Abstract- Here is presented a way to detect configuration inconsistencies that may affect a communication when multiple intermediate systems are crossed and modify data units they receive to match a protocol or a security policy.

1. INTRODUCTION

Setting up a communication between systems connected to a computer network often implies to cross many equipment belonging to many third parties. Each device receive either packets, frames and execute specific functions like retransmission, routing, firewalling, blocking, modifying some fields, etc. to be compliant with a policy design by a user or an administrator. Keeping a global view of all treatments executed by these devices is inherently a distributed and complex task [4]. Vendors are providing devices and generally a specific language is provided to create a file containing multiple rules. Inconsistencies in the configurations of network equipment are frequently encountered due to misconfiguration: errors are sometime voluntary, majorly not, but may cause serious security problems. Our objective is to bring a modelling system and a tool to help users, operators (administrators) and designers (engineers) to be sure that crossing a network will not be a source of problem. Our modelling system is based on a data flow representation, aim to study the feasibility of specific configuration and make the right decision in analysing the impact of each security device. (Clearly is it possible to combine different transformations inside a single or multiple systems). For example on a single system, two rules or mechanism be may opposite (e.g. one filtering rule allows a data flow to cross an interface while another one forbid it). An end to end communication requires the use of a specific TCP port number, but datagram are encrypted and while crossing a firewall the port number cannot be recognizable.

2. DATAFLOW REPRESENTATION

2.1. Data flow founding principles

In the OSI, IEEE, TCP/IP model, a frame is the result of a set of protocol encapsulation chain. Each protocol add new fields and affect values to those fields. One may analyse a frame and give all the different fields inside the frame. The data flow is composed by the set of protocol data unit that are transmitted from the source system to the destination one crossing multiple intermediate systems. Bridge, routeur, firewall… are transforming the data flow they receive into a new one depending on the crossed-system functionality and protocols that are run.

Protocol data unit (PDU) which are delivered to a system can be characterized using different views. A list of the protocol that were used to produce the PDU may be given, the list of fields accessible in the PDU, the list of fields for which the value contain an authenticated value, the list of fields that cannot be modified without leading to an error, the list of algorithm used to encrypt, an array that link the fields that are protected by an algorithm….  

2.2. Formal Data flow model

The basics of the formal data flow model have been introduced in[1] except \( \mathcal{L} \) set :

- \( \mathcal{A} \) is the set of possible attributes. When an attribute \( a \in \mathcal{A} \), it mean it exist a couple \(<\text{name}, \text{value}>\) where name is a field that can be found while a protocol is executed, and value is its content. For example an ip adress is an attribute linked to IP protocol.
- \( \mathcal{P} \) is the set of protocols, i.e., the set of logical blocks. Each protocol need to be identified and claim one ore more fields to be updated during its execution. Thus an instance of protocol \( p \in \mathcal{P} \) can be defined as a couple \(<\text{protoid}, \text{attributes}>\) where a) \( \text{protoid}=<\text{name}, \text{id}> \) is the name of the protocol and a unique identifier, b) attributes are defined on the Power-set of \( \mathcal{A} \), i.e., attributes \( \in \mathcal{P}(\mathcal{A}) \).
- \( \mathcal{S} \) is the set of security algorithms that can be run during the encapsulation chain of protocols (for instance, DES, 3DES, HMAC-SHA1, etc.)
- \( \mathcal{L} = \{\text{all, val}\} \) has been added into[2] in order to determine what is possible or not regarding an attribute. It give the state of an attribute that has been encrypted. If the attribute is completely encrypted (tag all), then it is not possible to get the attribute. When only its value is encrypted (tag val), the attribute can be accessed but its value cannot be retrieved except if the associated secret/keys are reachable.

Technically, an incorrect execution of a chain of protocols often implies that one or more fields contained inside a PDU are missing, wrong, suspect, impossible to retrieve, or to be replaced by a new one. Some protocol claim that payload and protocol control information must be encrypted, other one claim to discard any change of a fields because some of them are protected, encrypted and/or must not be replace or retrieve. This analysis led us to modelize a dataflow as :

\[ \mathcal{F} \subseteq \mathcal{E} \times \mathcal{AUTHN} \times \mathcal{CONF} \]

- \( \mathcal{E} \) is the encapsulation chain of protocols. For example \(<\text{HTTP, TCP, IP, CSMA/CA}>\),
- \( \mathcal{AUTHN} \subseteq (\mathcal{A} \times \mathcal{P} \times \mathcal{A} \times \mathcal{P} \times \mathcal{S}) \) contains the attributes of the data flow that have been authenticated. If \( (a_1, p_1, a_2, p_2, s) \in \mathcal{AUTHN} \) then it means that attribute \( a_1 \) of protocol \( p_1 \) guarantees the integrity of attribute \( a_2 \) of protocol \( p_2 \) via the security algorithm \( s \). For example (FCS_field, IEEE802.3, ipdest_field, IP_protocol, CRC_32) means that frame check sequence field generated using IEEE 802.3 protocol guarantees the integrity of IP destination fill in during IP protocol execution, while using CRC_32 algorithm implemented in IEEE802.3 solution.
• CONF \subseteq \text{BAG}(\mathcal{A} \times \mathcal{P} \times S \times L)$ represents the attributes of the data flow that have been encrypted, such that: $(a, p, s, all) \in \text{CONF}$ stand for attribute $a$ of protocol $p$ is completely encrypted via the security algorithm $s$. 

