Smart-TV security analysis: practical experiments

Yann Bachy, Frédéric Basse, Vincent Nicomette, Eric Alata, Mohamed Kaâniche, Jean-Christophe Courrège, Pierre Lukjanenko

Smart-TV security analysis: practical experiments

Yann Bachy*†§, Frédéric Basse¶, Vincent Nicomette*†, Eric Alata*†,
Mohamed Kaâniche*‡, Jean-Christophe Courrège§ and Pierre Lukjanenko*
*CNRS, LAAS, 7, Avenue du colonel Roche, F-31400 Toulouse, France
†Univ de Toulouse, INSA de Toulouse, LAAS F-31400 Toulouse, France
‡Univ de Toulouse, LAAS F-31400 Toulouse, France
§Thales Communications & Security, 3, avenue de l'Europe, 31400 Toulouse, France
¶Thales Communications & Security, 5, Rue Marcel Dassault, 78140 Vélizy-Villacoublay, France
* firstname.name@laas.fr, §¶ firstname.name@thalesgroup.com

Paper category: practical experience

***Abstract*—Modern home networks are becoming more and more complex with the integration of various types of interconnected smart devices, using heterogeneous networking technologies. Many of these devices are also connected to the Internet, generally through an integrated access device. Those smart devices are potentially vulnerable to several types of attacks. In this practical experience report we investigate the specific case of smart TVs. The main objective is to experimentally explore possible attack vectors and identify practically exploitable vulnerabilities and attack scenarios. In particular, the study covers local and remote attacks using different entry points, including the Digital Video Broadcasting (DVB) transmission channel and the copper-pair local loop. Several methods, allowing to observe and simulate service provider networks, are used to support several experiments considering four types of commercially available smart TVs for a comparative analysis. We also discuss several methods allowing to extract and analyze the embedded firmware, and obtain relevant information concerning target devices.**

I. Introduction

Digital technologies have become widely used in home networks. At their beginning, home Internet connections were only considered as an access to the World Wide Web. More services have been offered later to the users with the introduction of triple-play offers and the possibility to watch TV or initiate phone-calls using the Internet connection. The trend toward making every day home equipment more "Smart" has been strengthened during the recent years. As an example, kitchens are being equipped with smart-refrigerators, hospitalizations can be shortened by medical home equipment, and more and more housework can be supported by different smart automation systems. Many of these smart devices run operating systems and are connected among each other, by several networks, allowing them to interact and to be accessed remotely through the Internet. As a consequence the security of such devices has become a real concern[1]. As an example, the French Network and Information Security Agency (ANSSI) [2] is concerned by the possibility of Integrated Access Devices (IAD) or other connected home devices getting compromised and used to perform large scale attacks based e.g., on botnets.

A typical example concerns smart-TVs. Besides receiving TV programs using an aerial antenna or a satellite dish, these new-generation TVs integrate an operating system and an Ethernet connection, allowing them to offer more features to the users. Protocols, such as HbbTV[1], allow TV stations to combine interactive Internet content with their live TV shows. From a security point of view, two important aspects must be considered. Firstly, like any other embedded device, the embedded operating system may contain vulnerabilities that could be exploited by an attacker. Secondly, the simultaneous connectivity of the TV to several networks makes it possible for an attacker to use it as a gateway between these networks. Clearly, smart-TVs could represent a real security threat for home networks. Thus, it is important to analyze their security and then investigate protection mechanisms to mitigate the related security risks.

The main objective of paper is to experimentally explore possible attack vectors and identify practically exploitable vulnerabilities and attack scenarios. Our first contribution addresses the possibility of compromising a smart-TV through two public area networks: the ADSL network and the Digital Video Broadcasting (DVB) network. Our second contribution addresses the operating systems embedded in smart-TVs. We discuss a set of methods allowing to extract and analyze the firmware of smart-TVs. Indeed, the attack path explored in the first part of this paper has seldomly been investigated by related work in this area. We analyzed four types of commercially available smart TVs and observed significant differences from the security point of view. This attack path becomes even more dangerous if one has sufficient knowledge of the target device, and the embedded software. With this knowledge, far more subtle attacks can be performed, allowing an attacker to remotely alter the behavior of the firmware of any smart-TV or to replace the firmware by another. Whereas these attacks are usually performed using the local area network, they become feasible remotely through the new attack path combined with the knowledge of the firmware. Some examples of such attacks are discussed at the end of this paper.

This paper is structured as follows. First, Section II discusses attack entry points and related work. Section III describes the different solutions we used during our experiments. Then Section IV presents the different experiments we conducted and their results. Section V explores several techniques to retrieve firmwares. Section VI presents an attack scenario combining our novel attack paths and specific knowledge of target device firmware, leading to a dreadful attack.

[1]Hybrid Broadcast Broadband TV

Finally, Section VII concludes this work and outlines possible countermeasures and some perspectives for future work.

II. Attack surface and related works

This section presents the different attack paths concerning smart-TVs in a home network and their related works. Figure 1 illustrates a typical home network containing a Smart-TV with its multiple connexions. The different attack paths analyzed in this section are pointed out by green arrows.

