Systematic IoT Penetration Testing: Alexa Case Study*

Massimiliano Rak, Giovanni Salzillo,, and Claudia Romeo

University of Campania Luigi Vanvitelli, Department of Engineering, via Roma 29, Aversa (CE), Italy
{massimiliano.rak, giovanni.salzillo}@unicampania.it

Abstract

The Internet of Things paradigm arises many issues in terms of privacy and security. Systems that are commonly configured by personnel with limited experience manage incredible amount of personal data and have direct control over home systems (e.g. controlling home lights or home heating system). The purpose of our research is to define a methodology that automates as much as possible the penetration testing actions, in order to help a tester with limited security skills to find possible attacks and demonstrate them clearly to the home user. The core idea is that we rely on an existing automated threat modeling technique in order to build up the possible attacks to the system under test. The threats are concrete and understandable even to a non-expert, like home users, and help them at identifying real risks and possible countermeasures. The paper will demonstrate the proposed approach over a very typical use case, a smart home controlled through the Alexa Voice Assistant, demonstrating how it is possible to find a working attack on such a system, using very cheap dedicated hardware and with common tools.

1 Introduction

Internet of Things (IoT) is the new paradigm that relies on the idea of widely distributed interconnected objects that constantly cooperate and automate many of the common actions of our life. Vocal assistant and their smart home applications are one of the most clear examples of the paradigms: home users are able to control lights and setting alarms using voice commands in natural language. However, such a paradigm has a side effect in terms of privacy and security: all data are continuously shared through the local and public networks (e.g. vocal command sent to the voice service for processing, commands sent to devices and so on).

Home systems are configured and installed by the final customers (i.e. the home user), typically with limited computer science skills and without any competences in terms of computer security. However, even if a dedicated professional could be involved to configure the house network, the effort and cost needed to perform a valid security assurance procedure are too high to be considered in such a context.

One of the possible approach to address such type of issues is to find (semi-) automated techniques for penetration testing that helps at identifying the possible attacks and, consequently, apply the needed countermeasures. Examples of such approaches can be found in [1, 2, 3, 4].

The purpose of our research, which extends the results in [4] was to define a methodology that can be applied in an ordinary context, like our home or even our office, in order to find feasible attacks and demonstrate them clearly to the home user. The core idea is that we rely on threat modeling (which we automated in [5]) in order to identify threats that are near to home user perception, providing at the same time a way to test the threat in an almost automated way. It is worth noticing that the process is still not completely automated, but

*Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

aims at simplifying a process (the penetration testing) which is commonly adopted only by (costly and rare) security experts with long time frames.

This paper demonstrates the proposed approach over a very typical use case, a smart home controlled through the Amazon Echo Plus hub and the Alexa Voice Assistant, demonstrating how it is possible to find a working attack on such a system, using very cheap dedicated hardware and with common tools. Next Section 2 summarizes the existing penetration testing techniques, outlining the steps involved and the needed competences, while section 3 describes the methodology we propose and its main features. Section 4 discusses our case study and demonstrates a successful attack on to an Alexa-based home system. Finally, section 5 summarizes the conclusions and proposes future works.

2 Related Work

Penetration testing is mostly an expert-guided activity: despite the needs, as outlined in [6], at state of art, does not exist any standard devoted to describe penetration testing activities. In the following, we briefly summarize the most known penetration testing methodologies and security assurance techniques.

The **NIST**[1] **SP-800-115**'s special publication [7] describes several security testing measures and provides guidelines for organizations on planning and conducting different kind of security assessments. Among all, it defines a penetration testing methodology model based on the following four phases, repeated iteratively: *Planning*, *Discovery*, *Attack*, *Reporting*. Security objectives and penetration testing goals are established during the Planning phase. The Discovery phase covers information gathering, scanning and vulnerability analysis. The Attack phase is the core activity in which the previously identified potential vulnerabilities are verified and validated. The Reporting phase occurs simultaneously with the other three phases of the penetration test.

The non-profit OWASP[2] foundation, recently released the **OWASP Testing Guide** v4 [8] as part of the OWASP main project [9], an open document that describes several core testing techniques of web applications security testing. It defines a penetration testing model built on two main phases. In phase 1 (*Passive Mode*) the tester tries to understand the application's logic, then during phase 2 (*Active Mode*) application security is actually tested. The second phase can be further divided into the following sub-categories: *Information Gathering*, *Configuration and Deployment Management Testing*, *Identity Management*, *Authentication and Authorization Testing*, *Session Management Testing*, *Input Validation and Error Handling*, *Cryptography*, *Business Logic and Client Side Testing*.

