Smart TVs: Security of DVB-T

Tristan Claverie, Jose Lopes Esteves, and Chaouki Kasmi
prenom.nom@ssi.gouv.fr

Wireless Security Lab, ANSSI

Abstract. During the last decade, classical televisions have experienced the same evolution as computers and feature phones by integrating new capabilities. Reading emails, installing applications, surfing the web are now common usages of so-called Smart TVs. With new services and communication interfaces come new interests from IT security researchers. Multiple studies dealing with security and privacy issues on a large number of models have been released. Since 2013, the RF signal broadcasting digital television contents has been shown as a new attack vector, using interactive applications being received through the radiocommunication interface. In this paper, the compounds of a DVB-T stream as well as the mechanism of interactive multimedia applications are presented. A recent extension of those standards designed to mitigate the interactive applications attack vector has been published in early 2017 that will be discussed with regards to weaknesses uncovered during our experimentations. Furthermore, several shortcomings of this specification will be discussed listing possible efficient countermeasures for each of them.

1 Introduction

During the last decades, the evolution of most of electronic devices has been observed. From basic electronics to smart devices composed of multiple sensors and interfaces to increase data exchanges, new features have been integrated. Browsing the Internet in smart cars, streaming movies on smartphones, sending emails from Smart TVs are now common usages. Nevertheless, the security of these smart devices running complex software on powerful hardware is still far from what users are expecting. One of the smart devices of high interest is the Smart TV. Used in companies reception desks as well as in meeting rooms, these devices become widespread and are permanently connected to the network and by extension to the Internet. Most security studies have focused on the operating system running on these devices, applications available on dedicated stores, update mechanisms as well as physical attacks against peripherals. Smart TVs possess numerous interfaces for communicating with the outside world. Those interfaces are very potent and could do harm if controlled by an attacker. Smart TVs can be attacked in order to

access some or all of them, depending of the original goal. The literature states two different purposes for taking control over a Smart TV: either access some secret data that would be stored in it, or use a compromised Smart TV as part of a larger attack, *e.g.* as an entry point for an entreprise network.

While every Smart TV model is different, there are common interfaces that every device has. They are presented based on the type of access required to attack them:

- **Physical access** - Under physical access falls every physical port present on a Smart TV. Usually those are Universal Serial Bus (USB), High-Definition Multimedia Interface (HDMI), RJ45, Common Interface (CI+)... It also contains a micro and/or camera either built-in or pluggable through USB, which is valuable for an attacker.
- **LAN access** - It represents every internal network service that runs on a television and is accessible through the LAN.
- **Radio access** - It includes the broadcast interface with which the terrestrial/satellite signal is received but also the WiFi and Bluetooth interfaces which have been observed on the different models of Smart TV studied.
- **WAN access** - Under this one fall all attacks which primarily rely on the television accessing a malicious web page. Also, users have the ability to install additional applications on their Smart TVs just like on smartphones. Some attacks are making use of that and fall in the same category.

It is worth mentioning that these types of attack are not the prerogative of Smart TVs alone. The use of malicious applications to gain privileges happens on desktop and mobile devices. Network services are attacked on many Internet-enabled devices and not just Smart TVs. As for physical ports, they are not specific to Smart TVs either. Interested readers will find a summary of general attacks against Smart TVs in Appendix A. Considered as inaccessible for a long time, the RF broadcast channel has been considered as out of scope of most studies. With the appearance of dedicated RF dongles and software defined radio tools in parallel of the introduction of new features in the RF channel, this interface has become of high interest for the information security community. Researches have been published demonstrating critical flaws directly related to the absence of signal authentication in digital television. Going further, the introduction of unsigned and non authenticated interactive applications have provided a cheap and ready to use remote code execution on web-based technologies (XML, HTML, JavaScript). Those previous statements

and findings raise the question of the security of Smart TVs in their role of television, that is the security of their broadcast interface. The analysis of related work about the security of the RF broadcast channel of Smart TVs is proposed in the next section.

1.1 Related work

The broadcast interface is always on, and there exists no way of turning it off on Smart TVs. Moreover, there is no authentication of any kind and the data coming from the radio interface is considered trusted by receivers. This broadcast interface hence makes a very powerful attack vector for Smart TVs. Apart from one [24], all reasearchers have focused on interactive applications in Digital Video Broadcasting — Terrestrial (DVB-T) streams. An interactive application is a set of files that is made available by broadcasting certain data in a digital television stream. Those files belong to a particular channel and the television interprets them when displaying this channel. Interactive applications usually rely on an Internet connection in order to dynamically enhance the content displayed. Several standards compete, but nowadays Hybrid broadband broadcast TV (HbbTV) is the most deployed standard for interactive applications in Europe. HbbTV applications are enhanced web applications and have the control over several elements of the television (*e.g.* the screen).

Interestingly, one researcher attacked not a Smart TV receiver, but an MSTAR DVB-T decoder [24]. After interfacing himself with the decoder on a hardware level, he extracted the firmware and started reversing it. He replaced it with a modified version allowing him to have a live debugger and to call whichever function desired. After finding the function responsible for parsing packets from the DVB-T stream, he fuzzed[1] it and found a stack-based buffer overflow vulnerability which he exploited to root the decoder. The same approach could be used to test for vulnerabilities in the DVB-T decoder of a Smart TV, but no research of this kind has yet been published.

A team of researcher has taken interest in privacy concerns with HbbTV and followed the evolution of those issues in [14–16]. Their findings are that broadcasters are able to and do accurately track users regarding their watching habits. They also found that this data is sent to third party services whithout explicit consent from users.

[1] Fuzzing is an automated software testing method which consists in sending lots of unexpected data as inputs to a target to look for vulnerabilities in a protocol implementation.

For hacking Smart TVs with interactive applications, the root publication that highlighted a lot of problems is due to Herfurt [17] in 2013. After reading the standards related to DVB-T and HbbTV he noticed that the mechanism of interactive application is essentially **broadcasted and non-authenticated remote code execution**.

He describes two kinds of attacks in his paper. The first one is done at the network level or at the DVB-T stream level. The second involves getting a malicious HbbTV application to execute on a television.

- **Fake Analytics**: Operators use online analytics services to monitor the number of people watching a show. By generating lots of fake requests with network proxies, it should be possible to falsify this data and impact marketing decisions about continuation of a certain show or not.
- **Content attacks**: This type of attacks involves modifying the content Smart TVs access. It is possible to modify the HbbTV application they access by providing a crafted digital television stream, or it is possible to provide an attacker content using network man-in-the-middle methods as the majority of applications are fetched over HTTP. Also, it is possible to directly infect the operator's server to serve malicious content. The end goal of this type of attack is to serve a malicious application that will be executed by Smart TVs
- **Fake news tickers**: Once the television executes attacker-controlled JavaScript code, it is possible to display content on the screen, for example a fake news ticker on the bottom of the screen, as often seen on news channels.
- **Cryptocurrency mining**: It is possible to use the computing power of Smart TVs to mine some cryptocurrencies and earn money.
- **Arbitrary video display**: There are HbbTV Application Programming Interfaces (APIs) that allow to display arbitrary videos on the screen in place of the current channel. This could be used to hijack the screen of watchers.
- **Native APIs**: There are some APIs which could allow to gather information about the watching habits of a user, for example accessing the favorite channel list.
- **Network pivotting**: Using XMLHttpRequest objects, a Smart TV could be used to further attack the LAN, for example UPnP and HTTP services.

Essentially, the conclusion of Herfurt is that as HbbTV applications are web applications, the full range of browser attacks could be replicated

on Smart TVs, but he did not create any proof of concepts during his research.

The second founding publication for DVB attacks dates from 2014 [27]. Researchers described an experimental setup to broadcast their own HbbTV applications and described some additional attacks available once a Smart TV executes their application. Their setup allows them to capture, modify and replay a DVB-T stream. It is based on the open source tools VLC [6] for reception and OpenCaster [3] for modification. The hardware needed for receiving and emitting a DVB-T stream can be bought for about 150 € and is easily available online[2]. It is possible to override the legitimate stream with a stronger one, which in practice is easily achievable when the device is next to the emitter. This allows them to broadcast their own modified DVB-T stream that will be interpreted by the Smart TV.

