INXU - A Security Extension for RFC 8520 to Give Fast Response to New Vulnerabilities on Domestic IoT Networks

Sávyo V. Morais[1], **Claudio M. Farias**[1]

[1]Programa de Pós Graduação em Informática (PPGI) – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{savyo.morais,cmiceli}@labnet.nce.ufrj.br

Abstract. *As domestic Internet of Things (DIoT) devices become more popular, the number of devices connected to the Internet increases. It may also represent a risk to the end-user's security and privacy. The infected devices can be used in DIoT botnets affecting the Internet's stability. Although there are efforts to enhance IoT security, such as RFC 8520, there still needs for improvements in the DIoT context. To ensure DIoT security, this paper proposes INXU, an extension of RFC 8520 that enables blocking traffic related to well-known malicious activities. INXU introduces the concept of Malicious Traffic Description, a data model to describe traffic related to malicious activities, and enables Security Operation Centers to protect domestic networks.*

1. Introduction

The Internet of Things (IoT) is a socio-technological phenomena that comes from the human need to monitor and control the environment in which is inserted, jointed with the growing development of Information and Communication Technologies during the last two decades [Kramp et al. 2013]. Thus, the IoT comes assuming its role in transforming devices that originally are incommunicable – such as fridges, doors, cars, other common objects of our daily lives, or even environments with sensors and actuators – into devices communicable from any part of the world *via* Internet. This enables automation of tasks, replacing manual activities, and accelerating the growth of the number of devices connected to the Internet that went from 8.4 billion in 2016 to the forecast of 20.4 billion in 2020 [van der Meulen 2017].

Currently, one of the most common uses of IoT is in domestic environments. In this context, much data about the end-user's daily life can be extracted from IoT devices, putting at risk its privacy and security if an attacker get access to the devices. One example of this is Shodan[1], an online platform that enables users to access vulnerable security cameras connected to the Internet [Lin and Bergmann 2016]. Besides privacy, the exposure to cyber-physical systems can incur in physical harm, such as if a denial of service attack disables a smoke alarm during a fire [Habibi Gharakheili et al. 2019], or an attacker maliciously controlling remotely a thermostat.

The vulnerability of these devices causes troubles not only to its end-users since that most of the devices are being infected and incorporated into networks of remotely controlled devices that are being used to drive malicious activities on the Internet, the so-called botnets [Marzano et al. 2018]. One of the most common uses of botnets is on

[1]http://shodan.io/ – accessed in 08/26/2020

the interruption of online services through Distributed Denial of Service (DDoS) attacks, what is affecting the Internet's stability, given that such attacks commonly take advantage of Domain Name System (DNS) infrastructure to amplify the attacks [Schutijser 2018]. Another common use of botnets is on mining cryptocurrencies, where the funds generated are credited to the attacker, but the costs are transferred to the infected IoT devices owners [Pires et al. 2019].

To enhance security on IoT devices, the Internet Engineering Task Force (IETF) released the RFC 8520 – Manufacturer Usage Description (MUD) Specification –, an Internet Standard that defines a way for the manufacturers to describe the minimal network configuration that each IoT device needs to properly work [Lear et al. 2019]. Despite MUD's focus on operations, the standard reinforces security, reducing the device's threat surface when defining that only the described traffic has to be allowed, otherwise dropped.

But from a security point of view, MUD does not completely protect devices due to the only security authority keep on the manufacturer. This becomes a problem if the manufacturer does not care about security or is inexperienced, using vulnerable software components, or hard coding weak access credentials [OWASP 2018]. One example of this is the Mirai botnet, which uses a dictionary of hardcoded Telnet credentials to exploit multiple exposed devices on the Internet [Kolias et al. 2017].

Specifically in domestic environments, another problem is that the end-users are commonly inexperienced with IoT devices operation and configuration. This usually incurs in misconfigurations and opens security breaches on IoT devices [Goutam 2019, Schutijser 2018].

Some works focused on the domestic environment are trying to protect the IoT ecosystem refining MUD rules. In [Schutijser 2018] is proposed an algorithm that traces network communication profile for each connected IoT device, blocking the traffic that is out of the device's common behavior – regardless if the traffic is allowed by MUD. The main issue of this proposal is that a profile can be built while a malicious activity is happening. Another point to take into consideration is that this proposal does not provide the means to share knowledge about malicious activities detected.