The Penetration Testing Execution Standard (**PTES**) is an open document that has been designed to provide a common language between the businesses world and security service providers in order to perform penetration testing activities. It is divided into seven main generic sections: *Pre-engagement Interactions*, *Intelligence Gathering*, *Threat Modeling*, *Vulnerability Analysis*, *Exploitation*, *Post Exploitation*, *Reporting*. Since the standard does not provide any technical guidelines to execute an actual penetration test, a technical guide have been created to support the standard. [10]

The Open Source Security Testing Methodology Manual (**OSSTMM**) [11] is another open-source security evaluation methodology introduced by the Institute for Security and Open Methodologies. This methodology allows to audit operational security of physical locations, human interactions, systems and network communications, aiming at quantitatively assess the

[1]National Institute of Standards and Technology
[2]Open Web Application Security Project

attack surface and the deployed security measures. OSSTMM does not provide a list of tools to actually test every technology domain, but defines what needs to be tested and what to do before, during and after a security test, also behaving as supporting reference in several security certification processes. OSSTMM's workflow is divided into four phases: (i) Induction, (ii) Interaction, (iii) Inquest, (iv) Intervention.

Type of tests and security properties to be tested are defined during the induction phase. The interactions phase lets the penetration tester determine and select the target systems. During the inquest phase, the penetration tester retrieves as much information as possible about the targeted assets. Only in the last phase, security properties and measures are actually assessed. A reporting phase must follow after every test.

The Penetration Testing Framework (**PTS**)[12] is a very detailed hands-on testing guide to protect several assets and security attributes. It does not clearly specify a model to be employed in a penetration testing process, instead it describes several techniques to conduct practical and targeted security assessments. It covers multiple areas (e.g. password cracking, VoIP Security, wireless penetration) and targets, including vendor-specific products (e.g. Cisco, Citrix and AS/400).

The Information Systems Security Assessment Framework (**ISSAF**) [13], although no longer maintained, is another security framework designed to evaluate network, system and application security controls. The framework defines three main phases: (i) Planning and Preparation, (ii) Assessment, (iii) Reporting, Clean-up and Destroy Artefacts.

The first phase sets the security objectives to assess and plans the security tests which must be conducted during the second phase. In turn, the second phase can be further divided into the following operational sub-phases: *Information gathering*, *Network mapping*, *Vulnerability identification*, *Penetration*, *Gaining access and privilege escalation*, *Enumerating further*, *Compromise remote users/sites*, *Maintaining access*, *Covering tracks*. The third phase covers the reporting process and removes any artefacts leftover from the actual penetration testing stage.

The Metasploit Framework (**MSF**) [14] is one of the most used penetration testing framework. Its modular structure lets to easily develop and execute exploits against a remote target machine. Though it does not clearly define a penetration testing methodology, MFS assists testers into many penetration testing phases, thanks to its auxiliary modules (e.g. used for scanning and vulnerability testing) and exploitation and post-exploitation tools.

The above analysis outlines that all existing penetration testing methodologies rely on the security expert experience, offering guidelines over an experience-based activity. Techniques like OSSTMM, which aims at supporting certification processes, introduce quantitative evaluation on the processes, while techniques like OWASP and PTES are more oriented to focus on identifying original attack processes. However, only PTES includes a threat modeling phase. The above-illustrated approaches focus on discovering technical vulnerabilities, instead of relating possible attacks to high-level threats understandable to the end user. The approach we propose, on the contrary, starts from the end-user perception of the risks, aiming at finding a possible attack whose success affects the clear interests of the system customer.

3 The proposed methodology

It is provable that all the existing security testing methodologies and, accordingly, all the existing penetration testing techniques cannot grant full completeness and non-redundancy attributes against every possible security threats. Completeness and non-redundancy properties are two qualitative indicators which evaluate the goodness of a security testing methodology. A security testing methodology is said to be complete if it considers all the feasible security threats

for a given **SuA** (System under Attack). It's non-redundant if it does not count not-applicable security threats.
Since none of the existing penetration testing methodologies is nor complete and non-redundant, in recent years considerable efforts have been made to try to obtain more reliable models and techniques.
The methodology proposed in this work evolves the automated penetration testing model presented in [4], inheriting several concepts and common definitions. This new model has been designed to be almost entirely guided by the threat modeling and risk assessment processes in the attempt to maximize both completeness and non-redundancy attributes, as well as the overall penetration testing quality process.

This approach enables less-skilled penetration testers to perform a full threat-modeling-driven penetration testing security evaluation, guiding them to look for system vulnerabilities on a per threats-basis. Still, as will be clear further on this chapter, it is needed a suitable way to identify all the threats that are applicable to the **SuT** (System under Test). Our methodology relies in the following four phases, described in details in next sections : (i) **System Modeling** (semi-)formal description of the system under test, (ii) **Threat Modeling** identification of the threats, (iii) **Planning** planning the tests and possible attacks to perform and (iv) **Penetration Testing**: actual execution of the attacks.

3.1 System Modeling

The proposed methodology entirely relies on the correctness and the accuracy of the SuT model, which must be built during phase 1 and will be used to drive the following activities. In a white-box penetration testing approach scenario, penetration testers have access to the whole system description and information, whereas in a black-box approach he/she must retrieve those kind of information by means of suitable scanning tools. In the intermediate grey-box scenario, penetration testers have access only to a restricted set of information, which must be enriched by the same information-gathering process as in the previous black-box approach.
Then, is necessary to put the collected information in a suitable representation form, so that is possible to easily visualize the whole system architecture and the dependencies between different components. According to our approach, the SuT model is described by the Multi-cloud Application Composition Model (MACM) formalism, a graph-based system modeling formalism initially introduced to describe architectural components and security properties of cloud-based applications [15]. In MACM, system components are modeled with graph nodes, whereas relationships between system components are represented with directed links between nodes. The proposed formalism allows to properly represent system architecture components and also enable to annotate security-concerning information in a human and machine-readable format. It can be easily extended to describe new technology domains, as in [5] for IoT systems.