With those results, they performed some experiments and defined the following attacks that can be performed with HbbTV applications:

- **Distributed Denial of Service**: A malicious application can be broadcasted and used to perform a DDoS on a service, without leaving any traces back to the attacker.
- **Unauthenticated Request Forgery**: It is possible to use Smart TVs to interact with websites such that they will leave comments or simulate clicks on ads.
- **Authenticated request forgery**: If a user logged into a service which set a cookie, it is possible to retrieve it using HbbTV. This is because it is possible to broadcast an application in a DVB-T stream and to state the origin of this application without verification. For example it is possible to broadcast an application stating it belongs to a known Universal Resource Locator (URL) (*e.g.* `http://gmail.com`) and to retrieve cookies for this website. This attack is no longer possible due to an update of the specification. Applications transmitted only in a digital television stream have their URL rewritten by televisions in a way that prevents this attack.
- **Intranet Request Forgery**: This attack makes use of the fact that the Smart TV is likely connected to an internal network and can be used to pivot in this network or gather information about available services and open ports. It is very close to the one presented in [17].
- **Phishing/Social Engineering**: Using the screen of the television, it is possible to make the user perform dangerous actions by posing as the television system. For example, it could instruct him to enter secret data (*e.g.* his WiFi password) and retrieve it.

[2] More details on the hardware required will be presented in section 4.1.

– **Exploit distribution**: Due to the time necessary to roll out patches for Smart TVs, it is possible to quickly broadcast an exploit for a specific software (*e.g.* WebKit) or library (*e.g.* ffmpeg) before the patches could be distributed.

The researchers at the origin of this study also mentioned managing to run the Browser Exploitation Framework (BeeF) [1] in an HbbTV application but did not investigate further with it. However, they have demonstrated in depth the security problems that arise when it is possible to run JavaScript code on a Smart TV without user interaction, knowledge nor approval.

As mentioned there are other standards of interactive applications. In 2014, HbbTV was not being rolled out in all Europe, in particular UK still relied on the MHEG-5 specification for its interactive applications. A researcher used an interactive application to access pay-to-watch content on a Smart TV [20]. The channel broadcasted two empty video and audio streams, two video and audio streams containing the actual content and an interactive application that was executed at startup and responsible to check if the user had paid. The researcher exploited the way the channel broadcasted its content by modifying the interactive application. The modified application automatically switches to the proper streams at startup, thus efficiently avoiding the paywall.

In [9], researchers studied the implementation of the same-origin policy in the HbbTV browser of several models of Smart TVs. In the four models studied, they found that one did not implement it correctly. They managed to exploit this vulnerability by successfully performing a port redirection from the LAN to the Internet. This allows, from a broadcasted application, to access internal services of the television and test them for vulnerabilities from the WAN.

Two teams of researchers have successfully implemented the **Exploit distribution** attack presented by [27]. The initial vector is a broadcasted application which is transmitted in a DVB-T stream and that will exploit a local vulnerability on a television. In 2015, Michèle described in a book [23] how to exploit a vulnerability in the media player of the television. In 2017, Scheel described [28] how to exploit a vulnerability in the *Array.prototype.sort* function of Apple WebKit and gain full compromission of the Smart TV. In both cases, the exploit did not require any user interaction and was conducted entirely over the air, installing a rootkit on the Smart TV for further use. Security researchers have taken an interest in Smart TVs because they are heavily deployed and have access to lots of valuable data. Because they affect a wide area and do

not leave traces, attacks on the broadcast interface are critical. With the mechanism of interactive applications which allows to execute remote code without any kind of restriction, attacking the broadband interface of Smart TVs is very powerful. Researchers have shown that having this vector unprotected can lead to serious damages, and there are already examples of full exploitation of Smart TVs over the air. In the end, interactive applications as they have been implemented so far are a major vulnerability on Smart TVs. Until recently, the only mitigation available was the ability to turn off the execution of HbbTV applications in the **Settings** menu of some Smart TVs. In order to improve on that subject, the DVB working group in conjunction with the HbbTV consortium have developed a scheme to secure interactive applications at the stream level, which will be presented in this paper.

1.2 Summary of the contribution

From the literature, the following aims of compromission have been drawn by security researchers:

- Use lots of compromised Smart TVs to mine cryptocurrencies of fake ad clicks;
- Use lots of compromised Smart TVs to attack Internet-facing services and perform distributed attacks;
- Use the Smart TVs peripherals (micro/camera) to spy on users;
- Track users and their watching habits;
- Use Smart TVs as an entry point to attack entreprise networks.

Giving the possibility to make intermediate conclusions about the security of those devices:

- Application run with high privileges (admin/root). Thus exploiting an application essentially allows to take over the entire device.
- Smart TVs run a lot of outdated software and libraries, some of which can be exploited using known vulnerabilities.
- Cryptography is lacking in general. Sensitive data stored by a device like passwords and cookies is not protected.

As these devices may be found in critical infrastructures and most of the time be connected to a network while having access to the RF broadcasted signal, the need for testing their security has become very high. **Since 2015, the Wireless Security Lab of the ANSSI has been developing a plateform for assessing the security of Smart TVs related**

to the RF broadcast channel. In this paper, the main results of the experiments are described. A focus on the standard and recent mitigations of discovered security flaws are given. Issues remaining in the proposed update as well as vulnerabilities are outlined. The contribution of this study are: first, works related to the analysis of the security of Smart TV are summarized. Moreover, the main results related to the broadcast interface of those devices are provided. After listing the internals of a digital television stream, the technical details on its use as an attack vector are given. Then, a focus is made on the update of the norm ETSI TS 102 809 which would have prevented the most critical of those attacks. However, some flaws still present in the current specification are addressed. The experimental setup built at the ANSSI designed to help in Smart TV security study is presented next. This setup has been used to uncover another way to bypass the protection designed in the specification using interactions between the HbbTV specification and the norm ETSI EN 300 468. Finally, the impact of this vulnerability is discussed and several mitigations are proposed.

2 Dissecting a digital television stream

DVB is the set of standards which defines the layers and components of a digital television stream. Depending of the transmission media, the physical layer changes: DVB-C/DVB-C2 define transmission over cable, DVB-S/DVB-S2 define transmission via satellite and DVB-T/DVB-T2 define terrestrial transmission. The basics of a digital television stream are presented in order to provide the necessary knowledge for studying current attacks and protections designed.

2.1 Basics of a Transport Stream

In order to reduce the number of radiofrequencies used by digital television, several channels are multiplexed together in the same stream at a given frequency. Each channel broadcasts several components: a video stream, several audio streams, subtitles, interactive applications... Each of these components is encoded in an Elementary Stream (ES), which are then chunked into 188-byte long packets. Those packets are interleaved together, forming a proper MPEG-2 Transport Stream (TS). In order to reconstruct each of the original ES, the MPEG-2 header of each packet contains a Packet IDentifier (PID) which is unique and constant for each ES.

In addition to data, there are also metadata which are transmitted in a TS. Those metadata can be the number of channels present in a TS, the

compounds of a particular channel, the name of the different channels, the Electronic Program Guide (EPG) which holds the schedule for the next programmes, etc. The MPEG-2 TS and DVB specifications define a certain number of tables to carry those information. Tables are also chunked into 188-byte long packets with their own PID and transmitted in the TS.

Some important tables are:

- **Program Association Table** (PAT): This table has PID 0 and is mandatory. It contains the number of channels present in this multiplex and for each channel the PID of its Program Map Table.
- **Program Map Table** (PMT): This table holds a channel's (called **program** in MPEG-2's terminology and **service** in DVB's terminology) components and associates for each of them a PID.
- **Service Description Table** (SDT): This table holds the information about channel names. Each PMT holds an identifier for the channel, and the SDT maps this identifier to the channel name.
- **Event Information Table** (EIT): This table holds information about the EPG. Each programme described has a name, a description, a starting time and a duration, as well as additional information. This allows a television to parse this information from the DVB stream and display it directly on the screen, usually by pressing the **Guide** or **Info** button of the remote.

Figure 1 highlights how a TS is formed at the Transport Layer level. Modifying a digital television signal thus involves reconstructing each table and ES, modifying their content then re-fragmenting them into a modified TS. In practice, a single TS contains several hundreds of PIDs.