In [Al-Shaboti et al. 2018] a security framework for domestic is proposed combining Mandatory Access Control (MAC), a concept similar to MUD, with Discretionary Access Control (DAC), what enables customization of network access control. The proposal, however, only specifies means to the end-user manage DAC, which affects the effectiveness of security measures, since they are potentially inexperienced and can not understand the risks they are exposed to. The authors also suggest outsourcing DAC to a third party, but there are no details of the proposal, neither mentions privacy protection mechanisms to this service.

On the other hand, [Hyun et al. 2017] proposes a data model for security incident signatures focused on IoT networks, which supports the knowledge sharing of IoT malicious activities. The data model concatenates concepts from the NETCONF protocol [Enns et al. 2011] and Suricata[2] intrusion detection signature model to create a simple model focused on mitigating DDoS attacks. For using IP addressing on rules definition, the conversion of these rules to a distinct network is not direct, needing analysis and com-

[2]https://suricata-ids.org/ – accessed in 08/26/2020

parison of the rules between the different networks to the conversion be done, making more difficult the sharing process.

Thereby, there is a lack of solutions that support decision-making on security measures and preserves end-user's privacy, at the same time that enables damage mitigation of recently discovered attacks in multiple distinct networks. Envisioning overcome this lack we propose INXU (Intra Network eXposure analyzer Utility), a security extension of MUD that takes advantage of the mapping of network communications performed by each IoT device made by MUD to detect exposure to well-known vulnerabilities and block its associated traffic.

The core component of INXU is the Malicious Traffic Description (MTD), a document made by a security authority that describes ongoing malicious activities and well-known vulnerabilities, and supports INXU on finding connections of connected IoT device that can expose these attacks. Besides MUD's protection on reducing threat surface, INXU inserts another security that enables protecting against incidents not addressed by the manufacturers on the MUD, or even other malicious activities driven by the manufacturer itself.

The data model of MTD, as well as in MUD, makes use of addressing abstractions to allow describing traffic of malicious activities that can happen on domestic IoT networks without the need of knowing the network's addressing schema or the connected devices. This protects end-users privacy by not exposing private information to third parties on the security measure decision-making process, at the same time that simplifies the knowledge about attacks between distinct networks.

Another important feature is that, due to its architecture, INXU enables a Security Operation Center (SOC) to protect multiple distinct networks by sharing the MTDs. This makes INXU a tool to protect not only the end-users but all the Internet's ecosystem while hardens the operation of botnets and other attacks that affects the Internet's stability.

The rest of this paper is organized as follows. In Section 2 we present the basic concepts. The related works are discussed in Section 3. Section 4 describes the proposal, and the conclusions are in Section 5.

2. Basic Concepts

In this section, we present the basics concepts that are fundamental to the understanding of the work. This section also defines the scope for which INXU was designed.

2.1. Domestic IoT

In general terms, domestic IoT devices are characterized have easy operation and installation. Due to this, the operation of these devices commonly passes through a cloud server maintained by the manufacturer – even if the device and the user are on the same network. Besides simplifying the configuration process, this communication model facilitates the remote operation of the devices, overcoming end-to-end communication issues commonly experienced in domestic networks due to the usage of Network Address Translation (NAT).

Most of the domestic IoT devices implement the TCP/IP stack. On the physical layer, envisioning to take advantage of the pre-existing infrastructure of domestic net-

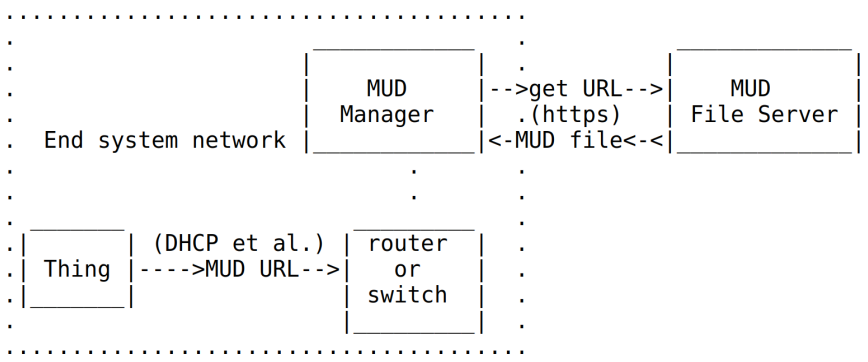

Figure 1. MUD Architecture – by [Lear et al. 2019]

works, the devices commonly use protocols of the IEEE 802.11 (Wi-Fi) and IEEE 802.3 (Ethernet) families. Less common are implementations of IEEE 802.15.4 (Zigbee), but in these cases, the network communication is closed to devices from the same vendor and there are no management options, besides the needing of a gateway to convert the traffic to the most common protocols.