3.2 Threat Modeling

Based on the enriched system information available through the MACM representation, penetration testers must identify the threats each component is potentially subjected to and build a complete system Threat Model, used as a basis for actual penetration testing. However, threat enumeration and identification is one of the most challenging tasks and may result very tough or heavily time-consuming for the less experienced penetration tester. We use the same threat catalogue-based approach introduced in [5], which really simplifies and speeds up the threat identification process and, consequently, the threat model construction. The threat catalogue is a knowledge base initially developed in the context of the two European projects SPECS and

MUSA, which collects several well-known threats grouped, among several fields, per affected asset and technology domain attributes. It includes threats for many software components and protocols (Ethernet, IP, TCP, TLS, XMPP, OAUTH, Zigbee, Bluetooth, GSM) and it is currently maintained and constantly updated. The catalogue is constructed in such a way that MACM nodes coincide to threat asset-type. Each threat in the catalogue is associated with a STRIDE category [16] and the CIA property (Confidentiality, Integrity, Availability) it undermines. Given the comprehensive-assumed threat catalogue, the threat model construction occurs in different steps. In the first sub-step threats are searched and identified by asset-type: starting from the MACM representation, the threat catalogue is queried multiple times per each node of the graph, collecting aside all the resulting threats and, simultaneously, all the relevant threat agents. Subsequently, only the actually applicable threats are filtered out from the previously obtained list and the retrieved threats, among with the associated threat agents, are finally grouped together in the resulting system's threat model.
Because the threat model built in this way could be very broad and not all the threats discovered could be real threats to the SuT, threat model entries could be further quantitatively classified through a risk assessment stage. This process aims to assign a risk value to each threat discovered, taking into account likelihood and impact of a successful exploitation. Thus, threats could be sorted by risk value and tests could be planned later on by threat severity.

3.3 Planning

In the planning phase, penetration testers select the right test planning schemes from a pre-build knowledge base. This knowledge base is continuously updated with exploitation techniques, described in terms of tools and actions to execute, associated with specific threats. Each exploitation technique can be used to perform actual attacks. Based on the threat model obtained in the previous phases, the penetration testing planning step is substantially divided into two sub-phases: the selection of the threatened assets and the following threat testing actions planning.
It's worth noting that threats and assets test prioritization could be partially or fully driven by the risk value obtained in the risk assessment stage. In practice, we built up a planning knowledge-base in the form of a relational DB, that contains:

- asset- and threat- specific attacks, described in terms of the tools, parameters and actions to be performed to execute the attack;
- a mapping among known vulnerabilities and known attacks to asset-specific threats.

It's worth noting that insertion of attack information into the DB is a manually-conducted task at the moment. In fact, we continuously enrich the knowledge-base by manually looking at multiple and often not normalized public sources. Though, we expect to automate this activity in the next future. The DB containing the knowledge base is still work in progress and will plan to release it publicly in future

Once selected the targeted component and the security objectives to be tested, penetration testers look for available vulnerabilities and, eventually, set up the appropriate tools or framework to exploit and put into effect the chosen threat.

The output of this phase is the logical chaining of all the steps, the commands and the environment configurations required to operate the planned attacks. Table 1 describes the field attributes of the penetration testing planning knowledge base.

Table 1: Penetration Testing Planning Field Description

Field	Description
Asset	The asset under analysis. This field is mapped to MACM nodes.
Asset Type	Assets are grouped into class by common attributes and referred to by asset type.
Threat	The threat associated to the current asset.
Description	A briefly description of the threat.
Hardware	The hardware needed for the attack implementation.
Tools & Frameworks	The software layer needed for the attack implementation.
Actions	High level description of the operations needed to perform the attack.
Associated commands	The actual execution chaining command list.
Notes	Attack annotations.
Compromised asset	Assets that are going to be consequently compromised.

4 A Case Study

In order to demonstrate the proposed methodology, we focused our attention on home assistant solutions. Among the various proposals on the market, we chose Echo Plus, the Amazon assistant based on Alexa, which make use of cloud-based voice services to perform actions like answering basic questions or controlling home automation devices. Echo Plus has a built-in hub that supports and controls ZigBee smart devices, such as light bulbs and door locks, which can be bound to the home assistant asking Alexa to "discover the devices".
To apply our testing methodology we considered a test-bed composed of three additional devices beside the Echo Plus: a Philips Hue White and a Philips Hue Motion Sensor, respectively a smart light bulb and a smart motion sensor, and Osram Lightify, a smart light bulb equipped with the color change function. The testing environment also includes a penetration testing notebook located in the proximity of the home system network, configured with the Kali Linux distribution and equipped with a USB interfacing dongle for ZigBee networks.