Figure 2 presents the overlook of a DVB stream. Indirections are used a lot: tables point to other tables which point to other tables, etc. An important abstraction is that when a compound (table or ES) is defined by a PMT, its scope will be at most the channel defined by this PMT. Therefore, there is a difference between elements that are global to the multiplex (PAT, SDT, EIT...) and components that are defined for a specific channel.

2.2 Interactive applications: HbbTV

An HbbTV application is essentially a web application that is sent in a broadcasted signal and executed by receivers, that are Smart TVs. In its latest version (2.0.1) [29], HbbTV relies on the Declarative Application

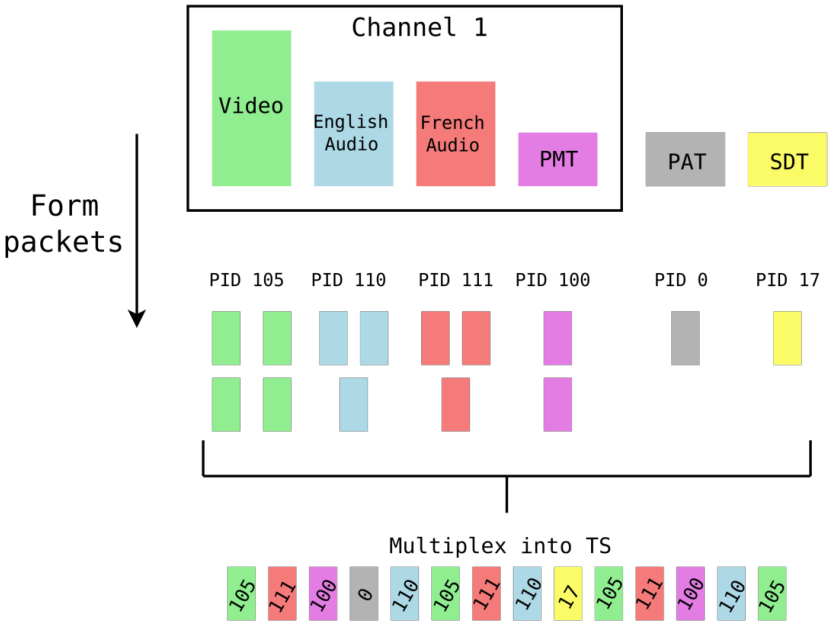

Fig. 1. Construction of a TS

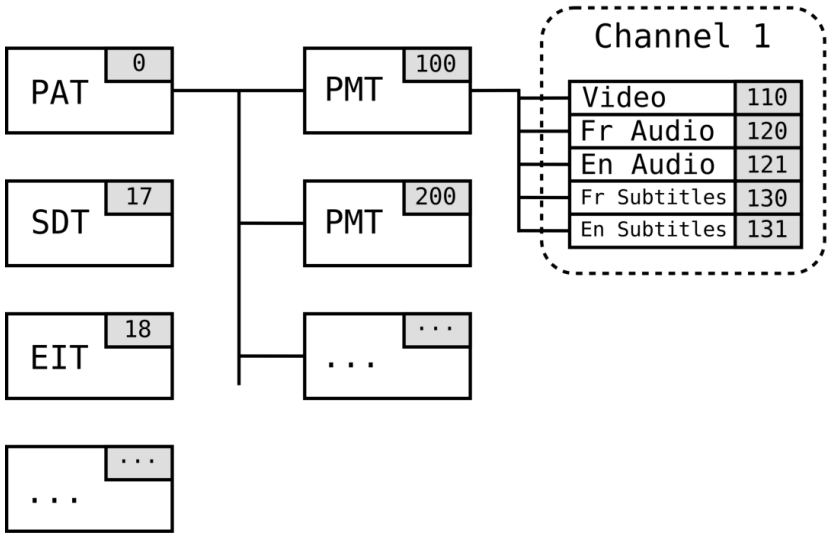

Fig. 2. Architecture of a DVB stream

Environment (DAE) v2.3 [26] released by the Open IPTV Forum (OIPF). This specification, among others, defines some objects that can be retrieved and used from the global context of the applications. For example, it defines the **oipfConfiguration** object which can be used to retrieve information about the **vendor** and **model** of the device. The HbbTV specification integrates the majority of the objects defined in DAE with some exceptions whose implementation is either restricted or left to the choice of the manufacturer. The idea behind this architecture is that the EcmaScript specification and Web APIs are not suited to manipulate concepts related to televisions. Hence it is needed to define additional APIs, like the possibility to switch channel or control the volume of the Smart TV.

	No of bits
application_information_section() {	
table_id	8
section_syntax_indicator	1
reserved_future_use	1
reserved	2
section_length	12
test_application_flag	1
application_type	15
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
reserved_future_use	4
common_descriptors_length	12
for(i=0 ; i<N ; i++) {	
descriptor()	var.
}	
reserved_future_use	4
application_loop_length	12
for(i=0 ; i<N ; i++) {	
application_identifier()	48
application_control_code	8
reserved_future_use	4
application_descriptors_loop_length	12
for(i=0 ; i<N ; i++) {	
descriptor()	var
}	.
}	
CRC_32	32
}	

Fig. 3. Structure of the Application Information Table

In order to execute an interactive application, it must be signalled in the DVB stream. For this purpose, there exists a specific table which is responsible of that: the Application Information Table (AIT) whose schema is detailed in figure 3. If a channel broadcasts an application, its PMT must reference a component whose PID will point to an AIT for this channel. Also, each application defines the way it should be started. It is possible to specify for an application to be autostarted as soon as Smart TVs received it. This is similar to the Web model where scripts are executed automatically when the page has loaded.

An application defines the way it should be retrieved by specifying a list of URIs to use to access it and an entry point. There are two possible transport modes[3]:

- The application is retrieved using HTTP(S) over the broadband interface. In this case, the URI looks like `https://hbbtv.sstic.org/`.
- The application is retrieved from the DVB stream. It is possible to embed a set of files in a specific component: a Digital Storage Media Command and Control (DSM-CC) also called object carousel. In this case, the carousel must be referenced by the PMT and the application must provide a URI such as `dvb://hbbtv.sstic.org/` along with the identifier of the carousel to use.

Fig. 4. Signalling applications in a DVB stream

Figure 4 summarizes the signalling of interactive applications in a DVB stream. The bases of how HbbTV applications are formed, signalled in a digital television stream and retrieved by Smart TVs have been presented. The HbbTV standard also defines the notion of *trusted* and *non-trusted* applications. This distinction is used to restrict some APIs to only trusted applications. In this paper are considered only HbbTV applications properly signalled in a DVB stream, hence dependant from the broadcast interface. It is specified that such applications are trusted by default. There

[3] The specification also defines a third transport mode which is the Common Interface with URI starting with `ci://`, though this possibility has been left aside because it has not been studied nor experimented with.

are other ways to start HbbTV applications (*e.g.* Companion Screen), which would be non-trusted by default, but they will not be presented in this paper.

2.3 Security of DVB

The attacker model considered is one that has complete control over the DVB-T signal and over the network. This is coherent with the previous researches which show that the material required costs about 150 € and that open-source tools exist to record, modify, create and emit a DVB-T signal. It is not necessarily assumed that the television has an Internet connection available, but it must obviously have an antenna for DVB-T.

This attacker is able to redirect all channels to HbbTV applications hosted on a controlled web server, or is able to embed an arbitrary application in an object carousel and to signal it on all channels. This leads to the **Exploit Distribution** scenario presented in section 1.1 and implemented already twice. This comes from the lack of protection in the stream: signalling of interactive applications is left unprotected while it is sensitive data and should be secured accordingly.

More specifically, in order to be protected from this attack, several mitagation steps are provided. Implementation of those steps results in an attacker being unable to execute a malicious interactive application on a Smart TV. An attacker should not be able to:

- Forge or modify the signalling data: it should be protected in integrity;
- Modify on the fly an interactive application retrieved from the Internet. Only HTTPS should be used to fetch an application over broadband, insecure transport modes are to be discarded;
- Forge or modify the contents of a carousel. Carousels should be protected in integrity so that an attacker could not create a valid one.

With a proper implementation of those countermeasures, it is expected that the attacker defined above would not be able to make Smart TVs execute malicious applications. Assuming a correct implementation, all attacks which rely on the execution of an attacker-controlled interactive application would be nullified, which would greatly improve the security of Smart TVs.