2.2. MUD

The Manufacturer Usage Description is specified by RFC 8520. It is a component-based architecture that enables a device to inform the network manager about the minimal network access configurations the device needs to properly work [Lear et al. 2019]. Despite the focus on support operations, MUD has security benefits when defining that non described communications have to be dropped, reducing the device's threat surface and the possibility of new infections.

MUD's architecture contains 4 components: (i) MUD file, which contains the descriptions of the communications implemented on the device; (ii) MUD file server, that is the server, managed by the manufacturer, responsible for delivering the MUD files; (iii) MUD URL, that is the URL delivered by the IoT device that indicates the location of its respective MUD file; and (iv) MUD manager, that has the function of receiving the MUD URLs on the local network and manage the acquisition, interpretation, and processing of the MUD files.

To describe the communications which the IoT device is enabled to perform, MUD makes use of the Access Control List (ACL) YANG data model defined in [Jethanandani et al. 2019]. An ACL is a list of Access Control Entries (ACEs), that are the atomic entities that describe the traffic performed by a device, specifying the header information of the TCP/IP stack Network and Transport protocols carried on the device's communication.

MUD's basic workflow is illustrated in Figure 1, which starts when an IoT device connects to the network. When connected, the device sends its respective MUD URL to the MUD manager, which requests the MUD file to the MUD file server. Finally, with the MUD files of all connected devices, the MUD manager process them all and generate the network graph, indicating the network configurations that have to be made according to what the IoT devices manufacturers indicated.

3. Related Work

In the context of this paper was considered as related work proposals that in some way improve MUD security for domestic environments or proposals that support the knowledge sharing of IoT related malicious network activities.

Intending to refine MUD rules by defining devices network profiles, [Schutijser 2018] proposes a behavior analysis approach to mitigate DDoS attacks. To do this, the proposal takes into consideration various aspects, such as packet frequency and common sockets. When a behavior deviation is detected, the outlier traffic is blocked, even if allowed by MUD. The proposal, however, only provide mains to block malicious communications with the Internet, not considering other attacks held inside the local network, nor provide means to share knowledge about the discovered malicious activities. Another attention point to take into consideration is that the profile tracing of a device can happen while a malicious activity is happening, which makes the proposal treat one malicious traffic as normal.

[Al-Shaboti et al. 2018] proposes a framework for domestic environments that implements prevention against well-known malicious activities. To do this, the proposal builds the network access configuration based on two types of rules: Manufacturer Access Control (MAC) rules, which role can be assumed by MUD; and Discretionary Access Control (DAC) rules, that can be defined by the end-user or by a trusted third party. With this combination, the framework reduces the threat surface of the IoT devices. However, the proposal does not provide for ways to share knowledge about vulnerabilities with different networks to promote the protection of the entire Internet ecosystem. Also, despite allowing third parties to interact with the definition of security measures, the model does not detail how the integration with external entities should be, nor does it detail the level of information that will be exposed to third parties for due analysis and decision-making, not taking into consideration privacy implications when exposing information from a private network to third parties.

On knowledge sharing, [Hyun et al. 2017] proposes a data model for intrusion detection signatures that supports organizing information about DDoS attacks. The proposed data model concatenate concepts of NETCONF protocol and Suricata intrusion detection signatures to create a YANG data model focused on protecting IoT networks from well-known DDoS attacks. The problem in this proposal is that the conversion of the rules under this model for other networks requires knowing both networks since it does not create abstraction tools for a simple conversion.

4. Proposal

In this section, we present the proposal of INXU: a security extension for MUD to give fast responses to new vulnerabilities in domestic IoT networks. INXU was designed to have as main features: (i) enable fast responses to new vulnerabilities; (ii) enable mitigation of the damages of a new vulnerability, simultaneously in multiple distinct networks; and (iii) enable a decision-making process about security measures on the network edge, avoiding the disclosure of private information to third parties. The lack of solutions that provide means to share information about vulnerabilities, associated with the difficulty of generating generic security countermeasures to multiple distinct networks justifies features (i) and (ii), while the principle of preserving privacy justifies (iii).