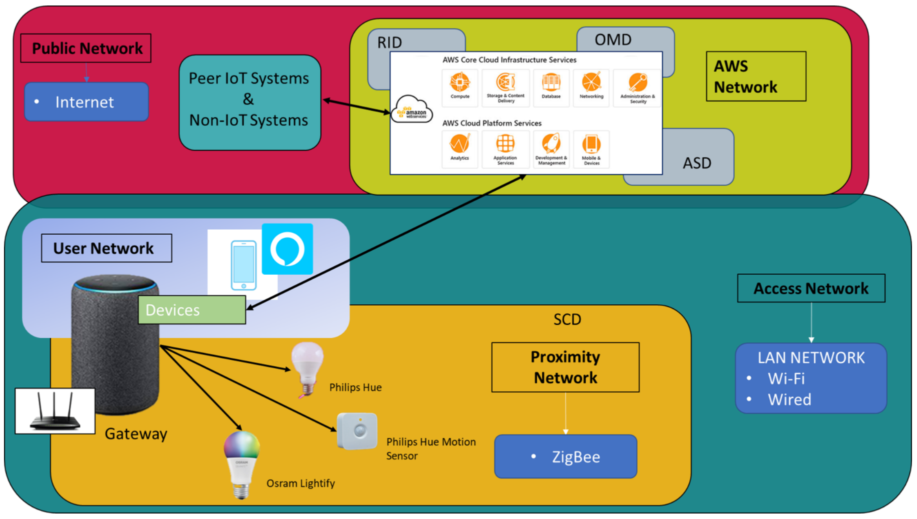


Figure 1: System Model of an Alexa-Based home system

The communication view ([17]) in Fig. 1, summarizes the main elements of the system and their connections. In the picture, the *Access Network*, i.e. the customers LAN home-network,

enable Echo Plus to access the Internet (*Public Network*) and the AWS Services. Users may have additional devices (like mobile phone) that through a *User Network*, are able to access to the Echo Plus, which then communicates with devices through the *Proximity Network* (i.e. ZigBee). It's worth noticing that Echo Plus supports ZigBee Home Automation 1.2 (HA1.2) standard, since the whole attack described in the following section is entirely based on a vulnerability of this protocol.
Remind that the goal of the proposed methodology is to perform a penetration testing assessment by searching the SuT asset types from a threat catalogue knowledge base and obtain a list of the all associated threats together with the operations to be carried out to implement the actual attacks.

4.1 System Model

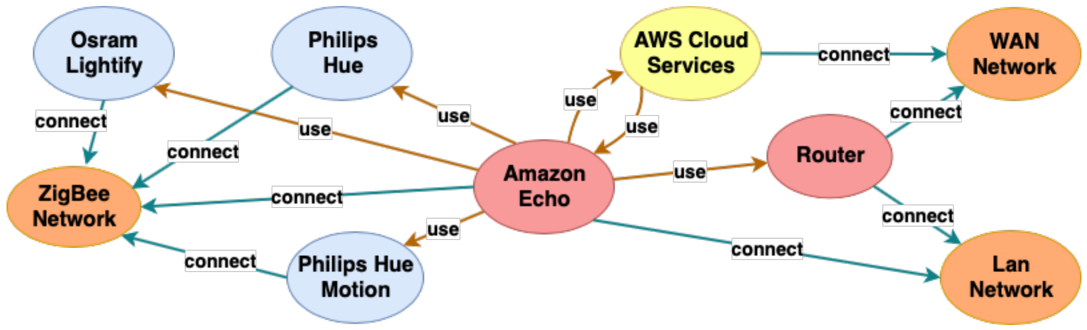


Figure 2: MACM System Model

As already introduced in 3, we use the MACM representation to model the SuA architecture. To correctly model our specific testbed we made use of four types of nodes (*IoTDevice*, *IoTGateway*, *Network*, *Service*) and two kinds of relation (*use*, *connect*), all already defined in [15, 5]. In Fig. 2 both system components and relations are represented through the MACM formalism. In more detail, blue nodes represent our devices: the *Philips Hue*, the *Osram Lightify* and the *Philips Hue Motion Sensor*. These are all connected to the ZigBee network through the *connect* relationship. Networks are modeled by orange nodes and, beyond the ZigBee network, note the presence of two other network types: a LAN network and a WAN network. This two networks are eventually connected by a router, an *IoTGateway*, and represented in figure with a red node. The Amazon Echo Plus, an *IoTGateway* node as well, acts as a gateway between the ZigBee network and the LAN network, as outlined by the *connect* relationship. On the other hand, the *use* relationships link Echo Plus to the controlled smart devices available in the home system network and connect the hub to the Wi-Fi network passing through the router. Finally, Echo Plus requires a connection to the AWS Cloud Service, which is then modeled with the yellow node. A connect relationship links this node to the WAN network node to indicate that these services pass through Internet.

4.2 Threat Modeling

The goal of the Threat Modeling process is to list all the possible threats for the SuT and, according to the technique described in [5], we retrieved them in an automated way through ad-hoc queries on a threat catalogue. In order to support the technologies involved within this work, we enriched the catalogue with ZigBee-related known threats. The whole resulting Threat Model is quite long and, for brevity's sake, we reported in Table 2 only two of the threats related to the ZigBee network assets. Note that threats are classified by *Asset Type*, in this case *Network* and its specialization *ZigBee Network*.