3 Evolution of the norm

Following the results of [23], the HbbTV consortium and the DVB working group worked to integrate security for the signalling and data of interactive

applications. Right after the presentation [28] DVB updated its standard ETSI TS 102 809: Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environment [30]. This is the specification which defines the structure of the AIT, it now includes a section about the security of interactive applications.

3.1 Presentation of the protection scheme

The chosen approach to add security in a DVB stream is to include additional messages and tables which will carry protection messages. The update develops two distinct elements: first on how to establish and maintain a set of trusted public keys in a broadcast environment. Second, given a set of trusted public keys, on how to authenticate data related to interactive applications. In order to establish and maintain trust, the standard uses certificates. Based on a root of trust, a certificate chain is built for each service (*i.e.* channel; possibly, each service can have a different root of trust). Each certificate is signed by its parent (self-signed if this is the trust anchor) and references it to enable verification of the chain. At the stream level, a certificate chain belongs to a channel and is referenced by the PMT as one of the channel's component[4]. The last certificate of the chain holds the key that is used to authenticate the data to protect.

In order to prevent replay attacks, certificates have an interval during which they are valid, implemented with a **notBefore** and a **notAfter** field. The root certificate can also be changed. For this case, each root certificate advertises the new public key that will be used by the successor certificate. In the end, given a trust anchor, this scheme allows to derive for each channel a different certificate chain for each channel in a multiplex.

There are two possible root of trusts: manager certificates and coordinating entities. A coordinating entity is an external entity whose certificate is already present in a receiver. A manager certificate is a certificate that has been defined as being the legitimate root of trust for one or more services. In order to become a manager certificate, a valid certificate chain must be broadcasted regularly during a probation period.

Given a channel whose certificate chain is valid and trusted, this means that there is a trusted public key that originates from the broadcaster, which can be used to authenticate data. The extension defines as *protectable*

[4] Actually, it can also be carried by the component it is meant to protect but for simplicity sake this case is not developed, because the same PID would reference two very different messages and although valid, this has been purposely left aside.

streams the components of a channel that hold either an AIT or an object carousel. The extension specifies that the broadcaster must hash then sign this data, and send those authentication messages in another table. Upon reception of the authentication message, the Smart TVs must store the hashes in a queue. When there is a match between a stored hash and the hash of an incoming AIT or object carousel, this means that this data originates from the broadcaster and it can be safely processed.

3.2 Security of the extension

At this point, no experiments could be performed to validate or reject the attacks presented in this section because no device implementing this specification was available. Indeed, after going through the specification, there are some possibilities which are left open and which could allow an attacker to bypass the security in place. For now, the previous version of the specification will be called **legacy version**, where applications are **unsigned** while the one described above will be labelled **secure version**, where applications are **signed**.

First and foremost, implementing the security is a choice left to manufacturers. In the case of a receiver that would support both versions, it is possible to perform a **downgrade attack** by stripping all protection messages and removing every element that has been added to this end. This way, the resulting stream stays valid with regards to the legacy version, hence can be freely modified by the attacker. This would result in the same security issues as before and would not improve security of Smart TVs. In order to mitigate that, the **protection of the interactive application signalling and data must be enforced by receivers**. In the rest of this section, it is assumed that the specification is perfectly implemented and that it is not possible to use the legacy mode anymore.

Regarding the transport protocol used to fetch an application, this is specified in the HbbTV specification and not in the ETSI TS 102 809. This means that there have been no changes compared to previously: applications can still be fetched using insecure transport protocols such as HTTP. In order to prevent an attacker from modifying an HbbTV application after a successful Man-in-the-Middle, **insecure transport possibilities must be removed and only HTTPS should be allowed for accessing an application over broadband**. The HbbTV specification [29] already defines in section 11.2 TLS parameters to use when fetching an application using HTTPS.

It is possible to abuse the lifecycle of a certificate chain. In particular, for services (*i.e.* channels) that have not yet been visited and which

do not depend on a coordinating entity, it is possible to define a self-signed certificate as the manager certificate (*i.e.* the root of trust) for this service. More precisely, if a new service advertises a certificate chain that is coherent, stable and present for the length of a probation period then this certificate chain is considered trusted and the top-level certificate of the chain becomes the manager certificate for this service. This means that if the channel has not been visited before, an attacker can set its own certificate chain which will eventually become trusted at the end of the probation period. The specification sets the probation period in this case to 300 seconds, which can be extended at will by manufacturers. Also according to the specification, the last certificate of a chain should be renewed every few months, meaning that the probation period can not exceed this value (else, the Smart TV would never trust any certificate chain which does not come from a coordinating entity). Thus, when a user goes to a new channel never visited before an attacker can forge a certificate chain and will eventually manage to make the Smart TV interpret arbitrary applications.

The previous attack can be extended, by forcing the Smart TV to interpret a channel as a new one. From a Smart TV's point of view, what identifies a channel is not what it displays on the screen, but its internal identifier: the **service_id**. With the chosen attacker model, it is possible to create a **perfect-looking copy** of a given Transport Stream by changing every internal identifiers but keeping the relations between tables and the content displayed. In this case, it is likely that the television would interpret this TS as an entirely new multiplex, while the user would notice nothing. Also, as coordinating entities are defined for a set of services, those would no longer be the reference root of trust because the Smart TV would see yet unknown services. As a result, all services in this TS would appear as new, hence leveraging the possibility to perform the attack described above for all channels.

The ability to set an self-signed certificate as a manager certificate without any other verification than waiting the duration of the probation period is nothing but a backdoor which nullifies the security brought by the extension and allows an attacker to circumvent the mechanisms in place. Hence, only coordinating entity should be kept as root of trust. In the case of a new service appearing, it should not be possible to define the root of trust for this service as it is currently the case with manager certificates.

As a result, the security measures defined by DVB to protect interactive services and data are considered insufficient to provide a decent security

level. In particular, it has been shown that retrocompatibility of the scheme for authenticating interactive application signalling and data could be used by an attacker to strip the security from the elements being protected. In addition, it has been shown how the lifecycle of trust for a service could be used by an attacker to make a Smart TV trust its own certificates using probation periods for new channels. In order to correct those problems, the following countermeasures should be implemented:

- Authentication of interactive applications and data must be enforced. Unauthenticated content should by dropped by receivers.
- Applications fetched over broadband must use HTTPS and not HTTP. The latter does not prevent the studied attacker from providing malicious applications that will be executed by Smart TVs.
- The root of trust for each channel must be provided by an external coordinating entity. Manager certificates which can be forged by an attacker and which are trusted after a probation period must be removed from the specification.

3.3 Where are we now ?

Let's consider that the modifications depicted above have been integrated to the specification and are perfectly implemented by receivers. This means that at every moment, there is at most one trusted public key per channel. The public keys are changed every few weeks to few months according to the specification and are part of a certificate chain which is valid and goes back to the public key of an external coordinating entity. This entity serves as a trusted third party and is responsible for maintaining the trusted set of keys in a multiplex. Channels whose public key has been signed must use their private key to sign their AIT(s) and object carousels. It is assumed that Smart TVs implement perfectly the specification and verifications involved.

With those modifications, an attacker is no longer able to create valid signalling data which would make a Smart TV get and execute an application in control of the attacker. An attacker can no longer modify the application directly because object carousels are also protected and by hypothesis Smart TVs do not use insecure transport modes to fetch applications. As a result, an attacker in full control of the network and of the DVB-T stream would still be unable to make Smart TV execute arbitrary applications. With the improvements of the specification described in this paper, the majority of known DVB attacks described in the state of the art are rendered impossible to conduct.

Security researchers have focused almost exclusively on interactive applications for attacking the broadcast interface of Smart TVs. However, there is much more data that travels in a digital television stream, which are currently left unprotected. With further scrutiny from the security community, it is likely that more problems will be found in the future. In the rest of the paper, some experiments conducted on Smart TVs are presented. In particular, it demonstrates a **new kind of attack** on a DVB stream which enables an attacker to run an HbbTV application, even with all the described protections. The impacts of this vulnerability and countermeasures are discussed in order to improve the security of Smart TVs.

4 Experimental results

In the framework of our research activities, some experiences have been performed on Smart TVs. While part of them involved replicating previous results present in the literature and following the work of Yannick Darriet at ANSSI, a tool for assessing the security of the broadcast interface of Smart TVs has been developed. After presenting this program and its capabilities, another vulnerability regarding signed HbbTV applications will be presented along with efficient countermeasures.