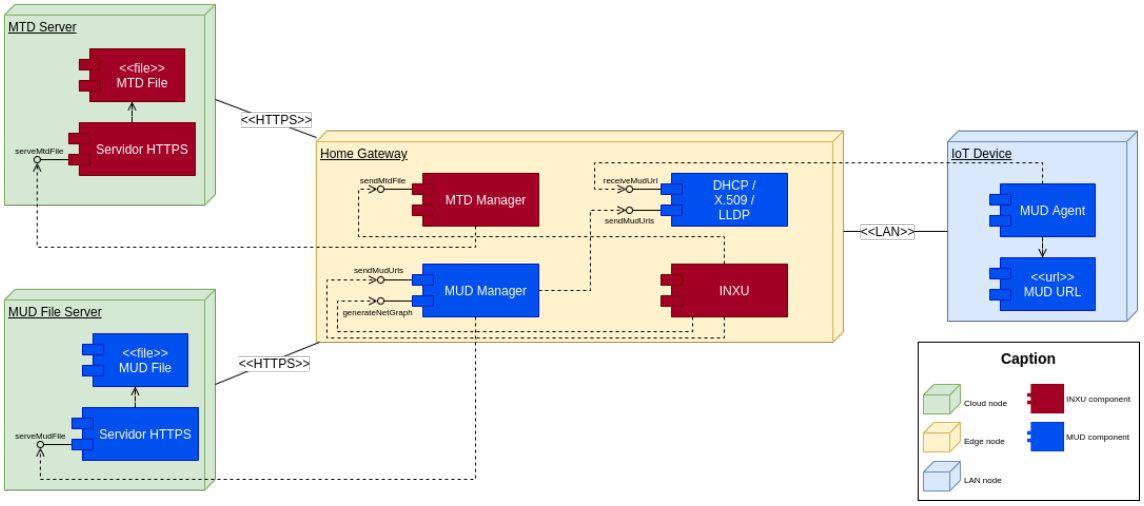

Figure 2. INXU Architecture

In an overview, while MUD builds a network access allow list for each connected IoT device, INXU creates a block list over MUD's allow list to protect the network from malicious activities. To do this, INXU allows a security authority to describe the traffic of ongoing malicious activities in MTDs. With this, the security authorities can use the INXU to prevent multiple distinct networks when releasing new MTDs for every new malicious activity discovered, in a process similar to the antivirus programs vaccines. The domestic network receives the MTDs and process it on the edge, comparing them with the network graph generated by the MUD manager, and then identifies exposures to vulnerabilities, blocking them when discovered.

4.1. Architecture

The architecture of INXU follows the illustrated on the Deployment Diagram in Figure 2, with the components distributed between the following nodes: *MTD Server*, *MUD file server*, *Home Gateway*, and *IoT Device*.

The *Home Gateway* is the main node, placed on the network edge, have the responsibility of collecting MUD related data, which includes receiving the *MUD URLs* and collecting the *MUD files*. It is also responsible for manage MTD related information, such as collecting *MTD file* and processing it to identify exposure to vulnerabilities. The *Home Gateway* contains the following software components: *MTD manager*, *MUD manager* and *INXU*.

The *IoT Device* node, situated on the Local Area Network (LAN), represents the IoT devices connected to the domestic LAN. In the context of INXU, the *IoT Device* is responsible for informing the *MUD URL* of its respective *MUD file* to the *MUD manager* using the extensions to DHCP, X.509, or LLDP created in RFC 8520 to support MUD operation. The *IoT Device* is composed by the software component *MUD Agent* and by the data component *MUD URL*.

The *MUD file server* is placed on the cloud. It is maintained by the IoT device manufacturer and is responsible for responding requests made by the *MUD manager* looking for the *IoT Devices MUD files* using the workflow described in Section 2.2. These

components interact with the *MUD managers*, serving the *MUD files* and assuring the verifiability of its authenticity. The *MUD file server* is composed by the data component *MUD file* and by the software component *HTTPS Server*.

