e-Security coursework - A blockchain based public Electronic Program Guide

TBC

Edinburgh Napier University School of computing, Edinburgh, Scotland

Friday 24th March, 2023

Contents

List of Figures

Listings

1 Introduction

In today's global world, information is disseminated in many formats, in many locations, users are truly global citizens, can live and work at different places and within different time zones. Besides all this different languages are used not only in the countries where they originate.

An example of this, would be a user, residing two weeks per months in Turkey, the remaining in Switzerland. This worker can use many languages both in its private and work life, depending on the circumstance, he would also be subjected to time zone fluctuations in his timetable.

One common problem resulting from this is the lack of a space where users can find information that is relevant to the location's time zone, desired language, can be obtained publicly and in the desired language.

A person going to the cinema in the UK, would see a movie in English language, in many other countries, you would be able to choose between the local language, original version, with or without sub titles.

The same applies to other forms of entertainment, like TV shows. For the savvy world traveller, satellite TV is a must and today's technology allows one to receive literally thousands of TV channels over one dish to a satellite box or receiver. Such systems with many tuners cheaply available, allow to watch on TV, record to storage, stream to devices TV programs. In this context a whole family could use such a system where the members do not always reside in at the same location, do not necessarily want to watch a program in a predefined language.

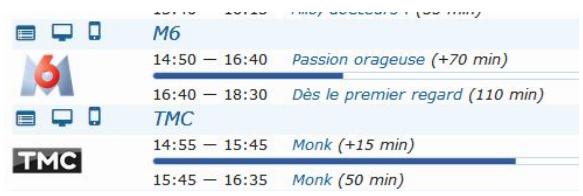
One popular system allowing all this is the Enigma2 Graphical User Interface (GUI) that runs on many set top box platforms, it was originally developed by Dream Multimedia GmbH. The project is now open source and can be found at http://cgit.opendreambox.org/opendreambox.git.

In this work, we will focus on a particular aspect of the its operations, the Electronic Program Guide (EPG).

2 The Electronic Program Guide

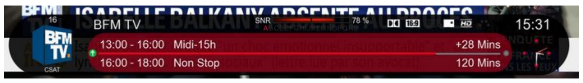
An Electronic Program Guide is an interactive feature that will display on a web interface (Figure 1), or TV (Figure 2), or a mobile application information about a program, and/or a listing of programs over a given period of time. It is possible to interact with the EPG, to schedule a recording of program for example, the information is recorded to a file linked to the record in this case. This information can be displayed again when viewing the said program at a later time.

Some examples of EPG screens are shown below.



M6
14:50 — 16:40 *Passion orageuse (+70 min)*
16:40 — 18:30 *Dès le premier regard (110 min)*
TMC
14:55 — 15:45 *Monk (+15 min)*
15:45 — 16:35 *Monk (50 min)*

Figure 1: Web based EPG



BFM TV
SNR 78 %
15:31
13:00 - 16:00 Midi-15h +28 Mins
16:00 - 18:00 Non Stop 120 Mins

Figure 2: On Screen Display EPG

2.1 The Enigma2 EPG system

On an Enigma2 system, the EPG is configured to include the following information:

- Storage, where the data will reside (set top box internal memory, hard drive, or USB key).

- A local program database used for matching program's information to channels (the lamedb at http://www.satsupreme.com/showthread.php/194074-Lamedb-format-explained).
- A schedule for downloading the EPG data.
- A list of files residing on local storage that describe where to from download the program descriptions.

So in order to operate, the CrossEPG system on the set top box will wake everyday at the given time, and process the configuration files specific to selected countries, channels, and languages. It will download compressed xml formatted files containing program descriptions, process them and finally import the result into the lamedb database. Once reloaded into the set top box's memory, the EPG data is available.