Experimenting with the broadcast interface of Smart TVs involves emitting a radio signal over protected frequency bands. For this reason, experiments were run in an isolated environment.

4.1 Test environment and open-source tools

The experimental setup relies on both software and hardware components. The one reproduced in the laboratory is similar to those presented in [9, 23, 27].

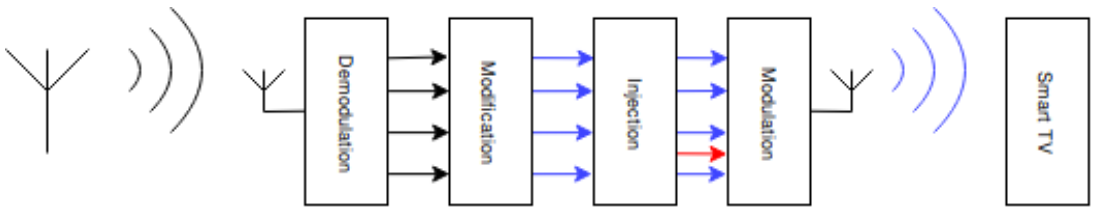

Fig. 5. Experimental setup

Figure 5 presents the architecture used to perform security tests on Smart TVs. Conceptually, this is no different than a MitM attack on a network interface, except that the protocols and the physical layers are not the same. The role and tools used to implement the different components of this setup are detailed.

Demodulation - Capturing a DVB-T stream is very well documented on the Internet. A DVB-T receptor based on the RTL2832 chip has been used, whose cost is ~15 €. On the software side, the program **tzap** [5] is used to tune the receiver to the correct frequency while **dvbsnoop** [2] is used to save it to a file. The result is a `.ts` file which contains the entire Transport Stream.

Modification, Injection - The open-source tool suite OpenCaster [3] from Avalpa has been used to modify and add content to the TS. Each tool is specialised, but they usually take an input `.ts` file and output a modified `.ts` file. From this suite, **oc-update** has been used to generate an object carousel from a directory. The tool **tsmodder** can be used to modify all packets with the given PIDs in a TS and replace them with another content. This can be used to modify an existing AIT or carousel inside a Transport Stream. The tool **tscbrmuxer** can be used to add new packets with non-existant PIDs to a TS. This can be used for example to add a completely new AIT to a channel. Also, the OpenCaster suite provides a library to generate `.ts` files from DVB tables. Those can then be used by the described tools to be added to the original TS.

Modulation - The modulation takes a transport stream as input and emits it in direction of a Smart TV. There are two approaches which can be used to this end: either use specialised hardware or software-defined radio. For this study, the DVB-T emitter UT-100C from HiDES has been used. It is a cheap emitter (169 $) which was deemed suited for a laboratory environment. Further possibilities for emission are studied and compared in [23].

Smart TVs - Several models of Smart TVs, coming from leading manufacturers have been used as test subjects in the experiments. Three different models of Smart TVs were available, dating from 2014, 2015 and 2017. As a result, none of them implemented the protection of interactive application signalling and data.

In order to understand the DVB standards and to see how the tables are used in practice, the program **dvbsnoop** allows to dump a specific table in an understandable format. This is very helpful to understand which modifications should be done to specific tables in order to get the desired result.

Using this setup, it is possible to start injecting malicious applications into Smart TV and observe their behavior. In order to iterate among experiments, two cases are to be considered: if the application is served over broadband or over broadcast.

- With a broadcasted application, the object carousel must be changed each time the application changes. This means stopping the emission, repacking the modified application into the transport stream, rebooting the TV for resetting the application executed, start the emission with the new transport stream and navigate to the channel infected.
- With an application served over broadband, there is no need to modify the transport stream. With the web server serving a malicious application, it can be directly changed and the new application will be served. The Smart TV must still be rebooted in order to load the new HbbTV application.

The drawback of this way of doing is that televisions are not meant to be turned off and on efficiently: loading a modified application takes time in the span of tens of seconds (depends of the model under test). This unnecessary waiting time greatly limits the number of experiments that can be performed. An additional problem with Smart TVs is the absence of debugging tools: the JavaScript console is not visible from the HbbTV browser. The solution implemented to palliate these problems is presented.

4.2 Using a JavaScript console for introspection

The DAE and HbbTV specifications have many elements and APIs, being able to test them efficiently is valuable for studying the security of the HbbTV environment of Smart TVs. To this end, a particular HbbTV application was designed, which allows to iterate among experiments with a much lower overhead than rebooting a Smart TV.

A reverse console has been implemented in order to ease the work of introspection and debug of a Smart TV. Compared to the original workflow which consists in reloading a full HbbTV application for each experiment, this console uses a slave HbbTV application that is initially loaded on televisions. This application communicates with a web server and when a button of the remote is pushed, fetches new code to execute it inside the HbbTV browser.

Figure 6 shows the look of the console running on a Smart TV. The screen of the television is used to display information, including caught exceptions. The full application relies on three components:

– **Slave** - An HbbTV application to be signalled in a Transport Stream, that will be executed by televisions. This handles remote control events and communicates with the server to retrieve new code to execute. It displays on the screen the results of an execution and posts them on the server;
– **Master** - A client web application to be executed on the computer of the operator. It allows to set new code to be executed and displays the result of the execution that was returned by Slaves;
– **Server** - A web server application that works as intermediary between the Master and the Slaves.

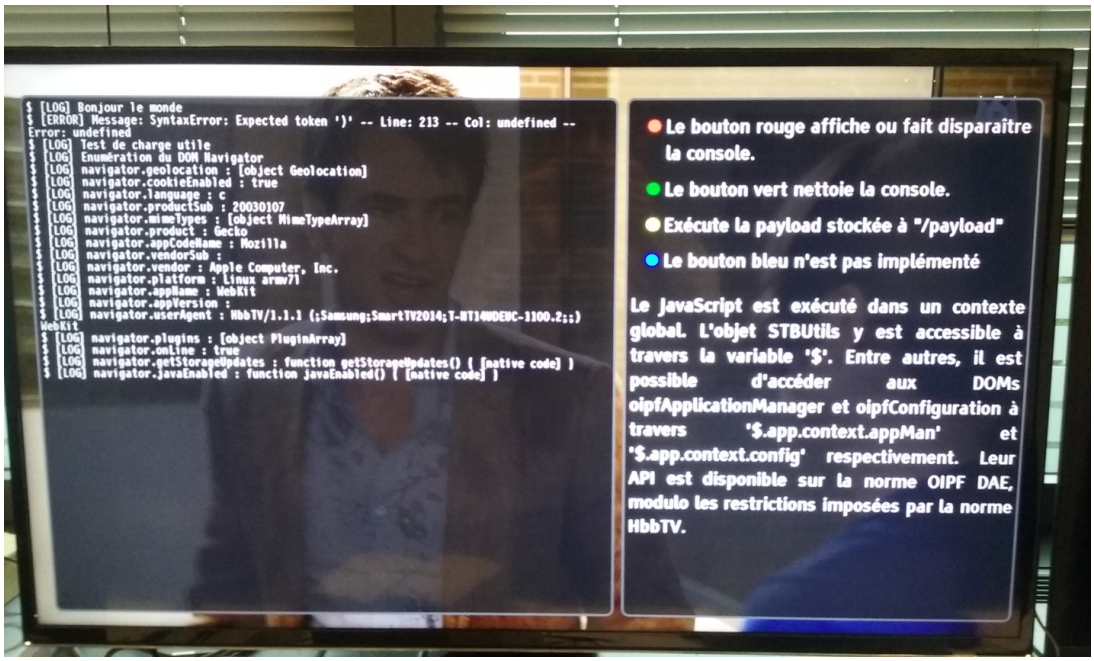

Fig. 6. Console running on a Smart TV; there is no international version of this application

This architecture, depicted in figure 7, simplifies tests on the HbbTV environment of Smart TVs, while being simple to operate. Compared to rebooting Smart TVs between each test, it provides the following advantages:

– Rather than carefully designing HbbTV applications and experiments before attempting to execute them, this tool allows a direct interaction with Smart TVs;
– Originally, executing an HbbTV application on a Smart TV yields only a binary response: either it works or it does not. A Console API

has been implemented into the slave application, which serves as a feedback loop for the operator. This way, it is possible to have richer information and much better debugging capabilities;
– It can run on several devices at the same time, which enables parallelizing tests on several Smart TVs.
– It brings the possibility to execute existing .js files in the browser of the television without having to embed them in an HbbTV application.