Also placed on the cloud, the *MTD Server* is responsible for storing and delivering the malicious traffic descriptions made by a security authority. This component was designed to enable reliable security authorities to share knowledge about domestic IoT well-known malicious activities, and domestic IoT networks to make use of this knowledge to protect themselves. The *MTD Server* is composed by the software component *HTTPS Server* and by the data component *MTD File*.

4.1.1. Data Components

The following data components are included in INXU architecture: *MUD file*, *MUD URL* and *MTD File*.

As indicated by the name, the *MUD file* is a file that describes the network access necessary for the associated IoT device work properly. This component is under the device's manufacturer responsibility and is represented by a JSON, which describes the network accesses performed by the device using Access Control Lists (ACLs) under the MUD YANG data model defined in RFC 8520.

The *MUD URL* indicates the location of a *MUD File* by a URL in HTTPS schema. This component also classifies the device type, enabling the identification of its manufacturer, model, and sometimes the firmware version. This component was incorporated from RFC 8520.

The *MTD File* is a file where the network traffic associated with malicious activities are described to INXU. This component also contains data for version control, authenticity, and validity time. The content of a *MTD file* is defined by a SOC in a JSON file, following the YANG data model defined on the Section 4.2.

4.1.2. Software Components

The system's architecture has the following software components: *HTTPS Server*, *MUD manager*, *MUD agent*, *MTD Manager*, and *INXU* (Intra-Network eXposure analyzer Utility).

The *HTTPS Server* is fundamental for the functions of the nodes *MUD file server* and *MTD Server*, where this component is responsible for delivering the description files – *MUD file* and *MTD File*. It is also a responsibility of this component to provide means to verifying authenticity, not only by the HTTPS protocol but also by hosting signature files to offline verification.

The *MUD manager* is the software that centralizes the operation of MUD, receiving the connected IoT devices *MUD URLs*, and collecting its respective *MUD files*. This component also interprets the content of *MUD files* to generate the network graph, detailing what are the communications held over the Network and Transport layers of the TCP/IP model. Another function under *MUD manager's* responsibility is the authenticity

verification of the received files. This component was incorporated from RFC 8520.

The *MUD agent* is responsible for sending the IoT device's *MUD URL* to the *MUD manager*. This component is executed every time that the device connects to the network. The communication with *MUD manager* can be over extensions of the protocols DHCP, X.509, and LLDP. This component was incorporated from RFC 8520.

The *MTD Manager* has the function of managing the *MTD File* on the system. It is responsible for requesting the file to the *MTD Server*, verify authenticity, and request a new file when the current file validity expire. The most common way to ensure *MTD file* authenticity is by HTTPS protocol, but the *MTD Manager* can also use the means described in the Section 3.1 of [Rescorla 2000]. On the end of the process, the *MTD Manager* forwards the *MTD File* to *INXU*.

The *INXU* (Intra-Network eXposure analyzer Utility) is the main component of this proposal. It is responsible for verifying all the network communications trying to identify possible exposures to malicious traffic. *INXU* compares the malicious traffic described in *MTD File* with the network graph generated by *MUD manager*. The description of the exposure analysis algorithm will be detailed in Section 4.3.

4.2. Data Model

The data model for describing malicious traffic has to enable defining traffic in a way that distinct networks can interpret and implement security measures, no matter what are the connected IoT devices or network topology. Another important feature to be addressed by the data model is to allow the association between the detected exposure and the malicious activity that exploits it, as well as the grouping of exposures related to the same malicious activity. Since, as far as we know, there is no other data model to address these requirements, we designed the data model described below to the MTD.

The MTD data model uses the ACLs [Jethanandani et al. 2019] under YANG language [Björklund 2016] to describe the malicious traffic, addressing the classification feature. Furthermore, such as in MUD, we defined two network address abstractions to enable defining the traffic in a way that different networks can adapt the description to its context: one abstraction for addresses in the local networks, and the other for using domain names to hosts on the Internet. The data model also includes control fields that support the manageability of the *MTD File*, so the contained data can be categorized in control data and description data.

The tree view of the proposed YANG data model is shown in Figure 3. The specification of the control fields are in the Section 4.2.1, and traffic description field are specified in Section 4.2.2.

4.2.1. Control Fields

There are four control fields: mtd-url, last-update, mtd-signature, and cache-validity. Their function are specified bellow.