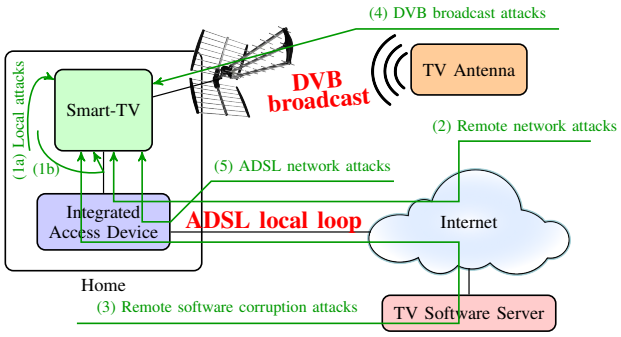

Fig. 1. Case Study: Smart-TV home network

A. Local and Remote attacks

Local attacks require a physical access to the target device $(1a)$, or an access to one of the local networks the target device is connected to $(1b)$. Therefore, these kinds of attacks are executed by someone in or nearby the target device's home. These local attacks are commonly addressed in the literature [3][4][5]. In [6] authors monitor the local home network traffic in order to infer which TV program the user is watching. There are many on-line forums centralizing information on how-to compromise a specific device, or group of devices ([7] [8]), e.g., by changing the firmware of the TV or enabling some hidden features.

Anyway, remote attacks consist in exploiting, from the Internet(2), some vulnerability included in a Smart-TV connected to a home network. Remote network attacks are often hampered by the presence of an IAD, using NAT techniques. Indirect attacks, using corrupted firmwares or malicious applications(3), are more common, since Smart-TVs present almost the same attack vectors as a personal computer[9][10][11].

B. DVB and ADSL network attacks

TV and ADSL[2] networks, are traditionally not considered as a source of threats. However, potential attacks exploiting vulnerabilities of such service provider networks could have an impact on the security of devices connected to these networks. In Figure 1, these networks are highlighted in red. DVB network attacks (4) were investigated recently, in particular in [12] that explored HbbTV privacy related issues. The first HbbTV attack scenarios are studied in [13]. To our knowledge, ADSL network attacks from the local loop (5) have been seldom addressed. These attacks could allow the attacker to gain access to a smart-TV in a user's home, compromise privacy, install malicious software or carry out denial-of-service attacks. In a more critical scenario, a smart-TV could be used as a gateway to attack other devices connected to a home network.

[2]Asymmetric Digital Subscriber Line

C. Discussion

We can notice that traditional attack paths $(1a)$, $(1b)$, (2) and (3) are commonly covered in the related state of the art research, unlike the other attack paths targeting DVB (4) and ADSL networks (5) that have been seldom addressed. There is also a lack of experimental results covering such attack scenarios. Moreover, to our knowledge, no previous study explored experimentally the potential security threats related to the possible combination of these different attack paths. The objective of this paper is to experimentally analyze vulnerabilities related to the connection to these networks and show how the exploitation of these vulnerabilities may endanger the security of the whole home-network.

In the following, we first describe our experimental setup. This setup includes, for both TV and ADSL networks practical techniques: i) to observe legitimate traffic on the network and identify related vulnerabilities, and ii) to simulate the service provider and then check the feasibility of some attack scenarios. Then in Section IV, we describe the different experiments we conducted on both ADSL networks and terrestrial DVB broadcasts and discuss the results.

III. Experimental setup

In this section, we describe the different solutions we have used or developed in order to conduct our experiments. Some of these solutions are based on existing software, others require a specific hardware setup. These platforms allowed us to carry out some comparative studies which are presented in section IV. For each study (ADSL local loop, DVB), we proceed in two steps 1) analyze and understand the legitimate traffic and behavior of the network, and 2) simulate the service provider on the other side of the network with whom the smart-TV communicates.

A. Local loop

The first study that we have carried out focused on the analysis of the local loop supporting the ADSL network. This network was historically installed for public switched telephone networks (PSTNs). Beside telephony, this network transports other services such as Internet and TV. This implies that many data communications, possibly critical, use this network. Considering all this makes it legitimate to consider the security of this network.

There are 4 main physical supports used for the local loop: copper pair, coaxial, fibre optics and radio waves. We hereinafter only consider copper pair local loops. The method developed in our study can be applied to any kind of local loop by using analogous hardware. This copper pair is terminated by a MoDem (*Modulator and Demodulator*) on the user's side, and by a DSLAM (*Digital subscriber Line Access Multiplexer*) on the provider's side. These two equipments modulate and demodulate a digital signal into high-frequency analog signals which are transmitted over the copper pair.

1) Traffic Observation setup: This setup is intended to observe any traffic on a copper pair local loop and is composed of a DSLAM and an ADSL modem.

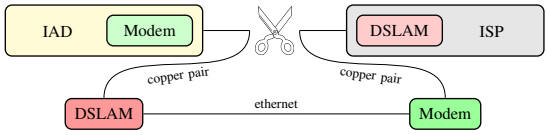

Fig. 2. Local loop traffic capturing

The DSLAM and modem on each end of the copper pair fulfil the exact opposite operation of each other. Indeed, the modulation used on the up-stream differs from the one used on the down-stream. Therefore, it is possible to physically cut the line and insert a new DSLAM and a new ADSL modem, while preserving the connectivity between the customer and the provider. In fact, this modification changes the way the IAD communicates with the provider's DSLAM: it synchronizes and communicates with the inserted DSLAM; the inserted DSLAM communicates with the inserted ADSL modem which, in turn, synchronizes and communicates with the provider's DSLAM. As the local network interface included in both modems and DSLAM are most of the time Ethernet, this manipulation finally consists in transforming one copper line into two copper lines interconnected by an Ethernet LAN (see Figure 2). As sniffing on an Ethernet LAN is very easy, the communications sent and received by the IAD during setup can be observed.