Table 2: The Threat Catalogue (selection of rows referred to the ZigBee Protocol)

Asset	**Asset Type**	**Threat**	**Description**	**STRIDE**	**CIA**	**Threat Agents**
ZigBee Network	Network	Selective Forwarding	An attacker can forward a packets that traverse a malicious node depending on some criteria	Spoofing, Information Disclosure	Integrity, Confidentiality	Network Intruder
ZigBee Network	ZigBee Network, Network	Network Key Discovery	The intruder add a physical sniffer and/or reconfigure the network to read all data, including Network key.	Information Disclosure	Confidentiality	Network intruder

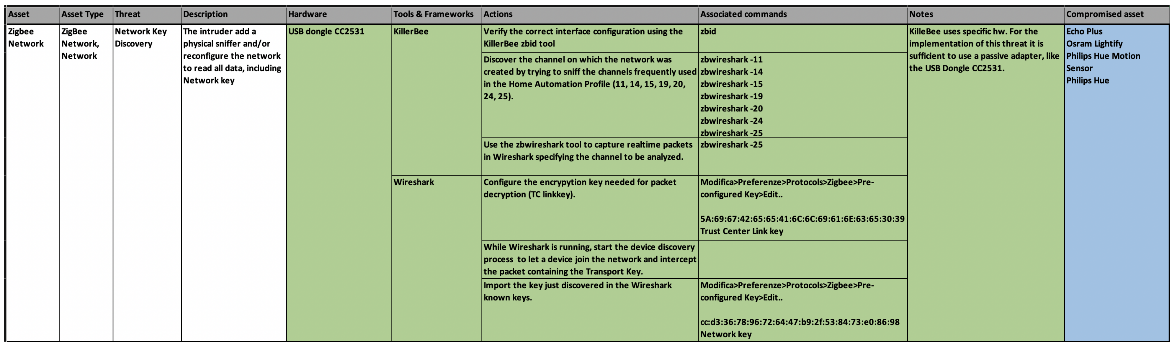
A ZigBee network is subject to the same typical threats of generic networks, such as eavesdropping and message replay. In addition, a specific threat of the ZigBee networks is shown in the last row of the table, the "Network Key Discovery" threat, which is the one we chose for the implementation phase.

4.3 Planning Penetration Testing

Planning phase produces a detailed test plan to check the feasibility of each threat. The threat we decided to test into the planning phase is, as already aforementioned, the ZigBee related "Network Key Discovery" threat. Basically, the threat represents the intruder's capability to sniff or inject packets into a ZigBee network by sniffing the private network key that must be shared during the join process and later used to secure the communication on the radio channel. Fig. 3 illustrates the implementation plan trying to exploit the vulnerability that would put into effect this threat and that relies onto the automation profile of the ZigBee protocol. The table follows the same structure introduced in 3 and clearly displays the steps and the preconditions needed to perform the actual attack against the ZigBee protocol, having as result the compromise of all the controlled smart devices. It's worth noticing that the attack described later on in the selected plan relies on a quite simple vulnerability. All the ZigBee Home Automation 1.2 devices, including Amazon Echo Plus, must implement a set of security attributes, namely Startup Attribute Sets (SAS). It turned out that Echo Plus utilizes the Default TC Link Key "ZigBeeAlliance09" to set up the same Network Key for every smart device, used in turn to encrypt the traffic flowing among the ZigBee network. As consequence, it's sufficient to intercept the initial handshake between the Echo Plus and one of the smart devices in order to get the Network Key and be able to read or inject commands to all other controlled devices. In the next subsection, we analyze more in detail the steps needed to the test and carry out the actual attack.

4.4 Penetration Testing Implementation: Successful attack description

Looking at the prerequisite listed in Fig. 3, namely the *Hardware* and *Tools & Frameworks* columns, we used a CC2531 USB dongle to physically interface the penetration testing laptop to the ZigBee network, in addition to both KillerBee and Wireshark as software layer to capture and analyze ZigBee packets. The *Associated commands* column lists, per each tool, the commands that must be used.



Asset	Asset Type	Threat	Description	Hardware	Tools & Frameworks	Actions	Associated commands	Notes	Compromised asset
Zigbee Network	ZigBee Network, Network	Network Key Discovery	The intruder add a physical sniffer and/or reconfigure the network to read all data, including Network key	USB dongle CC2531	KillerBee	Verify the correct interface configuration using the KillerBee zbid tool	zbid	KilleBee uses specific hw. For the implementation of this threat it is sufficient to use a passive adapter, like the USB Dongle CC2531.	Echo Plus Osram Lightify Philips Hue Motion Sensor Philips Hue
						Discover the channel on which the network was created by trying to sniff the channels frequently used in the Home Automation Profile (11, 14, 15, 19, 20, 24, 25).	zbwireshark -11 zbwireshark -14 zbwireshark -15 zbwireshark -19 zbwireshark -20 zbwireshark -24 zbwireshark -25		
						Use the zbwireshark tool to capture realtime packets in Wireshark specifying the channel to be analyzed.	zbwireshark -25		
					Wireshark	Configure the encrypytion key needed for packet decryption (TC linkkey).	Modifica>Preferenze>Protocols>Zigbee>Pre-configured Key>Edit.. 5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39 Trust Center Link key		
						While Wireshark is running, start the device discovery process to let a device join the network and intercept the packet containing the Transport Key.			
						Import the key just discovered in the Wireshark known keys.	Modifica>Preferenze>Protocols>Zigbee>Pre-configured Key>Edit.. cc:d3:36:78:96:72:64:47:b9:2f:53:84:73:e0:86:98 Network key		

Figure 3: An example of the Penetration Test Plan

Initially, it is necessary to insert the CC2531 in a USB port and check, through a terminal, the correct interface configuration with the *zbid* tool.