- **mtd-url**: this is a required field which stores the URL where the security authority hosts the *MTD File*;

```
module: ufrj-mtd
  +--rw mtd!
     +--rw mtd-url                inet:uri
     +--rw last-update            yang:date-and-time
     +--rw mtd-signature?         inet:uri
     +--rw cache-validity?        uint8
     +--rw attack-descriptions
     |  +--rw to-device-attacks
     |  |  +--rw attack-lists
     |  |     +--rw attack-list* [name]
     |  |        +--rw name                -> /acl:acls/acl/name
     |  |        +--rw specific-devices*   inet:uri
     |  +--rw from-device-attacks
     |     +--rw attack-lists
     |        +--rw attack-list* [name]
     |           +--rw name                -> /acl:acls/acl/name
     |           +--rw specific-devices*   inet:uri
     +--rw malware-descriptions
        +--rw malwares-list* [name]
           +--rw name                   string
           +--rw specific-devices*      inet:uri
           +--rw to-device-attacks
           |  +--rw attack-lists
           |     +--rw attack-list* [name]
           |        +--rw name                -> /acl:acls/acl/name
           |        +--rw specific-devices*   inet:uri
           +--rw from-device-attacks
           |  +--rw attack-lists
           |     +--rw attack-list* [name]
           |        +--rw name                -> /acl:acls/acl/name
           |        +--rw specific-devices*   inet:uri
           +--rw not-attack-traffics
              +--rw to-device-non-attack-traffic* [name]
              |  +--rw name    -> /acl:acls/acl/name
              +--rw from-device-non-attack-traffic* [name]
                 +--rw name    -> /acl:acls/acl/name

  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw mtd
       +--rw local-networks?   empty
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated?   mud:direction
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3/acl:ipv4/acl:ipv4:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3/acl:ipv6/acl:ipv6:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
```

Figure 3. MTD YANG data model tree view

- **mtd-signature**: optional field used to store a URL where can be found the *MTD File* signature file. This is important for offline authenticity verification of the file;

- **last-update**: required field that contains the timestamp information of the *MTD File* generation;

- **cache-validity**: optional integer field that contains the amount of hours to the expiration date of the *MTD File*, counting from the time defined on last-update. This field supports integer values between 1 and 160, and if not defined has to be assumed as 48 hours by the *MTD Manager*.

4.2.2. Traffic Description Fields

The traffic description fields are divided in attack-descriptions and *malware-descriptions* containers. These two categories are needed due to the possibility one malware can deliver multiple different attacks, and can also use other traffic - here called not attack traffic - related to the malware operation.

The specification of each traffic description field is detailed bellow:

- **attack-descriptions**: container that holds all attack traffic incoming to or outgoing from an IoT device:
  - **to-device-attacks**: container that holds all the attack traffic targeting an IoT device on the LAN;
  - **from-device-attacks**: a container that holds all the attack traffic outgoing from an IoT device on the LAN.
- **attacks-list**: list type field to hold all the traffic on the same direction (incoming/outgoing) that is associated to one attack:
  - **name**: required string field with the name of the ACL that describes one attack;
  - **specific-devices**: optional list to specify the *MUD URLs* of the IoT devices that can be affected by the described attack. When this field is filled, *INXU* only considers the devices here listed as targets of these ACLs.

The description of malware traffic allows the aggregation of distinct attacks, and also other not attack traffic that only turn into malicious when related to the malware operation. This aggregation is important for the security measures decision-making process since that sometimes only a traffic combination makes the malware effective or blocking just one type of traffic can almost disable a malware, such as the Mirai's Command and Control traffic [Kolias et al. 2017]. The description of malware traffic follows the above-described nodes:

- **malware-descriptions**: list that holds the traffic description of all malware covered by the *MTD File*:
  - **name**: required string field to uniquely name the described malware.
  - **specific-devices**: optional list to specify the *MUD URLs* of the IoT devices that can be affected by the malware. When this field is filled, *INXU* only considers the devices here listed as affected by this malware;
  - **to-device-attacks**: a container that holds all the malware associate attack traffic targeting an IoT device on the LAN;
  - **from-device-attacks**: a container that holds all the malware associate attack traffic outgoing from an IoT device on the LAN;
  - **not-attack-traffic**: a container that holds not related to attacks, but that turns into malicious when in a malware context:
    - **to-device-not-attack-traffic**: list with all the ACLs that describes malware associate not attack traffic targeting an IoT device on the LAN;
    - **from-device-not-attack-traffic**: list with all the ACLs that describes malware associate not attack traffic outgoing from an IoT device on the LAN.