2) Simulation setup: This platform simulates a service provider based on the knowledge of our target system acquired using the traffic observation platform or other reverse engineering techniques such as those described in Section V. Here, instead of connecting our DSLAM to the provider's network using a modem, we connect it to a computer capable of simulating the behaviour of a service provider (cf. Figure 3). The installation of this server is accomplished step by step following an iterative method. We cyclically reboot the target system, observe any request coming from the target system and answer it by installing corresponding software based on the knowledge we have of the target system.

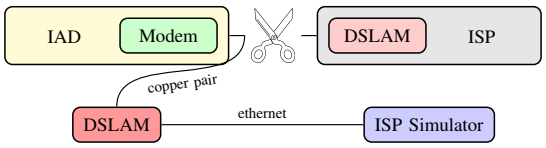

Fig. 3. Emulating an Internet Provider

B. DVB Broadcasts

The second service provider network we have tested is the aerial DVB-T system used for digital Television. TV broadcasts are the same for everybody and are generally considered as trustworthy. Classical television is a one-way system, which implies that consumers do not have to worry about privacy in any way. With Digital TV and smart-TVs, these assumptions are no longer valid, which makes it interesting to analyze their security. DVB Broadcast security concerns have already been addressed[13] using some relevant aspects of modern Smart-TVs. However, we consider some interesting experiments that have not been carried out so far, and we hereinafter present the solutions we have used to conduct some comparative studies on different types of smart-TVs available on the market.

1) DVB: DVB is a suite of standards for digital television transmission[14]. It allows transmission of MPEG-2 Transport Streams, containing multiple video, audio or data streams, over several supports such as satellite, aerial or cable connections. Hereinafter we only consider aerial connections, also known as terrestrial transmissions, which is the most used support over the world[15]. Such aerial transmissions are terminated by a DVB-T demodulator on the user's side, and by a DVB-T modulator on the provider's side.

2) Traffic observation setup: The objective is to observe every stream received by a TV tuned into a specific frequency. As listening to a DVB-T broadcast signal is the everyday task of any TV, this solution does not require any specific hardware. Any off the shelf DVB-T demodulator can be used. Open-source software such as DVBSnoop[3] allows DVB MPEG stream analysis. We use tzap software[4] to tune the demodulator into a specific frequency.

3) Simulation solution: This solution is intended to simulate a legitimate DVB broadcast. In order to broadcast a valid DVB channel, we need a DVB valid MPEG TS and a DVB-T modulator. Many popular open-source applications such as ffmpeg[5] and vlc[6], are capable of generating and modulating MPEG TSs. However, these applications cannot be used to obtain a DVB valid MPEG TS as they are not able to create DVB-required signalling tables. This can be achieved by the opensource application: Avalpa OpenCaster[7]. This task can be initiated by capturing a legitimate signal (cf. Figure 4 - traffic observation setup).

Since broadcasting TV without a licence is prohibited in most countries, DVB modulators are generally not available off-the-shelf. During our experiments, we have used two solutions. The first one uses hardware suggested by OpenCaster, which can be purchased online. This DVB modulator functions "out-of-the-box" with OpenCaster, but is limited in some modulation parameters, i.e., it only supports QPSK or QAM16 constellations, considerably reducing the available bandwidth. Since many countries use the QAM64 constellation, this hardware is unable to entirely simulate a DVB Multiplex. The second platform we have experimented requires more expensive SDR (Software Defined Radio) hardware. In our case, we used the Ettus N210 with the WBX daughterboard. Using GNU-Radio[8] makes it possible to turn such a device into any kind of radio modulator, including a DVB-T modulator. The popularity of GNU-Radio allowed us to find an entire DVB-T modulation scheme[16] online. Combining one of these two hardware solutions with OpenCaster allowed us to set up a fully functional DVB-T modulator (cf. Figure 4 - simulation solution).

[3] http://dvbsnoop.sourceforge.net/
[4] http://www.linuxtv.org/wiki/index.php/LinuxTV_dvb-apps
[5] https://www.ffmpeg.org/
[6] http://www.videolan.org/vlc/
[7] http://www.avalpa.com/the-key-values/15-free-software/33-opencaster
[8] http://gnuradio.org/

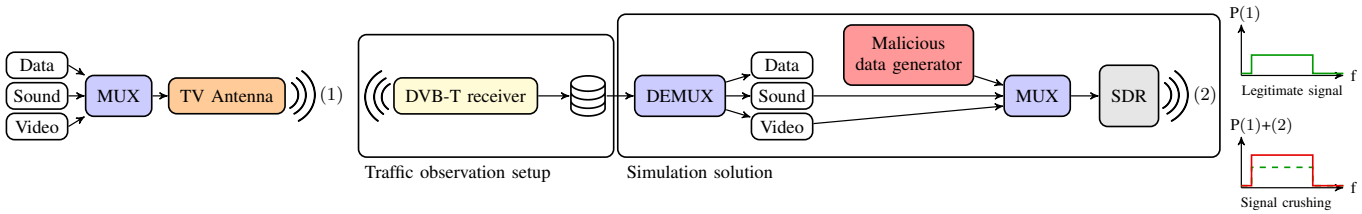

Fig. 4. DVB experimental setup

When considering wired networks, physically connecting the end user to the simulation platform forces the end users terminal to communicate with this platform. In our case, using aerial transmission, we can't just plug out the legitimate service provider and plug in our simulation platform instead. We need to crush the legitimate signal, which can be done by transmitting with significantly more power. The International Telecommunication Union defines [17] safety ratios, to make sure a weaker signal won't interfere. Thus, if one transmits above these safety ratios, the emitted signal crushes the legitimate one, and any TV around considers this signal rather than the legitimate one (cf. Figure 4).