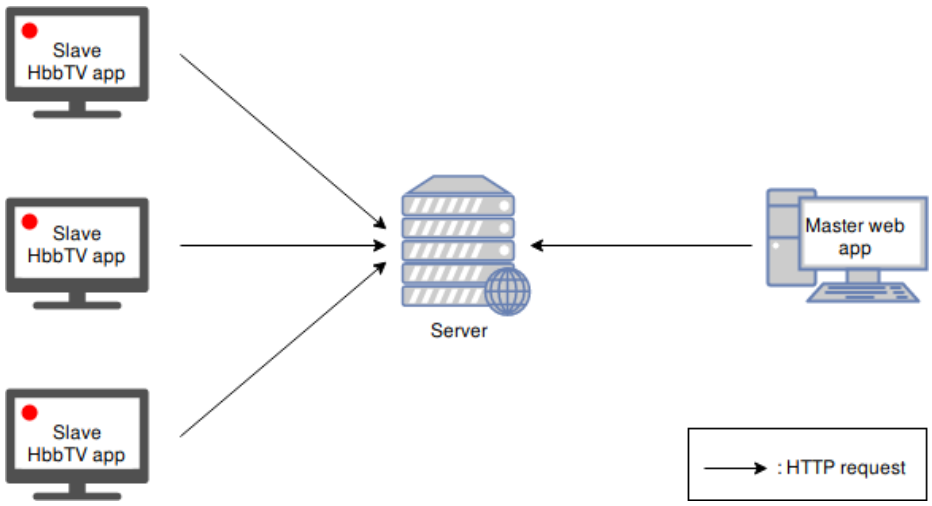

Fig. 7. Architecture of the console application

This tool has been used to perform extensive introspection about the HbbTV browser of the three Smart TVs at hand. In particular, this allowed to observe the behavior of the HbbTV APIs and to understand which functions were implemented or not. Being able to gather this sort of information is valuable for a researcher as it permits to gain insight about the inner working of the HbbTV browser of different platforms. This way, one can quickly estimate the capabilities of a specific Smart TV and its attack surface.

A short demonstration is available [11]. It presents the capabilities of the console and a re-implementation of the **Screen Hijacking** attack. There are regularly new vulnerabilities discovered which impact web browsers and lots of exploits are readily available on the Internet. As the console allows to take plain JavaScript files and to execute them remotely, it can be used to quickly test for batches of known vulnerabilities in the HbbTV browser. This could be used to automatically detect if a Smart TV needs to be patched in light of a new vulnerability.

The JavaScript console is a powerful tool which is meant to help security researchers in their study of the security of Smart TVs. It provides automation of tests and parallelization among devices which allows to quickly gain valuable information about the HbbTV browser of several Smart TVs. This has been used in order to study the models available and to uncover several vulnerabilities which have been coordinatedly disclosed to affected vendors. In particular, this console was used to uncover a vulnerability which allows to bypass the protection designed in the specification ETSI TS 102 809.

4.3 Exploiting a signed application with a DVB stream

As stated in section 4.1, none of the Smart TVs available for the tests do implement the protection scheme described in ETSI TS 102 809. By hypothesis, it is considered that those protections are perfectly implemented. In addition, it is assumed that the shortcomings of this specification have been mitigated according to the countermeasures developed in section 3.2. As a result, it is not possible to modify the signalling and data of interactive applications. This section describes how it is possible to hijack a legitimate HbbTV application using unprotected elements in the stream.

The DAE defines the **Channel** and **Programme** objects, which are also included in the HbbTV specification. As their names suggests it, those represent metadata about an actual channel and programme.

```
// The factory is available in the global context
var videoBroadcast = oipfObjectFactory.createVideoBroadcastObject();
var channelList = videoBroadcast.getChannelConfig().channelList;

// Display the name of each known channel
for (var i = 0; i < channelList.length; i++) {
    Console.log("Channel " + i + " has name " + channelList.item(i).
        name);
}

// Tune the video broadcast to the current channel
videoBroadcast.bindToCurrentChannel();

// Iterate through the programmes of this channel
for (var i = 0; i < videoBroadcast.programmes.length; i++) {
    var programme = videoBroadcast.programmes.item(i);

    // Display some information about the programme
    Console.log("Programme " + programme.name);
    Console.log("Short description " + programme.description);
    Console.log("Long description " + programme.longDescription);
}
```

If the above code is fed to the console application, all the channel names and programmes will be displayed on the screen. Apart from the *Console.log* calls which are defined by the application, all the other objects, properties and methods are APIs defined by DAE and are available on all tested models.

The problem here is that those APIs use elements that come straight from the DVB stream. More specifically, information about the channels like their name is taken from the SDT, while information about the programmes are taken from the EIT. Those elements are not protected in integrity inside the DVB stream, hence can be freely modified by an attacker.

Table 1 presents the global outlook of the SDT. It is comprised of several general properties, then a service (*i.e.* channel) loop which defines each service, then finally a descriptor loop inside each defined service. The name is defined in the descriptors of a service, either using a **service_descriptor** or a **multilingual_service_name_descriptor**.

Tables 2 and 3 show the structure of the descriptors as they are defined in ETSI EN 300 468 [13] from the DVB standards. Using either descriptor, the property **service_name** is used by Smart TVs to get channel names. Those will be used as values for the **name** property of the **Channel** class inside the HbbTV browser. The length of a descriptor cannot exceed 257 bytes, hence the longest name that can be inserted in a **service_descriptor** is 252-byte long, assuming an empty service provider name. The longest name that can be input in a **multilingual_service_name_descriptor** is 250 bytes-long, assuming a single language and an empty service provider name. In practice, studied Smart TVs truncated those values to 70 to 100 bytes, depending on the model. Also, on two out of the three models available, it was observed that the names were parsed from the SDT only during a channel scan. After that, values stored in persistent memory were retrieved by the HbbTV application, and not the ones being broadcasted.

Table 4 depicts the general structure of the EIT. It contains several general properties, then a loop of events (*i.e.* programmes), each event containing a descriptor loop. From the **Programme** class, the properties **name** and **description** are retrieved from the **short_event_descriptor** and the property **longDescription** is retrieved from the **extended_event_descriptor**.

Tables 5 and 6 show the structure of the EIT descriptors that can be used to inject attacker-controlled data inside an HbbTV environment through the Programme class. In the **short_event_descriptor**, the

Syntax of the SDT	No of bits
service_description_section() {	
various properties...	88
for (i=0;i<services_loop_N;i++) {	
service properties...	40
for (j=0;j<descriptor_loop_N;j++) {	
descriptor()	Variable
}	
}	
CRC	32
}	

Table 1. Simplified structure of the SDT

Syntax of the descriptor	No of bits
service_descriptor() {	
descriptor_tag	8
descriptor_length	8
service_type	8
service_provider_name_length	8
for (i=0;i<name_length;i++) {	
char	8
}	
service_name_length	8
for (i=0;i<name_length;i++) {	
char	8
}	
}	

Table 2. Structure of SDT's service descriptor

Syntax of the descriptor	No of bits
multilingual_service_name_descriptor() {	
descriptor_tag	8
descriptor_length	8
for (i=0;i<name_loop_N;i++) {	
ISO_639_language_code	24
service_provider_name_length	8
for (i=0;i<name_length;i++) {	
char	8
}	
service_name_length	8
for (i=0;i<name_length;i++) {	
char	8
}	
}	
}	

Table 3. Structure of SDT's multilingual service name descriptor

event_name element is used as the **name** property and the **text** element is used as the **description** property. The **longDescription** property comes from the **text** element of the **extended_event_descriptor**.

Syntax of the EIT	No of bits
event_information_section() {	
various properties...	112
for (i=0;i<event_loop_N;i++) {	
event_id	16
start_time	40
duration	24
running_status	3
free_CA_mode	1
descriptor_loop_length	12
for (j=0;j<descriptor_loop_N;j++) {	
descriptor()	Variable
}	
}	
CRC	32
}	

Table 4. Simplified structure of the EIT

Table 7 summarizes the properties of the classes Channel and Programme which are populated from the DVB stream. Several elements are mere identifiers limited in sizes, but textual fields are more interesting as they allow for tens to hundreds of bytes. Considering a legitimate (hence signed) application that would make use of those properties, this opens the path for injections that come from unprotected elements in the stream.