There are also augments on the ACL model. These augments are responsible for creating the abstraction for the traffic descriptions, enabling the portability of the knowledge to the different networks. The augments are described as follow:

- **mtd:local-networks**: optional leaf that, when present, represent that the current ACE applies to any device on the local IP networks;
- **direction-initiated**: optional field incorporated from MUD to specify the TCP connection starter;
- **src-dnsname and dst-dnsname**: optional field to enable the usage of DNS domain names to specify the remote host instead of using IPv4 or IPv6 addresses.

4.3. Exposure Analysis Algorithm

The exposure analysis algorithm of INXU use malicious traffic descriptions from *MTD file* to compare with the network communication graph generated by *MUD manager*, and tries to detect vulnerabilities on the network. In this context, one vulnerability is detected when some graph edge matches with any described malicious traffic.

Based on the *MUD files*, each host expected to communicate with the IoT devices, or the IoT devices itself, are represented by nodes on the network communication graph generated by *MUD manager*. The nodes are represented by the host network address, and in the case of IoT devices on the LAN, the *MUD URL* is associated with the node information too. The graph edges represent TCP or UDP sockets, or ICMP communications, where each communication path is represented by a directed edge. Protocols related to the 6lowpan stack are out of the scope of this proposal since these protocols are uncommon in the domestic IoT environment and MUD only accepts TCP/IP stack protocols in the ACLs.

Algorithm 1 presents the comparison logic for edges and ACEs. In lines 4 and 5, the algorithm iterates, respectively, over the graph edges and the *MTD File* ACEs to compare all the edge-ACE couples.

Line 6 verifies if the ACE is for specific devices. The comparison only continues if there are no specific devices listed, or if one of the nodes involved on the edge is present on the list of specific devices. In the negative case for both, the algorithm goes to the next iteration.

Lines 7 and 8 compare, respectively, the source and destination of the couple using the function *compare-addresses*. The function in both lines returns FULL-MATCH if the addresses are equals or if the ACE is an abstraction for local address and the node address is local, otherwise it returns NOT-MATCH.

Line 9 compares the protocols over IP using the function *compare-transport*. The function returns (i) FULL-MATCH if protocols are equals, or if the ACE is applicable for any protocol; (ii) PARTIAL-MATCH case ACE protocol is specific and edge allows any protocol; or (iii) NOT-MATCH if protocols are specific and different.

Lines 10 and 11 compare, respectively, the source and destination ports using the function *compare-ports*. The function returns (i) FULL-MATCH if ports are equals, or if the ACE is applicable for any port; (ii) PARTIAL-MATCH if ACE protocol is specific and edge allows any port; or (iii) NOT-MATCH if ports are specific and different.

Algorithm 1: Exposure Analysis Algorithm

Input: Network Graph as net-graph
Input: MTD File ACEs as mtds
Output: List of vulnerable edges, respective malicious ACEs and classifications

```
1  FULL-MATCH = 0x3;
2  PARTIAL-MATCH = 0x1;
3  NOT-MATCH = 0x0;
4  foreach edge in net-graph do
5      foreach ace in mtds do
6          if ace.specific-devices is empty OR edge.source.mud-url in
             ace.specific-devices OR edge.destination.mud-url in
             ace.specific-devices then
7              src = compare-addresses(ace.source, edge.source);
8              dst = compare-addresses(ace.destination, edge.destination);
9              transport = compare-transport(ace.transport, edge.transport);
10             t-src = compare-ports(ace.source-port, edge.source-port);
11             t-dst = compare-ports(ace.destination-port, edge.destination-port);
12             classification = src & dst & transport & t-src & t-dst;
13             if classification is FULL-MATCH then
14                 block-edge(edge.src, edge.dst, edge.transport,
                     edge.source-port, edge.destination-port);
15             else
16                 if classification is PARTIAL-MATCH then
17                     block-edge(edge.src, edge.dst, ace.transport,
                         ace.source-port, ace.destination-port);
18                 end
19             end
20         end
21     end
22 end
```

Line 12 performs a logic AND between the results of lines 7 to 11 to verify any level of exposure to vulnerability.