2.3. Data flow Operators

Operators must be provided to handle information contained inside the protocol list AUTHN, CONF sets assigned to a dataflow $f$. Here, they are (given into an intuitive form):

- $\text{proto} \leftarrow \text{Get\_Protocol}(f, \text{protoid})$
- $\text{attribute} \leftarrow \text{Get\_Attribute}(f, \text{protoid}, \text{attName})$
- $\text{flow} \leftarrow \text{Modify\_Attribute}(f, \text{protoid}, \text{attribute})$

Modifies an attribute belonging to a specific protocol. Before modifying an attribute it must belong to a protocol and must be readable.

- $\text{flow} \leftarrow \text{Add\_Proto}(f, \text{proto}, \text{protoid})$
- $\text{flow} \leftarrow \text{Delete\_Proto}(f, \text{protoid})$
- $\text{flow} \leftarrow \text{Add\_AUTHN}(f, \text{att1}, \text{proto1, att2, protoid2, algo})$
- $\text{flow} \leftarrow \text{Delete\_AUTHN}(f, \text{att1, proto1, att2, protoid2, algo})$
- $\text{flow} \leftarrow \text{Add\_CONF}(f, \text{attribute, protoid, algo, level})$
- $\text{flow} \leftarrow \text{Delete\_CONF}(f, \text{attribute, protoid, algo, level})$

3. Conflict detection using Petri nets SAMPL

Colored Petri Nets [3] are a formal specification language consisting of a set of tokens whose type is represented by a color, a set of transitions, and a set of places with a domain (which defines the types of tokens that can be stored in that place). They are well-known for their graphical and analytical capabilities for the specification and verification of concurrent, asynchronous, distributed, parallel and nondeterministic systems.

IPsec [5] and NAT are well known protocols that are not applicable one following the other [6]. Using the dataflow specification, we can detect conflicts while studying the impact of the transformation chain $t_{\text{CONF}}$ in净值($f$) on the given data flow $f = (\langle \eta, \gamma, \tau, \rho, \sigma \rangle, \{\}, \{\})$.

The IPsec/AH (Authentication Header) is designed to ensure integrity and authenticity of IP datagrams without data encryption. The integrity is guaranteed by the Authentication Data (AD) field. The AH protocol has two modes: transport and tunnel. In transport mode, AH is inserted between the IP header and the next layer. It protects the entire IP packet except for the mutable fields (i.e. the fields DSCP, ECN, Flags, Offset, TTL, Header Checksum). In tunnel mode, the inner IP header carries the ultimate source and destination addresses, while an outer IP header contains the addresses of the IPSec peers. It protects the entire inner IP packet. In AH Tunnel mode, the entire original IP header and data become the “payload” for the new packet protected by AD field. Figure 1 presents a sample of a Petri that is built. Data flow is transformed into $f' = f_{\text{CONF}}^{\text{AH}}(f(\langle \text{ip2, ah, ip1, tcp1}, \text{AUTHN, } \{\} \rangle))$. First, as the dataflow contain a classical IP packet $ip1$, before its transformation, then the attributes can be retrieved, and tunnel mode can operates: action$_{\text{AH}}, \text{Tunnel}$ is implemented using a function ApplyIPsec($f$, proto-ipsec, mode, gw, algo) with the parameters: (1) the data flow $f$, (2) the protocol, (3) the mode, (4) the gateway and (5) the cryptographic algorithm hmac-md5.

Next, the NAPT mechanism attempt to perform Action$_{\text{NAPT}}$, which is an expression calling Modify$_{\_\text{Attribute}}()$ in order to change the dest, add, the dest, port and the checksum. But when the NAPT operates, the token representing data flow $f'$ will stay blocked inside it. Two different explanation explain this situation: a) Basic NAPT refers to a situation where the payload field can’t be examined at all. The flow $f'$ can’t be transformed because the protocol encapsulated directly after IP1 is AH and that protocol does not contain any attribute called ports b) In Advanced NAPT configuration the former IP1, TCP1 attributes are considered to be accessible. Port value may be retrieved using Get_attribute. Consequently they could also be modified but since the attribute “port” belongs to and is protected by the set AUTHN, the modify attribute could not be executed: $f''$ will not be produced and the error will then be detected.

4. Conclusion

Network security requires the coordination of various heterogeneous and interdependent devices but conflicts may occur while these devices are crossed. Here has been presented the basics of a formal approach using a data flow oriented-framework. If only one example has been given here, it was applied to other well know protocols. Conflicts have been detected without requiring any a priori knowledge or experience. An other ongoing work is to design a generic model for network security mechanisms.

5. References