It is important to note that we did not carry out such large-scale experiments. Instead, we limited our experiments to our research lab making sure that our experiments did not interfere with any legitimate signal.

IV. Experimental results

We used the traffic observation and service provider simulation platforms described in Section III to carry out several experiments on a panel of 4 2013 main-brand midrange Smart-TVs, which were the most sold smart-TVs in Europe for 2013, and are therefore representative of the average domestic smart-TV. Hereinafter, these TVs are anonymously referenced as A, B, C and D.

A first set of experiments uses the local loop setups in order to analyze and compare communications between a Smart-TV and its smart-content service provider. As the smart TV firmware update procedure is critical, a first experiment focuses on potential vulnerabilities of this update procedure. A second experiment focuses on the smart-TV's integrated Web browser and its compliance to the same-origin security policy[9].
A second set of experiments uses our DVB Broadcasting platforms, allowing us to analyze and compare the behaviour of each Smart-TV when receiving a legitimate and a compromised signal. A first experiment shows another interesting way of compromising a DVB-T Broadcast. A second experiment analyzes again the compliance to the same-origin security policy, but now using the embedded HbbTV browser.
The result of this last experiment allows us to illustrate a combined attack scenario, compromising a home network due to lack of security in Smart-TVs and their respective networks.

Table I summarises the experiments that we have carried out considering local loop and DVB broadcast attacks. The results are detailed in the following subsections.

[9]The same-origin security policy defines that the local execution of a code (i.e. javascript), downloaded from a remote Web site, cannot send data to a Web site who's origin (URL) differs from the one of the original Web site.

TABLE I. Experiments summary

	1^{st} experiment	2^{nd} experiment
Local loop Setup	Firmware update procedure analysis	Web-Browser & same-origin policy
DVB Setup	DVB Broadcast authentication	HbbTV Browser & same-origin policy

A. Local loop experiments

The first two experiments are carried out using our local loop observation and service provider simulation platforms. These experiments could easily have been carried out on the local area network if the attacker was inside the target home. By operating on the local loop, these experiments demonstrate the possibility to compromise someone else's TV. Operating directly on the copper pair local loop is technically more complicated as it requires a specific hardware setup. However, operating directly on the local loop presents a higher security impact, as it possibly allows to compromise any home.

1) Smart-TV firmware update procedures: The first experiment carried out on our panel of smart-TVs intends to analyze whether the communication durign the firmware update procedure are protected. Our local loop observation platform (cf. III-A1) was placed behind the IAD, allowing us to observe any communication between the smart-TV and its smart-content provider[10]. The results of this experiment are presented in table II.

TABLE II. Smart-TV firmware update procedures

		A	B	C	D
Negotiation	protocol	HTTP	HTTP + HTTPS	HTTPS	HTTP
	content	unknown	XML + –	–	XML
Transfer	protocol	HTTP	n/a	HTTP	HTTP
	content	Binary	n/a	Binary	Binary

In each firmware update procedure two phases are distinguished. First, a "negotiation" phase that checks if a more recent firmware version is available for this Smart-TV. Secondly, if a newer version is available, a "transfer" phase wherein the actual firmware is transferred. For each phase, we observed the protocols that are used, and the type of content. Negotiation phases of A, B and C are very similar, they either use the secured HTTPS protocol, in which case the content is ciphered, or, when a non-secured protocol such as HTTP is used, the content uses an unknown encoding and is therefore also ciphered. The negotiation phase of D uses the non-secured HTTP protocol and its content is human readable

[10]In this particular case, the smart-content provider of a smart-TV is considered as the servers belonging to the corresponding brand of the TV.

XML. This allows a classic "man-in-the-middle" attack to substitute the URL leading to the new firmware and force the TV to download a different firmware.
All observed firmware transfer phases use the non-secured HTTP protocol. For each of these TVs, we carried out a "man-in-the-middle" attack simulating the update server during this phase, proposing legitimate but outdated firmware[11]. Smart-TVs A and B refused our outdated firmware without specifying any reason. It is likely that some signature is exchanged during the negotiation phase. Smart-TV D on the contrary accepted our outdated firmware. This security breach allows any attacker to exploit previously corrected security flaws on this TV.

2) Smart-TVs and the same-origin security policy: The second experiment on our panel of smart-TVs intends to verify the compliance of the integrated Web browser of each Smart-TV to the same-origin security policy. During this experiment, we connected each TV to our ISP simulator (cf. III-A2). On our ISP simulator, we hosted a Web site containing malicious JavaScript code vulnerable to an XSS attack. This JavaScript simply attempts to perform a HTTP POST request on a different[12] Web site. When a Web browser fully complies to the same-origin security policy, it must first either send out an "OPTIONS" request as shown in fig. 5, or in the worst case simply ignore the request.