Possible network blocks are determined between lines 13 and 19, considering the result of line 12. In the case of FULL-MATCH, the communication of the edge is completely blocked. If the result was PARTIAL-MATCH, a block is made over the protocol specified on the ACE. Case NOT-MATCH, the edge is not exposed to any vulnerability and the algorithm can go to the next iteration.

5. Conclusion

In this paper, we presented INXU, a security extension for MUD to support giving fast responses to new vulnerabilities on domestic IoT networks. INXU compares the network flows generated by MUD with the MTD described by a security authority to identify flows that expose a local IoT device to one malicious activity. In this way, INXU supports protecting not only the domestic IoT device end-users, but all the Internet ecosystem since that it enables security authorities to protect a large number of heterogeneous networks from being targets of attacks, and even preventing the same networks from being the source of the attacks as members of botnets.

INXU is still a work in progress, and some improvements are on our road map. One important upcoming next step is implementing an INXU prototype and run some experiments over domestic networks under attacks or hosting botnet activities, using this to measure the gain in security protection when compared with MUD.

One important current limitation of INXU is that it still does not consider usability on the decision process for blocking a flow. The current model blocks any exposure to vulnerability, even if it does not represent a real risk, so blocking these traffic reduces the device's usability and do not provide real protection. To overcome this, we intend to provide means to the security authority building a *MTD File* define conditions for blocking one traffic. Two are the possible approaches to be taken: (i) define a risk threshold for each attack or malware, and associate a risk level for each ACL; or (ii) defining critical ACL sets for each attack or malware, blocking the flows only when the found exposures match with a critical ACL set.

References

Al-Shaboti, M., Welch, I., Chen, A., and Mahmood, M. A. (2018). Towards secure smart home iot: Manufacturer and user network access control framework. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 892–899.

Björklund, M. (2016). The YANG 1.1 Data Modeling Language. RFC 7950.

Enns, R., Björklund, M., Bierman, A., and Schönwälder, J. (2011). Network Configuration Protocol (NETCONF). RFC 6241.

Goutam, S. (2019). Hestia: Simple least privilege network policies for smart homes. Master's thesis, North Carolina State University.

Habibi Gharakheili, H., Sivanathan, A., Hamza, A., and Sivaraman, V. (2019). Network-level security for the internet of things: Opportunities and challenges. *Computer*, 52(8):58–62.

Hyun, D., Kim, J., Hong, D., and Jeong, J. P. (2017). Sdn-based network security functions for effective ddos attack mitigation. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 834–839.

Jethanandani, M., Agarwal, S., Huang, L., and Blair, D. (2019). YANG Data Model for Network Access Control Lists (ACLs). RFC 8519.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84.

Kramp, T., Van Kranenburg, R., and Lange, S. (2013). Introduction to the internet of things. In *Enabling Things to Talk*, pages 1–10. Springer, Berlin, Heidelberg, Berlin, Heidelberg.

Lear, E., Droms, R., and Romascanu, D. (2019). Manufacturer Usage Description Specification. RFC 8520.

Lin, H. and Bergmann, N. (2016). IoT Privacy and Security Challenges for Smart Home Environments. *Information*, 7(3):44.

Marzano, A., Alexander, D., Fazzion, E., Fonseca, O., Cunha, I., Hoepers, C., Steding-Jessen, K., Chaves, M. H. P. C., Guedes, D., and Jr., W. M. (2018). Monitoramento e caracterização de botnets bashlite em dispositivos iot. In *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Porto Alegre, RS, Brasil. SBC.

OWASP (2018). Owasp top 10 internet of things 2018. `https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project`. Acesso em 10/01/2020.

Pires, V. R., Coutinho, F. R., Menasché, D. S., and de Farias, C. M. (2019). Gatos virtuais: detectando e avaliando os impactos da mineração de criptomoedas em infraestrutura pública. In *Anais do XIX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. SBC.

Rescorla, E. (2000). HTTP Over TLS. RFC 2818.

Schutijser, C. (2018). Towards automated ddos abuse protection using mud device profiles. Master's thesis, University of Twente.

van der Meulen, R. (2017). Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. Disponível em `http://www.gartner.com/newsroom/id/3598917` (06/07/2019).