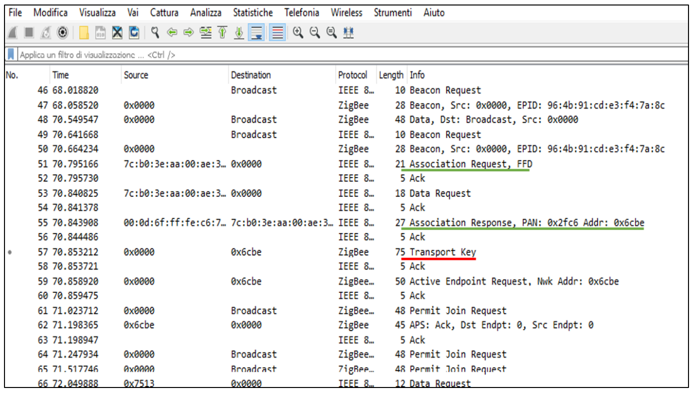
Successively, it is necessary to retrieve the correct channel on which the ZigBee network is currently operating, namely the Home Automation profiles preferred channels (one among 11, 14, 15, 19, 20, 24, 25). This can be achieved with *zbwireshark*, by testing each channel number and looking for traffic. In our testing environment, we found that the channel 25 was used by Echo Plus for setting up the ZigBee network.
To retrieve the Network Key it's necessary to configure the Default TC Link Key into Wireshark settings. As previously outlined, this is the default key defined into the ZigBee Home Automation 1.2 (HA1.2) standard.

Once set the Default TC Link Key, it's necessary to induce Echo Plus to start an association towards a smart device. We manually triggered Echo Plus through Alexa, asking it to search for the new devices and set the device into the association mode. The smart device joins the ZigBee network by following the procedure illustrated in Fig. 5. Fig. 4 shows the captured packets flow between Echo Plus and the Osram Lightify light bulb. The Amazon Echo Plus and the light bulb perform beacons exchange to recognize each other, as outlined by packet frames 46 to 50 in Fig. 4. At a certain point, the light bulb (with MAC address 7c:b0:3e:aa:00:ae:3d:20) sends to Echo Plus (with network address 0x0000) an association request. Fig. 6 shows the contents of the association frame.

There is no additional security measure enabled, except the Default Trust Center Key. The coordinator in response to an association request assigns a network address to the new associated device (short address). Lastly, the coordinator sends the Network Key to the new node as in *frame 57*, Transport Key. Fig. 7 shows the frame 57 contents and in particular the known Default TC Link key and the new Network Key. At this point, using the Network key and importing it into Wireshark is possible to decrypt all the packets subsequently exchanged between the nodes, e.g. intercepting light on/off command or when a different color is set.

5 Conclusions and Future work

We demonstrated that using very limited resources (a notebook and a USB dongle that cost less than 20 euros), it is possible to systematically test an home-based system, identifying and executing a successful attack against a common voice assistant-based system. Our analysis outlines the high level of risk of products in commerce, that need to address a trade-off among usability and security of the system: the device discovering systems adopted by the vocal assis-


File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonia Wireless Strumenti Aiuto
Applica un filtro di visualizzazione ... <Ctrl />

```
No.   Time       Source               Destination          Protocol  Length Info
   46 68.018820                       Broadcast            IEEE 8…   10 Beacon Request
   47 68.058520  0x0000                                    ZigBee    28 Beacon, Src: 0x0000, EPID: 96:4b:91:cd:e3:f4:7a:8c
   48 70.549547  0x0000               Broadcast            ZigBee    48 Data, Dst: Broadcast, Src: 0x0000
   49 70.641668                       Broadcast            IEEE 8…   10 Beacon Request
   50 70.664234  0x0000                                    ZigBee    28 Beacon, Src: 0x0000, EPID: 96:4b:91:cd:e3:f4:7a:8c
   51 70.795166  7c:b0:3e:aa:00:ae:3… 0x0000               IEEE 8…   21 Association Request, FFD
   52 70.795730                                            IEEE 8…    5 Ack
   53 70.840825  7c:b0:3e:aa:00:ae:3… 0x0000               IEEE 8…   18 Data Request
   54 70.841378                                            IEEE 8…    5 Ack
   55 70.843908  00:0d:6f:ff:fe:c6:7… 7c:b0:3e:aa:00:ae:3… IEEE 8…   27 Association Response, PAN: 0x2fc6 Addr: 0x6cbe
   56 70.844486                                            IEEE 8…    5 Ack
•  57 70.853212  0x0000               0x6cbe               ZigBee    75 Transport Key
   58 70.853721                                            IEEE 8…    5 Ack
   59 70.858920  0x0000               0x6cbe               ZigBee…   50 Active Endpoint Request, Nwk Addr: 0x6cbe
   60 70.859475                                            IEEE 8…    5 Ack
   61 71.023712  0x0000               Broadcast            ZigBee…   48 Permit Join Request
   62 71.198365  0x6cbe               0x0000               ZigBee    45 APS: Ack, Dst Endpt: 0, Src Endpt: 0
   63 71.198947                                            IEEE 8…    5 Ack
   64 71.247934  0x0000               Broadcast            ZigBee…   48 Permit Join Request
   65 71.517746  0x0000               Broadcast            ZigBee…   48 Permit Join Request
   66 72.049888  0x7513               0x0000               IEEE 8…   12 Data Request
```