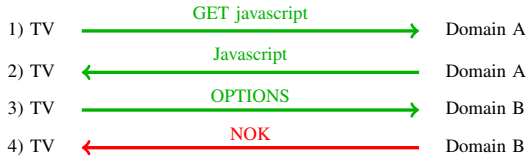

Fig. 5. Same-origin security compliant browser

TABLE III. SMART-TV WEB BROWSER AND THE SAME-ORIGIN POLICY

TV	A	B	C	D
Behaviour	Ignore	OPTIONS	OPTIONS	POST

Table III reports the behaviour of our four Smart-TVs. A, B and C are compliant to the same-origin security policy. D does not implement this security policy and sends out the malicious POST request. This security breach can be exploited by any phishing attack, i.e., by imitating a popular Web site.

B. DVB broadcast experiments

This set of experiments was carried out using our DVB observation and broadcast simulation platforms. They are aimed at investigating if the content of a TV channel is authenticated by a TV set.
DVB basically consists in broadcasting an MPEG TS, which is a multiplex of multiple Video, Audio and Data streams. DVB stipulates[18] that an MPEG TS can contain multiple channels of different quality. Oren and Keromytis [13] propose a solution similar to our DVB broadcast simulation solution (cf. III-B3). Their solution intends to compromise a stream in real-time and therefore requires directional antennas. Our experiments use a pre-recorded part of the target multiplex. We then use the same techniques to modify the content of a sub-stream in the multiplex. This technique is more visible to the end user as it will necessarily notice a flash-back in the TV program it is watching, but it requires less hardware.

1) Compromising a DVB Broadcast: This first experiment intends to demonstrate that none of the content of a DVB Broadcast is authenticated. In this experiment, we replace one of the video streams by a live webcam video feed. All the TVs of the test-panel ignore the legitimate signal when our platform is activated, and instead, show the live webcam feed. This experiment is achieved by recording a short duration of a legitimate DVB broadcast. Using our platform, we extracted some of the video streams out of the multiplex and then inserted our own video source instead.

2) HbbTV and the same-origin security policy: Similarly to the experiment described in subsection IV-A2, we tested the same-origin security policy compliance using the HbbTV protocol. HbbTV allows DVB to include Internet content in a multiplex, either by multiplexing the entire Web page into the broadcast, or by supplying the URL allowing the smart-TV to access the Web page using its Internet connection. Same-origin policy issues are discussed in [13], where authors are concerned about the possibility to define the origin when the Web page is entirely multiplexed into the broadcast. In this case no origin is defined and a specific property, `simple_application_boundary_descriptor`[19, S6.3] allows the malicious broadcaster to define it's own origin. We tested the same-origin security policy on our 4 smart-TVs with and without specifying this property. Results of these experiments are reported in Table IV.

TABLE IV. SMART-TV HBBTV AND THE SAME-ORIGIN POLICY

`boundary descriptor`	A	B	C	D
With	POST	ignore	OPTIONS	OPTIONS
Without	POST	ignore	OPTIONS	OPTIONS

These results are surprising and point out some interesting facts. First, there is no difference whether the `simple_application_boundary_descriptor` property is defined or not. This can be explained by the TV introducing a FQDN[13] when the Web page is embedded in the DVB broadcast, this FQDN appears in the origin headers as `dvb://1.1.1.b`. Therefore the `simple_application_boundary_descriptor` property is ignored because the application boundary can only contain one FQDN[19, S6.3]. Secondly, where the Web browser of A would respect the same origin security policy by ignoring the XSS attack, it now accepts to execute the attack and thereby ignores the security policy. On the other hand, D now fully respects the same-origin security policy while it did not in our previous experiment. Also B, even though it still complies with the security policy, behaves differently by ignoring the attack. This could mean that at least A, B and D don't use the same rendering software engine for the normal Web browser as for HbbTV. We exploited this security flaw on TV A by multiplexing a Web page, containing malicious JavaScript,

[11] We downloaded and archived legitimate firmware from the constructors Web sites.

[12] In XSS, we consider a different Web site as one having a different Fully Qualified Domain Name (FQDN).

[13] Fully Qualified Domain Name

directly into our malicious DVB signal[14]. This JavaScript code sends out a UPNP request to the local IAD, asking it to open specific ports to TV A. Doing so allowed us to conduct attacks which were, up to now, only feasible from the local area network.

All these experiments allowed us to demonstrate the pertinence of the attack paths highlighted in our study. Using these methods opens more attack possibilities, and allows an attacker to compromise remote locations using existing attacks which were up to now limited to local area networks. As an example our experiments allowed us to 1) replace the firmware of a TV and 2) access specific services of a TV over the Internet. However, all these experiments have been conducted without any particular knowledge of the firmware. These attacks can be far more dangerous with this knowledge, just like spear phishing is far more dangerous than classic phishing. Therefore, a second part of our research, is focussed on firmware analysis, in order to complete these works. The next section describes several techniques and some experimental results allowing to analyze such firmware.

V. Firmware analysis

This section describes several techniques to retrieve firmware of smart-TVs. Obtaining these firmwares allows us to analyze them in order to discover possibly important security flaws. Many smart-TV manufacturers make firmware updates available on their Web site. These updates usually contain a full firmware image that could allow us to start our analysis right away, even without owning the corresponding TV. However, the format and structure of a firmware image can be highly different from one TV to another. Moreover, these updates are often compressed or encrypted and therefore not directly useful. Hereinafter we discuss different techniques which may be used in case a firmware is encrypted in any way or not publicly available.

