013015691 A1	31-01-2013
US 2011167493 A1	07-07-2011	US 2011167493 A1	07-07-2011
		US 2014373150 A1	18-12-2014
		US 2016366169 A1	15-12-2016
		WO 2010011411 A1	28-01-2010
US 2017163666 A1	08-06-2017	EP 3387517 A1	17-10-2018
		US 2017163666 A1	08-06-2017
		WO 2017100364 A1	15-06-2017

WRITTEN OPINION

File No.	Filing date *(day/month/year)*	Priority date *(day/month/year)*	Application No.
SN71303	08.03.2018		NL2020552

International Patent Classification (IPC)
INV. H04L29/06

Applicant
Security Matters B.V.

This opinion contains indications relating to the following items:

- ☒ Box No. I Basis of the opinion
- ☐ Box No. II Priority
- ☐ Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- ☐ Box No. IV Lack of unity of invention
- ☒ Box No. V Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- ☐ Box No. VI Certain documents cited
- ☒ Box No. VII Certain defects in the application
- ☒ Box No. VIII Certain observations on the application

Examiner
Agudo Cortada, E

Application number

WRITTEN OPINION

NL2020552

Box No. I Basis of this opinion

1. This opinion has been established on the basis of the latest set of claims filed before the start of the search.

2. With regard to any **nucleotide and/or amino acid sequence** disclosed in the application and necessary to the claimed invention, this opinion has been established on the basis of:

   a. type of material:

   ☐ a sequence listing

   ☐ table(s) related to the sequence listing

   b. format of material:

   ☐ on paper

   ☐ in electronic form

   c. time of filing/furnishing:

   ☐ contained in the application as filed.

   ☐ filed together with the application in electronic form.

   ☐ furnished subsequently for the purposes of search.

3. ☐ In addition, in the case that more than one version or copy of a sequence listing and/or table relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

4. Additional comments:

Box No. V Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty	Yes:	Claims	1-44
	No:	Claims	
Inventive step	Yes:	Claims	
	No:	Claims	1-44
Industrial applicability	Yes:	Claims	1-44
	No:	Claims	

2. Citations and explanations

see separate sheet

WRITTEN OPINION

Box No. VII Certain defects in the application

see separate sheet

Box No. VIII Certain observations on the application

see separate sheet

Re Item V

Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

Reference is made to the following documents:

D1 EP 1 986 391 A1 (MITSUBISHI ELECTRIC CORP [JP]; MITSUBISHI ELECTRIC INF TECH [NL]) 29 oktober 2008 (2008-10-29)

D2 US 2017/195197 A1 (ZAMBON EMMANUELE [NL]) 6 juli 2017 (2017-07-06)

D3 US 2011/167493 A1 (SONG YINGBO [US] ET AL) 7 juli 2011 (2011-07-07)

D4 US 2017/163666 A1 (VENKATRAMANI ANJAN [US] ET AL) 8 juni 2017 (2017-06-08)

D5 GARCIA-TEODORO P ET AL: "Anomaly-based network intrusion detection: Techniques, systems and challenges", COMPUTERS & SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, deel 28, nr. 1-2, 1 februari 2009 (2009-02-01), bladzijden 18-28, XP025839371, ISSN: 0167-4048, DOI: 10.1016/J.COSE.2008.08.003 [gevonden op 2008-08-27]

1 The present application does not meet the criteria of patentability, because the subject-matter of claim 1 does not involve an inventive step.

1.1 D1 is regarded as being the prior art closest to the subject-matter of claim 1 and discloses a method of detecting anomalous behaviour in data traffic on a data communication network, a first host and a second host being connected to the data communication network, the data traffic on the data communication network providing a link between the first host and the second host, the method comprising:

a) parsing the data traffic to extract protocol field values of a protocol message of the data traffic (see D1, paragraph 50: "*in step 319 attributes are extracted from the received signaling flow. This extraction is done as explained above in relation to step 305*", which implies a

parsing of the data traffic);

b) deriving, from the extracted protocol field values, attribute values of attributes of one of the first host, the second host, and the link (see D1, paragraph 38: "*The attributes of the first class correspond to different attributes that are intrinsic to the VoIP protocol, particularly to SIP. Table 1 presents a non exhaustive list of attributes of this class*" and paragraph 40: "*The attributes of the second class are obtained by calculating correlation measures between the different signaling flows preceding the current one using the different values of the attributes indicated in the first class*");