In particular, the case of an application that would dynamically display the channel list or the EPG has been considered. This means that those elements were retrieved from the defined APIs and included in an HTML context. Such an application has been developed as a proof of concept of a vulnerable HbbTV application. It was successfull tested that it was possible to redirect the television to another application by broadcasting malicious channel names and programme names and descriptions. Those tests validated the injection vectors described.

```
<img src="1" onerror="location.href=&quot;https://evil.tv/&quot;" />
```

When set as a channel name, event name, short or long description, the above excerpt of HTML effectively redirects all tested models on the HbbTV application hosted at `https://evil.tv/`. Even considering

Syntax of the descriptor	No of bits
short_event_descriptor() {	
descriptor_tag	8
descriptor_length	8
ISO_639_language_code	24
event_name_length	8
for (i=0;i<name_length;i++) {	
event_name_char	8
}	
text_length	8
for (i=0;i<text_length;i++) {	
text_char	8
}	
}	

Table 5. Structure of EIT's short event descriptor

Syntax of the descriptor	No of bits
extended_event_descriptor() {	
descriptor_tag	8
descriptor_length	8
descriptor_number	4
last_descriptor_number	4
ISO_639_language_code	24
length_of_items	8
for (i=0;i<items_length;i++) {	
item_description_length	8
for (i=0;i<description_length;i++) {	
item_description_char	8
}	
item_length	8
for (i=0;i<item_length;i++) {	
item_char	8
}	
}	
text_length	8
for (i=0;i<text_length;i++) {	
text_char	8
}	
}	

Table 6. Structure of EIT's extended event descriptor

the protections in place which prevent an attacker from modifying the signalling and data of interactive applications, tests show that it is possible to get an unsigned HbbTV application running on a Smart TV. Compared to existing attacks, the prerequisites are more important because it is no longer possible to broadcast malicious carousel objects, hence the television must be connected to the Internet. Most of all, this proof-of-concept shows that the existing protections in addition to those defined in 3.2 are not sufficient to be completely protected from an attacker in control of the DVB stream.

Class	Property	DVB element	Max. size
Channel	channelType	—	—
	dsd	—	—
	idType-nid	—	—
	onid	SDT>original_network_id	16 bits
	tsid	SDT>transport_stream_id	16 bits
	sid	SDT>service_loop>service_id	16 bits
	name	SDT>service_loop>descriptor_loop> service_descriptor OR SDT>service_loop>descriptor_loop> multilingual_service_name_descriptor	250/252 bytes
	majorChannel	—	—
	terminalChannel	—	—
Programme	name	EIT>event_loop>descriptor_loop> short_event_descriptor	250 bytes
	programmeID	EIT>event_loop>event_id	16 bits
	programmeIDType	—	—
	description	EIT>event_loop>descriptor_loop> short_event_descriptor	250 bytes
	longDescription	EIT>event_loop>descriptor_loop> extended_event_descriptor	249 bytes
	startTime	EIT>event_loop>start_time	40 bits
	duration	EIT>event_loop>duration	24 bits
	channelID	— (same as Channel.ccid)	—
	parentalRatings	EIT>event_loop>descriptor_loop> parental_ratings_descriptor	8 bits

Table 7. Mapping between DAE classes and DVB tables

4.4 Impact and countermeasures

This vulnerability has no immediate impact at the time of the writing. That's because to the knowledge of the authors, there is currently no model of Smart TV which implements the protection system defined in

ETSI TS 102 809. As a result, all Smart TVs in Europe are still vulnerable to the **Exploit distribution** attack. Yet, this shows another shortcoming of this specification which does not efficiently prevent Smart TVs from executing malicious HbbTV applications.

This vulnerability opens an interesting question about the responsability of the several entities involved. Theoretically, neither the DVB standards are vulnerable nor is the HbbTV specification. When taken independently, the DVB standards (modulo the improvements) are secure because they efficiently prevent an attacker from getting to the execution of an interactive application on a Smart TV. The underlying assumption is that other elements in the stream can do no harm even if controlled by an attacker. When taken independently, the HbbTV specification is secure because it assumes the security of the underlying broadcast network, which is why applications which depend from the broadcast are considered trusted: HbbTV (and DAE) is independent from the broadcast and its APIs as well. However, when taken together problems arise because security assumptions that both standards did are no longer valid.

With this in mind, there are different entities that can take efficient countermeasures against this attack:

1. The DVB working group could extend the protection designed in ETSI TS 102 809 to all elements in a Transport Stream as the TS being unauthenticated is the root of all DVB attacks;
2. The HbbTV consortium could remove vulnerable APIs from the specification, such that no untrusted data can be used from within an HbbTV application;
3. Smart TV manufacturers could escape all elements that they get from the digital television stream before handling them, either internally or passing them to the HbbTV browser;
4. Application developers could escape all elements that are retrieved from the stream. However, this would require a precise list of APIs marked as dangerous by either the manufacturer or the HbbTV specification;
5. It is also possible to maintain the *status quo* and keep the current countermeasure which consists in disabling the execution of HbbTV applications from the Smart TV menu. In that case, security falls into the hands of individual users.

Each of those countermeasures would prevent the attack presented here. However, the interest among security researchers in attacking Smart TVs with the broadcast interface has just started. Though this study is the first to depict attacking this interface with other elements than the

AIT or the object carousel, this merely means that the vast majority of the DVB standards have not met scrutiny from security researchers, not that they are inherently secure.

On the contrary, the full range of DVB attacks are made possible by the fact that there is no authentication of the stream. As a result, authenticating every element inside a DVB stream would not only protect against this attack, but also prevent all those that have not yet been discovered and rely on this fact.

5 Synthesis

Smart TVs are becoming more and more deployed today. With their numerous communication interfaces and capabilities, they are interesting targets for an attacker. Be it for accessing a television's internal data and sensors or to use Smart TVs as an entry into local networks, Smart TVs must be secured against those attacks.

Various security researchers have studied the security of those devices, the global conclusion being that it is not satisfactory for such critical devices. A specific attack vector has proven very problematic for Smart TVs: the digital television signal. Radio-based attacks are extremely problematic because they have the ability to affect many devices in a specific area while being untraceable. On Smart TVs, these attacks can be used to remotely execute interactive applications without need for user interaction. This has led to multiple attacks being designed for this vector, which in two cases ended up with the full compromise of a device using nothing but a crafted radio signal. For years, the only countermeasure to this vulnerability was to manually disable the execution of interactive applications on Smart TVs.

On February, 2017 the DVB working group published a new version of the specification ETSI TS 102 809 with the objective of preventing an attacker in control of the DVB-T signal to get Smart TVs to execute malicious interactive applications. This paper is the first to present and discuss the security schemes designed. As a result, several problems with the current specification have been detailed, namely that the implementation of this norm must be enforced without possibility for legacy behavior. In addition, it should explicitely restrict interactive applications from being loaded from insecure transport modes such as HTTP. Finally, there is currently a safeguard procedure in the establishment of trust which would allow an attacker to bypass the security brought by the extension, simply with the broadcasting of a self-signed certificate: this safeguard

must be removed and trust be managed by one or several trusted entities specifically for this purpose.

Some experiments have been conducted on three models of Smart TV. The test environment has been detailed for both the software and hardware side. A tool meant to ease the work of security researchers and improve their efficiency to study HbbTV-based attacks has been presented, following the demonstration done in [11]. A practical use of this tool has been provided as it helped discover a new kind of attack where it is possible to bypass the security brought in ETSI TS 102 809 by injecting malicious content in channel names and programme metadata. This proof-of-concept shows that DVB is a very powerful attack vector, which is not limited to interactive applications.

As software-defined radios become more powerful while cheaper and with easily available specialised emitters for DVB-T, this shows that the absence of authentication in a DVB stream poses huge security risks. As a result, it is suggested to extend the security mechanisms designed in ETSI TS 102 809 and the improvements presented in this paper to the entirety of the DVB stream, instead of just AITs and object carousels.

The authors would like to thank Yannick Darriet and the Wireless Security Lab for their major role in this study. Also, the authors would like to thank the INSA de Rennes for their role in making this research possible.