A. Firmware updates analysis

In many cases, firmwares are directly available from the corresponding manufacturers Web site. These firmware images can be packaged into an update application. By executing this application on a computer, it connects to the smart-TV using network, USB or other available interfaces, and upgrades its firmware. While reverse engineering the update application is a possible option, another solution is to impersonate the TV, provided knowledge of the protocol. A common and standardized protocol is DFU, a USB device class specification for Device Firmware Upgrading (DFU) over USB. Some studies, detailed in [20] demonstrate how to capture a firmware upgrade by emulating a DFU device.

Another common case is the use of raw binary firmware blobs[15] with a not-so-obvious file format at first glance. Although the main file header and structure may be proprietary, common file formats are quite often found inside by carving the entire file. These parts of the firmware can be extracted and analyzed with popular tools such are Binwalk[16] and firmware-mod-kit[17]. These tools are able to analyze and extract compressed streams, known file types or binary machine code. Kernel and filesystem images are commonly extracted at this step. Otherwise, an entropy analysis may help to determine if a part or the entire file is compressed and/or encrypted. Encrypted firmware images are quite common, but most of the time, the chosen ciphers are vulnerable. For instance, encryptions based on a XOR operation, using a weak key, have been found and defeated multiple times[21][22]. Nevertheless, well designed update processes, ensuring confidentiality, exist. In these cases, it is necessary to explore other attack vectors with physical access to the target TV.

B. Physical access

A majority of smart-TVs use flash memories to store their firmware. There are two main types of flash memories. Firstly, NOR memories allow random access to the data (at byte level). This kind of interface allows processors to execute-in-place (XIP) code stored on the chip. Secondly, NAND memories, that were introduced later, offer better erase and write speeds so as greater density and therefore are cheaper. However, they do not allow random access to the data, but only on page (or block) basis. Also, NAND memories need to store additional information named OOB data, used for error correction codes (ECC) and bad block management.

Some TVs rely on internal bootROM or NOR chip to store XIP bootstrap code, which copies the bootloader from the NAND memory into RAM. Newer processors may integrate a NAND flash interface to support direct boot from NAND memory. If the target TV can be pulled apart allowing to desolder these memory chips, it is possible to directly interface to the flash chip and extract a raw dump[23][24]. After that, specific software can check ECC and strip out OOB data. Finally, Binwalk can be used to figure out the memory layout and extract interesting data. The counterpart of this type of attack is the necessity of hardware modification, which may potentially be destructive. Moreover, this attack fails if data is correctly encrypted or stored in the locked area (RPMB) of the flash memory. This leads us to explore live attacks.

C. Debugging interfaces

Popular attack vectors on embedded devices are debugging ports such as serial ports and JTAG. Serial ports usually output bootloader & operating system messages. They can be used to interrupt boot at bootloader stage. This gives access to the bootloader prompt, which may permit to read and write flash memory. This prompt may also be used to modify OS parameters. For instance, on a Linux kernel, it can be used to enable the serial console which activates a shell prompt once booted. Another common interface is JTAG. JTAG interfaces are designed for CPU debugging, and therefore allow flash memory reading and writing. However, interfacing with JTAG requires specific hardware specification knowledge of the target board and its CPU. Ultimately, depending on the device, each debugging interface can be restricted, locked, or even disabled.

[14]This attack can be achieved either by including the JavaScript in a data carousel multiplexed into the DVB signal, or by supplying an URL pointing to an external Web page containing this malicious code.

[15]Firmware containing small amounts of source-closed binary code.

[16]http://binwalk.org/

[17]https://code.google.com/p/firmware-mod-kit/

D. Vulnerability exploitation

Smart-TVs usually embed several communication buses that could be targeted by attacks, like network, storage or other peripheral interfaces. Successfully exploiting a software vulnerability to gain arbitrary code execution on a TV could lead to firmware extraction. But finding and exploiting these vulnerabilities is a quite complex task, without access to the firmware code. Nevertheless, in this context, it is possible first to focus on vulnerabilities that do not rely on memory corruption (because the memory layout and content are unknown). From the network interface, embedded Web services are ideal to conduct this kind of attacks. Web servers are commonly found vulnerable to path traversals and shell injections. One can also attack Web clients that run on the target device by impersonating the server or altering its responses. When initiating a TLS connection, some devices don't check server certificate and thus are exposed to TLS Man in the Middle attacks. By injecting arbitrary JavaScript coreferences by asde into a HTTP response, one can explore the JavaScript execution context (if any) and collect information for further attacks.
By using more advanced techniques, certain memory corruption bugs can be turned into exploitable vulnerabilities. As an example[25], describes a Blind Return Oriented Programming technique to write exploits without possessing the target's binary.

E. Experimental results

We were able to conduct several experiments on one Smart TV of our test panel. These experiments were performed using a mix of "Firmware updates analysis", "Debugging interfaces" and "Vulnerability exploitation" techniques.