~~c) selecting from a set of models, a model relating to the one of the first host, the second host, and the link,~~ wherein ~~the selected~~ a model comprises a plurality of attributes to describe the one of the first host, the second host, and the link (see D1, table 1, wherein some of the fields relate to the hosts or to the link), wherein at least one of the attributes is a semantic attribute, the semantic attribute expressing a semantic meaning for the one of the first host, the second host, and the link (see D1, tables 1 and 2 in paragraphs 38 and 40, wherein the second column is the meaning and for some attributes such as the URI relate to the hosts and/or the link, such as though the port indication);

d) updating the ~~selected~~ model with the derived attribute values, if the derived attribute values are not featured in the selected model upon selection (see D1, paragraph 51: "*Once the attributes are extracted, the received signaling flow is classified in step 321 based on the classification model built earlier in step 307*", wherein the derived attribute values from the previous step need to be input into the model corresponding location for the attributes, which can be considered empty and updated by the derived attribute values);

e) assessing if the updated~~, selected~~ model complies with a set of attribute based policies, each attribute based policy defining a security constraint of the data communication network based on at least one of the attributes of the first host, the second host or the link (see D1,

paragraph 51 mentioned in the previous step), and

f) generating an alert signal in case the attribute based policies indicate that the updated selected model violates at least one of the attribute based policies (see D1, paragraph 52: "*in step 323 it is determined whether the classification result corresponds to a known attack. If this is the case, then in step 325 an alert is generated*").

1.2 The subject-matter of claim 1 therefore differs from this known method in that a selection is made from a set of models of a model relating to the one of the first host, the second host, and the link and is therefore new.

1.3 The problem to be solved by the present invention may therefore be regarded as the use of the method known from D1 for a plurality of hosts or for a different type of link having a distinct behaviour which warrant a different model.

1.4 The solution proposed in claim 1 of the present application cannot be considered as involving an inventive step for the following reasons:

1.4.1 D1 already defines a decision tree classifier (see paragraph 58 and figure 5) which produces a set of branching policy conditions. In the example of figure 5, the kind of message (Request/Response) is determined at the root and the corresponding of two different (sub-)models for testing the anomalous behaviour depending on the type of message is selected.

1.4.2 It is straight-forward to have a first condition at the root for hosts or links having a clearly distinct behaviour which warrant a different model. Actually, even the derivation of the model known from D1 (see paragraphs 43-49) would naturally produce a decision tree with the transmitting/receiving host or the link type at the root if the hosts or links exhibit a distinguishing behaviour for the purpose of anomalous behaviour detection.

1.4.3 It is worth pointing out that the attributes corresponding to the the transmitting/receiving host or the link for which the model is specific could already comprise the corresponding attributes from the transmitting/receiving host or the link which need not be obtained from the data traffic. This is in relation to the condition "if the derived attribute values are not featured in the selected model upon selection" in step d) of the claim.

2 The same reasoning applies, mutatis mutandis, to the subject-matter of the corresponding independent claims 42-44, which therefore are also considered not inventive.

3 Dependent claims 2-41 do not contain any features which, in combination with the features of any claim to which they refer, meet the requirements of inventive step.

Claim 2 is known from D1 (see D1, paragraph 40). Claim 3 and 5 are derivable from the above argumentation regarding claim 1. Claim 4 and is known from D1 (se for instance figure 5). Claim 7 is known from D1, paragraph 38. Claim 8 is known from D1 (see D1, paragraph 40, wherein the correlation operations can be considered to be an heuristic). Claim 9 is known from D1 (see D1, paragraph 38, wherein a classification is implicit by the obtention of the attributes). Claim 10 is customary in the art of data processing, wherein a non-reliable information input is disregarded, wherein the reliability can be considered to be a level of confidence. Claim 11 defines a feature lacking a technical significance since it makes no sense to obtain two attribute values, one employing heuristic methods and another employing a classifier, if the latter is always discarded as defined in the claim. Claim 12 is known from D1, which does not define a passive detection. Claim 13 is derivable from the above argumentation regarding claim 1. Claims 14 and 15 are known from D1 (paragraphs 53 and 52 respectively). Claim 16 is derivable from D2, paragraph 20 or from D3 (paragraph 69), whose teachings can be straight-forwardly used in the method known from D1. Claim 17-41 merely define additional implementation details.