Figure 4: The Messages (packets) captured on the ZigBee network in discovery phase

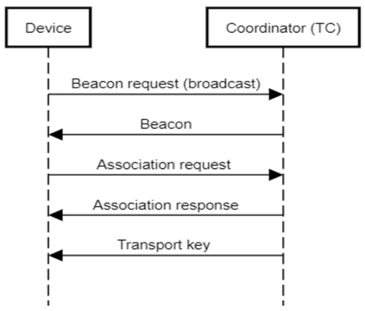
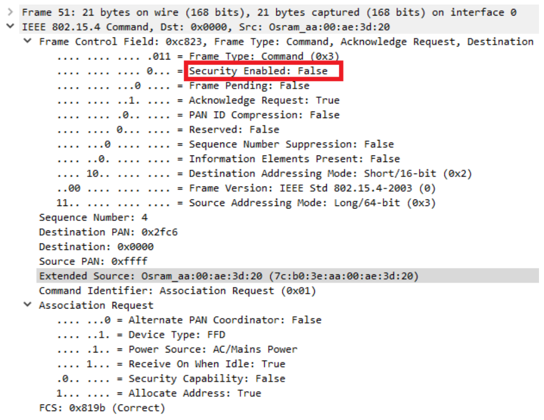
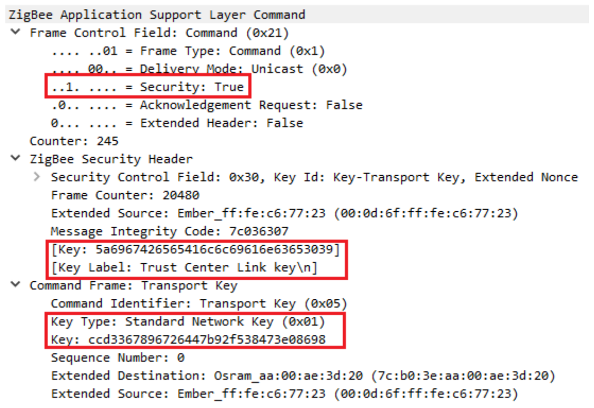


Figure 5: The Messages exchanged among Alexa and device during the discovery



```
> Frame 51: 21 bytes on wire (168 bits), 21 bytes captured (168 bits) on interface 0
v IEEE 802.15.4 Command, Dst: 0x0000, Src: Osram_aa:00:ae:3d:20
  v Frame Control Field: 0xc823, Frame Type: Command, Acknowledge Request, Destination
      .... .... .... .011 = Frame Type: Command (0x3)
      .... .... .... 0... = Security Enabled: False
      .... .... ...0 .... = Frame Pending: False
      .... .... ..1. .... = Acknowledge Request: True
      .... .... .0.. .... = PAN ID Compression: False
      .... .... 0... .... = Reserved: False
      .... ...0 .... .... = Sequence Number Suppression: False
      .... ..0. .... .... = Information Elements Present: False
      .... 10.. .... .... = Destination Addressing Mode: Short/16-bit (0x2)
      ..00 .... .... .... = Frame Version: IEEE Std 802.15.4-2003 (0)
      11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
    Sequence Number: 4
    Destination PAN: 0x2fc6
    Destination: 0x0000
    Source PAN: 0xffff
    Extended Source: Osram_aa:00:ae:3d:20 (7c:b0:3e:aa:00:ae:3d:20)
    Command Identifier: Association Request (0x01)
  v Association Request
      .... ...0 = Alternate PAN Coordinator: False
      .... ..1. = Device Type: FFD
      .... .1.. = Power Source: AC/Mains Power
      .... 1... = Receive On When Idle: True
      .0.. .... = Security Capability: False
      1... .... = Allocate Address: True
    FCS: 0x819b (Correct)
```