The first experiment consisted in the analysis of firmware updates, that can be automatically uploaded on each Smart TV. In the case of our Smart TV, this firmware is encrypted, using AES encryption in ECB mode. Thanks to 1) information displayed on the debugging interfaces and 2) vulnerability exploitation of a service network of the TV (directory path traversals vulnerability), we were able to extract the MIPS binary program that processes these updates of our Smart TV. Thanks to reverse engineering tools, we were able to understand the part of this program that checks the signature of the firmware and decrypts it. The analysis revealed that this decryption operation uses a key that is locally stored on the TV itself. Thanks to the directory path traversals vulnerability again, we were able to extract this key and then decipher the update. This allowed us to extract the firmware of the TV from the update. Let us note that this allows us to extract any future firmware that is encrypted in the same way.

The second experimentation allowed us to gain a live SSH access to the Smart TV, and revealed the whole file system of the firmware, letting us extract and analyze any file of this system. This attack was performed using the "Vulnerability exploitation" technique. More precisely, by sniffing the network communications of our Smart-TV, we discovered that the UPNP service was activated and that the version of libupnp (open source library) included a documented vulnerability. We gathered enough information on target architecture, OS, vulnerable code and used a compiler to predict the behavior and memory layout of the vulnerable process. With this information, we managed to remotely execute arbitrary code on the target and finally extract its firmware. More precisely, thanks to the successful exploitation of this vulnerability (a stack buffer overflow attack), we were able to execute commands on the Smart TV. These commands allowed us to start an SSH server on the TV. Using this access, it then becomes possible to access and analyze the contents of the file system of the firmware. These experiments, summarized in this section, are described in detail (in French) in [26][27].

VI. Combined exploitation

Using one of the techniques described in the previous section allows one to retrieve almost any firmware from embedded devices, or at least from any smart-TV. Analyzing the firmware allows an attacker to gain precious information about the target device and develop more specific and harmful attacks. This section shows how it is possible to combine the attack paths discussed in Sections III & IV with the knowledge obtained by analyzing a Smart-TV firmware, and perform a more dreadful attack.

In the previous section, we described a vulnerability exploitation allowing us to obtain the firmware of one of our smart-TVs. By analyzing this firmware we discovered other vulnerabilities in different services activated on this TV. Even if the exploitation of these vulnerabilities is particularly interesting for an attacker, their scope is limited, due to massive use of NAT techniques on home networks. The usage of NAT limits direct access from the Internet to devices connected through a local area network. Therefore these attacks can only be executed from machines connected to the LAN, i.e, from inside the house. The scope and the scale of these attacks may drastically change if they could be carried out from outside the house, i.e., from the Internet. This is where the attack path described in section IV-B2 comes in. Indeed, in this section, we presented a first exploitation on a smart-TV using an attack path combining the usage of a DVB broadcast and a home Internet connection. One part of this attack consisted in forging a UPNP request asking the local IAD to open a specific port to the target device. This operation can be repeated in order to open any other port. If we repeat this operation until we have opened all the available services on a smart-TV to the Internet, the TV is no longer protected by NAT. Attacks which would up to now require a direct access to the local area network, are now fully functional through the Internet. This means that 1) thanks to our attack paths, it is possible to exploit software vulnerabilities remotely on a TV and 2) thanks to the knowledge of the smart-TV firmware, it is possible to remotely perform quite complex and particularly dangerous attacks. This combined attack is described in figure 6. From there on, it is possible to imagine any other attack on a home network. Indeed, once the attacker gains control over the TV, he may use it as a gateway in order to exploit vulnerabilities in any other smart device inside the home. As a consequence, the corruption of a Smart TV from the Internet may possibly endanger other devices connected to the home network.

VII. Conclusion

In this paper, we have first presented a new attack path that allows remote vulnerability exploitation on Smart devices in a home network. This new attack path was instantiated on a panel of different types of commercially available Smart

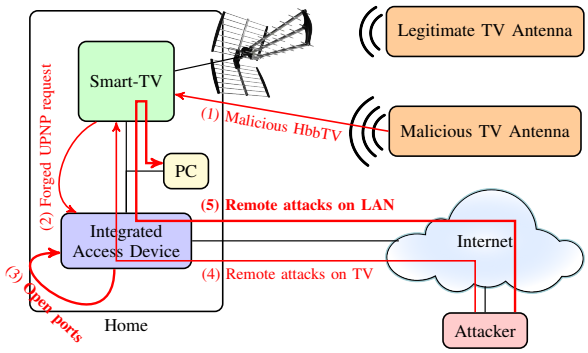

Fig. 6. Combined attack

TVs. Our experiments allowed us to successfully 1) modify the firmware of some models of Smart TVs and 2) exploit a cross-site vulnerability in HbbTV browsers of some models, which in turn can be used to modify the UPNP configuration of the home network IAD, opening a new remote access from the Internet to the Smart TV. These attacks were performed without any particular knowledge of the smart-TV firmwares. In the second part of this paper, we presented a set of techniques that can be used to extract and analyze the firmware of a Smart TV. These techniques allow to point out some interesting vulnerabilities that can remotely be exploited thanks to the new attack path proposed in the first part of this paper. The combination of this attack path and the precise knowledge of vulnerabilities in the firmware can lead to more subtle and dangerous attacks. An example of such an attack was presented at the end of this paper.