Re item VII

Certain defects in the international application

1 The independent claims should be cast in the two-part form, with those features known in combination from the prior art (see document D1) being placed in a preamble and with the remaining features being included in a characterising part.

2 Documents D1 to D3 should be identified in the description and the relevant background art disclosed therein briefly discussed.

3 The opening part of the description should be brought into conformity with any amended independent claims.

4 Furthermore, following the disclosure of document D1, the statement indicating the technical problem to be solved by the invention, requires revision.

5 Due care should be taken not to add any subject-matter which extends beyond the content of the application as originally filed.

6 Reference signs placed in parentheses should be inserted into all the claims to increase their intelligibility. This applies to both the preamble and the characterising portion.

Re Item VIII

Certain observations on the application

1 Independent claim 1 is not clear for the following reasons (wherein the deficiencies mentioned also applies to dependent claims employing the unclear expressions mentioned below):

1.1 Reference is made in step a) to a "protocol field" and to a "protocol message", which lacks a well-defined meaning, rendering the scope of the claim unclear. In particular, it is not clear which characteristics should be satisfied by a field and by a message in order to be considered to qualify as being of the "protocol" type.

1.2 Step c) refers to a "semantic value" expressing a "semantic meaning" of the first host, the second host, and the link, which is unclear. In particular, it is not understood what is a "semantic meaning" of hosts or links.

1.3 Step d) refers to updating the selected model with the derived attribute values, "if the derived attribute values are not featured in the selected model", which is broader than justified by the description, since according to the wording the attribute values could appear in a different attribute in order for the condition to be met (for instance, the value of the source IP attribute obtained by the deriving step in b) of claim 1 could correspond to the destination IP attribute of the model for the link in the example in page 12, line 13-page 13, line 33). However, support exists only for the condition being instead that a value for the corresponding attribute in the selected model has not been defined (represented by a question tag "?" as the value of the attribute in the example of pages 12-13).

1.4 The claim indicates in the introductory part that the data traffic provides a link between the first and the second host, which is unclear, since a link needs to exist in order to enable data traffic to exists between the two hosts and therefore the traffic cannot provide a link.

2 The reference to injecting a "stimulus" into the network made in dependent claim 12 is unclear. In particular, which kind of signals are considered to be a stimulus.

3 Dependent claim 16 refers to "combinations of at least two attributes" as being consistent or not, but it appears that it intends to refer to the combination of attribute values as being consistent or not (see for instance claim 18).

4 Dependent claim 17 specifies that the "the detecting, on the basis of the attributes derived from the monitored data traffic, if the monitored data traffic complies to the consistency rule, is performed before step e)".

First of all, no detecting step has been previously defined, rendering the reference to "the detecting" unclear. Secondly, step e) of claim 1 defines assessing if the updated, selected model complies with a set of attribute based policies and the feature which, according to claim 17, should occur before, i.e. detecting whether the traffic complies to the consistency rule, is itself a particular manner to perform step e), since the consistency rule is a specific type of policy. This claim is therefore unclear, since it defines that a sub-step of a step occurs prior to the step.

5 Dependent claim 18 refers to a "location" of data traffic which is unclear. This wording is sometimes used to refer to the time at which the data traffic occurs, but since the claim also refers to the time of occurrence of the data traffic as an alternative, the "location" necessarily should have a different meaning.

6 Dependent claim 19 specifies that the consistency rule related to the first host comprises an attribute of another host. This is unclear for a plurality of reasons. First of all, claim 16 on which it depends does not define a "consistency rule related to the first host" but rather a consistency rule which "defines consistent combinations of at least two attributes of the model related to the first host". Secondly, assuming that the wording of claim 19 intends to refer to the consistency rule defining consistent combinations of at least two attributes of the model related to the first host, according to claim 1, the model related to the first host comprises attributes of the first host, whereas claim 19 specifies the opposite, namely that it comprises an attribute of another host.

7 Claim 22 refers to "the action", whereas no action has been previously defined.