Figure 6: The message sent from Echo Plus outlines that security features for protecting the link key are disabled



```
ZigBee Application Support Layer Command
v Frame Control Field: Command (0x21)
    .... ..01 = Frame Type: Command (0x1)
    .... 00.. = Delivery Mode: Unicast (0x0)
    ..1. .... = Security: True
    .0.. .... = Acknowledgement Request: False
    0... .... = Extended Header: False
  Counter: 245
v ZigBee Security Header
  > Security Control Field: 0x30, Key Id: Key-Transport Key, Extended Nonce
    Frame Counter: 20480
    Extended Source: Ember_ff:fe:c6:77:23 (00:0d:6f:ff:fe:c6:77:23)
    Message Integrity Code: 7c036307
    [Key: 5a6967426565416c6c69616e63653039]
    [Key Label: Trust Center Link key\n]
v Command Frame: Transport Key
    Command Identifier: Transport Key (0x05)
    Key Type: Standard Network Key (0x01)
    Key: ccd3367896726447b92f538473e08698
    Sequence Number: 0
    Extended Destination: Osram_aa:00:ae:3d:20 (7c:b0:3e:aa:00:ae:3d:20)
    Extended Source: Ember_ff:fe:c6:77:23 (00:0d:6f:ff:fe:c6:77:23)
```

Figure 7: Messages exchanged among the device and Echo Plus, that are considered secure (Security Enabled)

tant, that needs a minimal intervention from the customer, become a security hole that enables any attacker physical near to the house to acquire control over all the home devices, being able to read messages and eventually control the devices. The main consideration is that a security-by-default approach should be imposed to systems with such a large diffusion. At the state of art, instead, the solutions offered have a low level of security in standard configuration and countermeasures should be applied by the end user. Device discovery should rely on an explicit authentication of devices (that implies a manual intervention of users) and communication keys should not be shared among multiple devices. The penetration testing methodology we adopted supports IoT-based systems and enable personnel with limited computer security skills to identify and demonstrate working attacks. This paper is a little step in the direction of a technique that aims at automating the processes related to security assurance and penetration testing, which are, in our opinion, one of the key requirements for the new generation of computing paradigms, where everything is offered as a service (cloud approach), self-managed (smart systems) and typically ubiquitous (IoT). In future works, we aim at improving our methodology in order to take into account the available knowledge base of attack techniques and tactics (e.g. ATT&CK) and attack patterns[3].

References

[1] R. Li, D. Abendroth, X. Lin, Y. Guo, H.-W. Baek, E. Eide, R. Ricci, and J. Van der Merwe, "Potassium," in *Proceedings of the Sixth ACM Symposium on Cloud Computing - SoCC '15*, 2015, pp. 30–42. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2806777.2806935

[2] J. Hu, Y. Wang, C. Tang, Z. Guan, F. Ren, and Z. Chen, "A Novel Framework to Carry Out Cloud Penetration Test," *I.J. Computer Network and Information Security*, vol. 3, no. 3, pp. 1–7, 2011. [Online]. Available: http://www.mecs-press.org/

[3] P. Kamongi, M. Gomathisankaran, and K. Kavi, "Nemesis: automated architecture for threat modeling and risk assessment for cloud computing," in *Proc. 6th ASE*, 2014. [Online]. Available: https://pdfs.semanticscholar.org/14ae/5e34f315c75c191f0c3325bbcd7e3475f4c4.pdf

[4] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Towards automated penetration testing for cloud applications," in *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, June 2018, pp. 24–29.

[5] V. Casola, A. D. Benedictis, M. Rak, and U. Villano, "Toward the automation of threat modeling and risk assessment in iot systems," *Internet of Things*, vol. 7, p. 100056, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2542660519300290

[6] W. Knowles, A. Baron, and T. McGarr, "The simulated security assessment ecosystem: Does penetration testing need standardisation?" *Computers and Security*, vol. 62, pp. 296–316, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2016.08.002

[7] A. C. Karen Scarfone, Murugiah Souppaya and A. Orebaugh, "Technical guide to information security testing and assessment," *NIST Special Publication 800-115*, 2008. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-115/final

[8] [Online]. Available: https://www.owasp.org/index.php/OWASP_Testing_Project

[9] [Online]. Available: https://www.owasp.org/index.php/Main_Page

[10] [Online]. Available: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

[11] P. Herzog, "Osstmm 3: The open source security testing methodology manual-contemporary secutiy testing and analysis–http://www. isecom. org/mirror," 2010.

[12] L. L. Matt Byrne, Arvind Doraiswamy and N. OUCHN. [Online]. Available: http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html#

[13] [Online]. Available: https://sourceforge.net/projects/isstf/

[14] [Online]. Available: https://metasploit.help.rapid7.com/docs/msf-overview

[15] M. Rak, "Security assurance of (multi-)cloud application with security sla composition," in *Green, Pervasive, and Cloud Computing*, M. H. A. Au, A. Castiglione, K.-K. R. Choo, F. Palmieri, and K.-C. Li, Eds. Cham: Springer International Publishing, 2017, pp. 786–799.

[16] [Online]. Available: MicrosoftCorporation,TheSTRIDEThreatModel,2016https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).

[17] International Organization Of Standardization, "ISO/IEC/IEEE 42010:2011 - Systems and software engineering – Architecture description," *ISOIECIEEE 420102011E Revision of ISOIEC 420102007 and IEEE Std 14712000*, vol. 2011, no. March, pp. 1–46, 2011.

[3]https://attack.mitre.orghttps://capec.mitre.org/data/index.html