Several countermeasures to address the weaknesses pointed out in this paper deserve to be analyzed. These include: 1) Generalizing the use of cryptographic methods during critical exchanges. In this study we can see that many Smart-TV manufacturers have already chosen to use the secure HTTPS protocol or other proprietary encryptions in order to secure their firmware update process. 2) Measuring variation of signal attenuation on the ADSL line, since this value should drastically change when one inserts our platform on the local loop. 3) Measuring variation of TV Signal strength, since this value should drastically change when one activates our DVB simulation solution. 4) Correctly implementing cross-site policy in Smart-TV browsers, as any other standard security policy. The browser used by a Smart-TV should be at least as secure as those used on PCs. 5) Implementing encryption or authentication techniques in DVB signals.

As future work, we plan to extend our experiments to other families of home networks, smart devices and analyze potential attack propagation possibilities.

REFERENCES

[1] N. Feamster, "Outsourcing home network security," in *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, ser. HomeNets '10. New York, NY, USA: ACM, 2010, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/1851307.1851317

[2] "Défense et sécurité nationale." Paris, France http://www.elysee.fr/assets/pdf/Livre-Blanc.pdf: Direction de l'information légale et administrive, 2013, pp. 44–45.

[3] N. Ruff, "Sécurité de l'adsl en france," in *proc. of Symposium sur la sécurité des technologies de l'information et des communications (SSTIC)*, Rennes, France, June 1st 2006.

[4] N. Sidiropoulos and P. Stefopoulos, "Smart tv hacking," in *Research project 1*, Amsterdam, Netherlands, January 2013. [Online]. Available: http://delaat.net/rp/2012-2013/p39/report.pdf

[5] Proofpoint, "Samygo." http://www.proofpoint.com/threatinsight/posts/your-fridge-is-full-of-spam-proof-of-a-Iot-driven-attack.php: Proofpoint.

[6] M. Ghiglieri and E. Tews, "A privacy protection system for hbbtv in smart tvs," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, Jan 2014, pp. 357–362.

[7] E. U. Altinyurt, "Samygo." http://www.samygo.tv: SamyGO.

[8] "Openlgtv," http://openlgtv.org.ru/wiki/index.php/Wiki_index.

[9] S. Lee, "Hacking, surveilling, and deceiving victims on smart tv," 2013. [Online]. Available: http://blackhat.com/us-13/briefings.html#Lee

[10] J. Y. Aaron Grattafiori, "The outer limits: Hacking the samsung smart tv," 2013. [Online]. Available: http://blackhat.com/us-13/briefings.html#Grattafiori

[11] B. Michele and A. Karpow, "Watch and be watched: Compromising all smart tv generations," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, Jan 2014, pp. 351–356.

[12] M. Herfurt, "Hbbtv security," 2013. [Online]. Available: https://events.ccc.de/congress/2013/Fahrplan/events/5398.html

[13] Y. Oren and A. D. Keromytis, "From the aether to the ethernet—attacking the internet using broadcast digital television," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 353–368. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/oren

[14] J. Stott, "The dvb terrestrial (dvb-t) specification and its implementation in a practical modem," in *Broadcasting Convention, International (Conf. Publ. No. 428)*, Sep 1996, pp. 255–260.

[15] European Commission, "Special eurobarometer 396 - e-communications household survey," http://ec.europa.eu/digital-agenda/en/news/special-eurobarometer-396-e-communications-household-survey, 2013.

[16] Bogdan, "Dvb-t implementation in gnuradio part 2." http://yo3iiu.ro/blog/?p=1220: YO3IIU.

[17] I. T. Union, "Itu r-rec-bt.1368-11 year=2014," in *Planning criteria, including protection ratios, for digital terrestrial television services in the VHF/UHF bands*.

[18] U. Reimers, "The dvb project-digital television for europe," in *DVB (Digital Video Broadcasting): The Future for Television Broadcasting?), IEE Colloquium on (Digest No.1995/142)*, Jun 1995, pp. 1/1–1/7.

[19] European Broadcasting Union, "Etsi ts 102 796 v1.2.1," in *Hybrid Broadcast Broadband TV*, November 2012.

[20] T. Goodspeed, "Emulating usb dfu to capture firmware," 2012. [Online]. Available: http://travisgoodspeed.blogspot.com/2012/10/emulating-usb-dfu-to-capture-firmware.html

[21] M. Jordon, "Hacking canon pixma printers - doomed encryption," 2014. [Online]. Available: http://www.contextis.co.uk/resources/blog/hacking-canon-pixma-printers-doomed-encryption/

[22] "Sitecom firmware encryption and wireless keys," 2014. [Online]. Available: http://blog.emaze.net/2014/04/sitecom-firmware-and-wifi.html

[23] M. Oh, "Reverse engineering nand flash memory," 2014. [Online]. Available: http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Reverse-Engineering-NAND-Flash-Memory-POS-device-case-study-part/ba-p/6581528

[24] J.-M. Picod, "From nand chip to files," 2014. [Online]. Available: http://blog.j-michel.org/post/86992432269/from-nand-chip-to-files

[25] "Hacking blind," 2014. [Online]. Available: http://www.scs.stanford.edu/brop/

[26] F. Basse, "Sécurité des ordivisions," 2014. [Online]. Available: https://www.sstic.org/media/SSTIC2014/SSTIC-actes/securite_des_ordivisions/SSTIC2014-Article-securite_des_ordivisions-basse.pdf

[27] ——, "Télévisions connectées : Des objets branchés sécurité?" in *MISC*, September / October 2014.