A Appendix: Previous work of Smart TVs

Several studies on the security of Smart TVs have been published so far. The following have been determined to provide an accurate overview of the landscape of attacks so far. Researchers have used various approaches to get those results, ranging from black-box to a full white-box approach. It is important to note that there are online communities committed in reversing firmwares and rooting Smart TVs. The Samygo [4] forum provides valuable information for rooting Samsung models. For example, this forum provides modified firmwares which root the TV, or applications to be installed with a developer account that will escalate privileges using a local vulnerability.

In [7], an undisclosed vulnerability was used to access the internal data of a Samsung Smart TV along with contents of a connected USB flash drives. Also, the researchers showed that it was possible to connect a software remote to the television, hence to further exploit the television by installing malicious applications. Though not disclosed, it appears in

the presentation that the entry point is a vulnerable service listening on the local network.

After reversing the firmware of a Samsung television, authors of [21] found several vulnerabilities in the way the application store fetched and installed new applications. They used this vector to install a rootkit which would listen and record every sound around to send it remotely. They included a fake-off mode to prevent the Smart TV from being turned off by remote control.

Smart TVs have a tendancy of running outdated software, which may contain known vulnerabilities. In [10] the Smart TV studied was vulnerable to CVE-2012-5958 in *libupnp*. It was possible to exploit this with crafted Universal Plug and Play (UPnP) packets and to take control of the device using a network access. The same approach has been used in [22] where researchers have identified the version of the *ffmpeg* library and developed an exploit using a known vulnerability for this version. In the end, rooting the TV consisted in playing a file either from a local storage such as a USB flash drive or from a media server.

In [18] it was put into light that LG Smart TVs sent viewing information and USB filenames back to LG's server over an Hypertext Transfer Protocol (HTTP) connection. This poses privacy and confidentiality issues as those information are open for man-in-the-middle attacks.

In [8] it was shown that a specific model of Philips Smart TV left open a WiFi Direct access point with default credentials. After connection, researchers further found a vlunerability in a service listening on the network and successfully took over the television.

Smart TVs are also susceptible of being ransomed. In particular, Android TVs are at risk because there already exist numerous strands of ransomware for the Android OS. This is the observation made independently in [12, 31] where researchers showed that a known ransomware for mobiles worked equally well on some Android TVs.

Samsung is maintaining its own operating system, Tizen, to be used in its phones and Smart TVs. A researcher audited the code and found 40 vulnerabilities [25] in Tizen. Though the entire list is unknown, those presented were heap overflow and the researcher created an exploit for one of them, which resulted in a TV crash. However, it is suggested that with more efforts, creating a reliable exploit for those vulnerabilities is possible.

In the line of operating system flaws, two security researchers presented vulnerabilities found in webOS [19], which is in use by LG's Smart TVs. They found local privilege escalations which could be used by applications, including a read and write access to the physical memory (*/dev/mem*).

References

1. BeeF. https://beefproject.com/.
2. Dvbsnoop. http://dvbsnoop.sourceforge.net/.
3. OpenCaster. https://github.com/aventuri/opencaster.
4. SamyGO. http://www.samygo.tv.
5. Tzap. https://www.linuxtv.org/wiki/index.php/Zap.
6. VLC. https://www.videolan.org/.
7. Luigi Auriemma and Donato Ferrante. Revuln - The TV is watching you, 2012. https://vimeo.com/55174958.
8. Luigi Auriemma and Donato Ferrante. Revuln - Having fun via WiFi with Philips SmartTV, 2014. https://vimeo.com/90138302.
9. Yann Bachy, Vincent Nicomette, Eric Alata, Mohamed Kaâniche, Jean-Christophe Courrege, and Lukjanenko. Protocole HbbTV et sécurité: quelques expérimentations. In *Symposium sur la sécurité des technologies de l'information et des communications*, 2015. https://www.sstic.org/2015/presentation/protocole_hbbtv_et_securite.
10. F Basse. Sécurité des ordivisions : exploitation de CVE-2012-5958. In *23rd USENIX Security Symposium (USENIX Security 14)*. SSTIC2014, 2014. https://www.sstic.org/media/SSTIC2014/SSTIC-actes/securite_des_ordivisions/SSTIC2014-Article-securite_des_ordivisions-basse.pdf.
11. Tristan Claverie. Sécurité des Télévisions Connectées, 2017. https://static.sstic.org/rumps2017/SSTIC_2017-06-08_P11_RUMPS_05.mp4.
12. Echo Duan. FLocker Mobile Ransomware Crosses to Smart TV, 2015. http://blog.trendmicro.com/trendlabs-security-intelligence/flocker-ransomware-crosses-smart-tv/.
13. ETSI EN. 300 468 V1.15.1. *Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems*, 2016-03. http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.15.01_60/en_300468v011501p.pdf.
14. Marco Ghiglieri. I Know What You Watched Last Sunday - A New Survey Of Privacy In HbbTV. Workshop Web 2.0 Security & Privacy 2014 in conjunction with the IEEE Symposium on Security and Privacy, May 2014.
15. Marco Ghiglieri and Erik Tews. A privacy protection system for hbbtv in smart tvs. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*. IEEE, 2014. https://www.sit.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_SIT/Publications/ghiglieri_aprivacyprotectionsystemhbbtv.pdf.
16. Marco Ghiglieri and Michael Waidner. HbbTV Security and Privacy: Issues and Challenges. *IEEE Security & Privacy*, vol. 14, no.(IEEE 14/3):pp. 61–67, 2016.
17. Martin Herfurt. Security issues with Hybrid Broadcast Broadband TV. 30'th Chaos Computer Convention, December 2013, 2013.
18. Jason Huntley. LG Smart TVs logging USB filenames and viewing info to LG servers, 2013. http://doctorbeet.blogspot.fr/2013/11/lg-smart-tvs-logging-usb-filenames-and.html\#comment-form.
19. Lee JongHo and Kim Mingeun. Are you watching TV now ? Is it real ?, 2017. https://www.youtube.com/watch?v=-aQbkQWmx-0.
20. Adam Laurie. Old skewl hacking: Porn free!, 2014. https://www.youtube.com/watch?v=1RPmpJi3VRM.
21. SeungJin Lee. SmartTV Security. In *CanSecWest*, 2013. https://cansecwest.com/slides/2013/SmartTVSecurity.pdf.

22. Benjamin Michéle and Andrew Karpow. Watch and be watched: Compromising all Smart TV generations. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*. IEEE, 2014.
23. Benjamin Michéle. *Smart TV Security - Media Playback and Digital Video Broadcast*. Springer Briefs in Computer Science. Springer, 2015.
24. Amihai Neiderman. DVB-T hacking, 2016. https://www.youtube.com/watch?v=GO-8fBYKhAo.
25. Amihai Neiderman. Breaking Tizen, 2017. https://www.youtube.com/watch?v=k_xCyml6XZY&feature=youtu.be.
26. OIPF. DAE V2.3. 2014-01. http://www.oipf.tv/docs/OIPF-T1-R2_Specification-Volume-5-Declarative-Application-Environment-v2_3-2014-01-24.pdf.
27. Yossef Oren and Angelos D Keromytis. From the aether to the ethernet - attacking the internet using broadcast digital television. In *23rd USENIX Security Symposium (USENIX Security 14)*, 2014. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/oren.
28. Rafael Scheel. Smart TV Hacking, 2017. https://www.youtube.com/watch?v=bOJ_8QHX6OA.
29. ETSI TS. 102 796 V1.3.2. *Hybrid Broadcast Broadband TV (HbbTV 2.0.1)*, 2016-04. https://www.hbbtv.org/wp-content/uploads/2015/07/HbbTV-SPEC20-00023-001-HbbTV_2.0.1_specification_for_publication_clean.pdf, consulté le 10/01/18.
30. ETSI TS. 102 809 V1.3.1. *Digital Video Broadcasting (DVB); Signaling and carriage of interactive applications and services in Hybrid broadcast/broadband environments*, 2017-06. http://www.etsi.org/deliver/etsi_ts/102800_102899/102809/01.03.01_60/ts_102809v010301p.pdf.
31. Candid Wueest. How my TV got infected with ransomware and what you can learn from it. *Symantec 's blog*, 2015. http://www.symantec.com/connect/blogs/how-my-tv-got-infected-ransomware-and-what-you-can-learn-it.