From the back-end perspective, teams usually split by countries or topic of interest, grab or download program data off the TV websites - since there is today no agreement or real standard to provide this service. They then massage this data, converting it to xml (code listing 1). A mapping of TV channels to satellite transponder is also provided (code listing 2). Finally the files are compressed and uploaded to the different web servers for distribution.

```
<?xml version="1.0" encoding="UTF-8"?>
<tv generator-info-name="Rytec" generator-info-url="http://forums.openpli.org">
  <channel id="NottsTV.uk">
    <display-name lang="en">Notts TV</display-name>
  </channel>
  <channel id="W.uk">
    <display-name lang="en">W</display-name>
  </channel>
    <programme start="20190515000000 +0100" stop="20190515003000 +0100" channel="
         NottsTV.uk">
    <title lang="en">Working Week</title>
    <sub-title lang="en">Episode 6</sub-title>
    <desc lang="en">We take a look at one of Nottingham's biggest employers,
         Experian, and meet a woman who set-up her own business after being made
         redundant and burgled. Plus Des finds out what was your best job ever!</
         desc>
  </programme>
  <programme start="20190515003000 +0100" stop="20190515013000 +0100" channel="
       NottsTV.uk">
    <title lang="en">F-Stop</title>
    <sub-title lang="en">Episode 3</sub-title>
    <desc lang="en">F-Stop showcases the best film makers, animators and video
         artists in the East Midlands. In this episode, we take a look at the work
          of Deborah Haywood</desc>
  </programme>
    <programme start="20190515000000␣+0000" stop="20190515004000␣+0000" channel="
        W.uk">
    <title lang="en">Grimm</title>
    <sub-title lang="en">[Series] Fugitive (S6E1)</sub-title>
    <desc lang="en">The fantasy crime drama returns for one final series, as
         Captain Renard claims power as the mayor-elect of Portland, posing a new
         threat to Nick and his allies.
David Giuntoli
Sasha Roiz
Claire Coffee
Silas Weir Mitchell
Bree Turner
Bitsie Tulloch
Russell Hornsby
Reggie Lee</desc>
  </programme>
</tv>
```

Listing 1: Enigma2 CrossEPG xml file

```
<?xml version="1.0" encoding="utf-8"?>
<channels>

<!-- 13.0E --><channel id="NASA.us">1:0:1:1F8:384:13E:820000:0:0:0:</channel><!--
      NASA -->
<!-- 15.0W --><channel id="ESPN2.us">1:0:1:386:1:1:D7A3141:0:0:0:</channel><!--
     ESPN2 -->
</channels>
```

Listing 2: Mapping of TV channels to satellite transponder

2.2 The problematic

This CrossEPG does function well, however, it suffers from some shortcomings. Because of its popularity, the load on the web servers hosting the data is quite high, and downloads are restricted to one per day, per set top box.

Sometimes a server's address or URL will change, because the system is based on free hosting (the group that is grabbing, converting the program data ceases to exist, or cannot afford hosting anymore). When this happens, the user will not be able to download EPG data and will have either to update the set top box software (to get the new web server URL), or search on forums for the new address and manually change the configuration files (code listing 3)

```
description=Switzerland - Basis (xz)
protocol=xmltv
channels_url_0=http://rytecepg.dyndns.tv/epg_data/rytec.channels.xml.xz
channels_url_1=http://www.xmltvepg.nl/rytec.channels.xml.xz
channels_url_2=http://rytecepg.epgspot.com/epg_data/rytec.channels.xml.xz
epg_url_0=http://rytecepg.dyndns.tv/epg_data/rytecCH_Basic.xz
epg_url_1=http://www.xmltvepg.nl/rytecCH_Basic.xz
epg_url_2=http://rytecepg.epgspot.com/epg_data/rytecCH_Basic.xz
preferred_language=eng
```

Listing 3: Enigma2 CrossEPG configuration file

Another issue with the current system, is updating the data, since we can only download the EPG once per day, any update to the program's data (like a schedule modification) can only be picked up twenty four hours later.

Besides data transfer is done using the HTTP protocol, and therefore there is no guaranty that data in not changed in transit (authentic), nor can we check the server's identity. This has a non negligible impact on the overall system's security.

3 An EPG based on the Ethereum blockchain

3.1 Background

As defined in [1], a blockchain is a decentralised transaction ledger. A block of transactions is validated by a proof of work concept where nodes will compete to resolve a puzzle against a reward thus validating the transaction.

The first generation blockchain systems suffers from a low scalability (transactions take a long time to be validated) issue, and are currency orientated. Second generation blockchains [2] are implementing "a trustful object messaging compute framework". That is smart contracts can be created into the chain, they contain code or logic and data. Miners are rewarded only for the work they carry out, or validating the contract.

In essence, one of the very useful feature of Ethereum for our EPG system is the Data Feed concept where external data (the program's information) is entered into the blockchain via a contract. Such Data Feeds have been described in both [3] and [4], two different methods are offered. A general method running "a Trusted Execution Environment" in [3] where the TEE is responsible for pulling the data off the website, and inputs the data into the smart contract; the TEE has to be trusted by the users since the data provider is not involved. This "Town Crier" method has the advantage of working out with the HTTP protocol. The later method called "Practical Data Feed Service for Smart Contracts, or PDFS" works at the application layer and the identity of the content provider is verified using a TLS certificate.

In the case of our system, both methods would work out fine however there are some advantages and drawbacks. The EPG system's data is pulled over many websites, some not implementing the TLS protocol, besides this data "grabbing" is performed with the data owner's involvement, and it is very unlikely that this would change in the future. Therefore the PDFS method would have to be reworked to allow for HTTP based transactions. The second paper describes "Town Crier" using a third party software and hardware. In our EPG, no provider would be willing to maintain such a "costly" set-up.

Therefore we selected to investigate the code from the PDFS system.

3.2 The code for our proof of concept

We downloaded the code from [4], and uploaded it to GitHub at the following URL `https://github.com/40417256/pdfs`, the orginal code is available at `https://gitlab.com/juan794/pdfs`. The goal was to modify the code to our liking and implement our EPG.

We were unable to run the code, despite setting up a full Ethereum private blockchain on a server with 32Gb of RAM and 16 processors. For this reason, it is impossible to show any code modification to PDFS. The specifications we wanted to implement were:

- Removal of the requirement for the TLS protocol.
- Modification of the code to add fields for TV schedules (described in listing 1).
- Build an interface, that would read the EPG in xml format and import the data into the blockchain.
- Allow for creating a new version of a program description.
- Provide code to read back EPG data off the blockchain.

The system we envision would have teams building the EPG data as usual, and then entering them into the blockchain. Set top boxes would require their Enigma2 code modified to be able to read the data from Ethereum. In this way, we there is minimal changes to what is being performed today, and the cost associated with the development negligible. As using the blockchain has a cost, we expect the users to fund the teams by transferring Eth funds, and paying for using the system. One could argue the users could mine the chain, get paid for it, and transfer back to the EPG teams, but simply speaking mining is CPU intensive and set top box hardware is not specified for this usage.

3.3 Testing the project

To test the aspects of our project, we consider the following elements should be investigated:

- The time it takes to enter a program's EPG data into the blockchain.

- The cost of doing it.
- Estimating the growth of the chain over time.

In our test EPG, we selected all channels (approximately 5000) available to us, and downloaded the xml formatted program data, this resulted into less than one Megabyte of information to enter into the blockchain per day.

Because we provide no code, we base the reasoning on the calculations provided in [4], the cost of entering data into the chain is negligible, and does not notably increases as the number of transactions grows over time.

As for the size of the chain, "pruning" should be considered, in [5], the bytecode describing the smart contract is re-used using pointers to previous contracts. The net effect is to reduce the space allocated on the blockchain by more than 70%. Since our contracts would never change except for the program data, we can infer that such a process would greatly save space on our implementation.

4 Conclusions

In this work, we investigate the usage of a blockchain used as a utility for satellite TV viewers. Although it was not possible to develop and execute our code, there is a strong indication that such a system is viable.

There not many utility systems based on blockchains in operation, the main reason is that to operate they require custom development to work on top of it, and general utilities would be provided for free. Investments in this area is mostly limited to creating wealth, and there are not many sponsors for this type of project.

We can however guess that in the near future, governments might be interested into providing public data to citizen and will certainly fund these developments. Hopefully, we expect this democratisation to bring blockchains dedicated to this purpose with standardised APIs to manipulate data around them.

Bibliography

[1] Satoshi Nakamoto et al. "Bitcoin: A peer-to-peer electronic cash system". In: (2008). URL: https://bitcoin.org/bitcoin.pdf (cit. on p. 8).

[2] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151 (2014), pp. 1–32. URL: https://gavwood.com/paper.pdf (cit. on p. 8).

[3] Fan Zhang et al. "Town crier: An authenticated data feed for smart contracts". In: *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*. ACM. 2016, pp. 270–282. URL: https://www.fanzhang.me/files/pubs/tc-ccs16-final.pdf (cit. on p. 9).

[4] Juan Guarnizo and Pawel Szalachowski. "PDFS: Practical Data Feed Service for Smart Contracts". In: *CoRR* abs/1808.06641 (2018). arXiv: 1808.06641. URL: http://arxiv.org/abs/1808.06641 (cit. on pp. 9, 11).

[5] B. B. Fiz Pontiveros, R. Norvill, and R. State. "Recycling Smart Contracts: Compression of the Ethereum Blockchain". In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Feb. 2018, pp. 1–5. DOI: 10.1109/NTMS.2018.8328742 (cit. on p. 11).