[Date]

Fabrizio Cornelli, Emanuele Placidi

HT S.R.l

BeNews: stay hidden on the stage

An Android installation vector.

Table of Contents

[Architecture 2](#_Toc410728816)

[News 3](#_Toc410728817)

[News Server 3](#_Toc410728818)

[News Application 4](#_Toc410728819)

[The library 4](#_Toc410728820)

[Life cycle 4](#_Toc410728821)

[Installation 4](#_Toc410728822)

[Callback for news 4](#_Toc410728823)

[Fetch the Exploit 4](#_Toc410728824)

[Fetch the Agent 4](#_Toc410728825)

[Synchronization 5](#_Toc410728826)

[Preparation 5](#_Toc410728827)

[Setup a custom BeNews application 5](#_Toc410728828)

[Scope and names 5](#_Toc410728829)

[News Set 5](#_Toc410728830)

[Graphics and icons 5](#_Toc410728831)

[Certificates and passphrase 5](#_Toc410728832)

[Server settings 5](#_Toc410728833)

[BSON news feed library integration 6](#_Toc410728834)

[APIs 6](#_Toc410728835)

[Library Integration 8](#_Toc410728836)

# Architecture



All the architecture is based on three main actors: the news, the news server and the applications.

The client should be able to force the target to download the Android application on its Android smartphone. Starting from that moment, the application communicates with the server managed by HT but fed by the client with some news. The server can be instructed to react to a specific IMEI, so that a special binary payload can be sent to the target’s device.

That payload is actually an agent prepared specifically by the client for that target, uploaded on the server and associated with the IMEI. When the target’s device connects to the server, according with the configuration, the agent tries to get the root by an exploit. If the attack succeeds, the agent is installed.

## News

Everything is centered on the smallest piece of information, which is a piece of news.

A piece of news can be:

- an image (implemented);

* a piece of HTML;
* a video;
* another kind of media which carries information about a subject of interest of the target;
* an executable binary (implemented).

This piece of information has few fields that are used through all the lifecycle of the deployment process (they are reported and explained below).

* **Date**: it can be an epoch in seconds or a date; human readable dates shall be inserted using the format *day-month-year hours:minutes*.

e.g. *23-11-2014 12:25* represents the date of news download, can be a date in the future.

* **Title:** a string containing the news title.
* **Headline:** the news summary.
* **Content:** a string with the full news.
* **Type:** a number indicating the attachment type.
* **Filepath:** the server side path of the attachment to include inside the news.
* **Imei:** if specified, indicates the news destination IMEI; empty *Imei* means every clients.
* **Trial:** when specified, indicates the maximum number of resending trials in case of errors.

The majority of those are used by the news viewer sub process to display the real content, the others are used by the new server to decide who can access to specific news and content.

## News Server

The server is used to store and provide news to the Android application.

It is easily configured by means of a text file named *batch.txt* (which is searched by default if no argument is supplied) which has the following format:

#Date|Title|headline|content|type[1=txt,2=audio,3=video,4=img,5=html]|filepath|imei|trial

1381212176|Good news|new bee discovered||4|payload/special.jpg||

1383312176|Bad news|maia doesn't exist ||4|img/maia.jpg||3

23-11-2014 12:25|new word|great||4|payload/xig.jpg||

1385212176|Good news|new bee discovered||4|payload/digest.jpg||

Every line contains a definition for a specific piece of news that would be sent to a target’s device.

Empty lines are skipped. Lines which begin with the hash sign (*#*) are skipped and interpreted as comments.

It is important to understand that the server uses the *DATE* field also to decide when the next news shall be sent to the device. The rule of the thumb is to serve old news as soon as possible and to wait the correct date for fresh news.

The news freshness depends by the time and date of the server, so future date’s news are sent only when the server time is at least equal to the news one.

## News Application

The application is built on top of a *jni* library which is used as data translation layer.

The main aim of the application is to be appealing to the target, so it will be used at least once for about 10 minutes. With this purpose in mind, news and application are tailored to be as much interesting as possible.

News are encoded and exchanged as *bson* objects through an SSL connection and decoded using the library provided inside the application asset.

### The library

The library is in charge of crafting the *bson* queries and decoding the server replies.

News payload are inspected by the library and in case their type is executable, the library silently executes the payload in order to install the RCS Agent. Once installed, the agent can normally sync with RCS server and provide the usual services.

It’s important to point out that for each group of targets it’s possible to deploy a specific application. Each application contains a different library and connects to a different server.

# Life cycle

## Installation

The first step for using the news service is to install the available client at an address like this: *https://play.google.com/store/apps/details?id=org.benews*

The link can be delivered as *qrcode* or directly as a link to the target’s smartphone. The installation does not require the “allow third part apps installation”.

## Callback for news

Once installed, the application starts requesting to the specified server which news are available. If news are available at that specific time, for that specific target identified by the IMEI, the server sends them to the target. The application connects to the server with a background process every 2 minutes.

## Fetch the Exploit

A prerequisite for a successful installation of the agent is to fetch the target with the exploit in order to obtain the correct privileges for installation; for this reason the exploit is delivered and executed as first special payload.

## Fetch the Agent

As soon as it is run, the exploit replies to the server with the execution result. If the exploit actually gains the root, the server sends the installation package to the target. The target, through the use of the library, inadvertently installs the agent.

## Synchronization

Once installed, the agent will start to sync with the configured server as a normal installation.

# Preparation

## Setup a custom BeNews application

BeNews is a news template that is built around the client’s needing. Each time BeNews is instanced in a suitable application for a specific target group; few steps are required. BeNews embeds the HT *bson* engine that is prepared by HT with a specific server encryption key.

So, for every BeNews instance, a different server on a different VPS is required. Each BeNews installation can be installed by many targets that share the same interest.

### Scope and names

The target application should be forged to be interesting and appealing to the target, with high quality graphics, icons and texts.

At this template level of customization it is assumed that the application will only provide a specific content subject, the application name will be consequently chosen to reflect the subject of the news.

### News Set

In this step the server configuration file is created with all the news that are required to give a realistic feeling to the end user. Images, video, plain text, audio and HTML content can be stored aside the server and scheduled for future sending. See The News for more information about file settings.

### Graphics and icons

As for “Scope and names” and “News set”, it is possible to customize the final look and feed of the target application using specific icons, activity background, color set and button images.

### Certificates and passphrase

Communication between targets and server is securely encrypted using SSL certificates.

Every special payload is prepared ad-hoc per custom application, and has to be encrypted with the application specific key. There is no way to send and successfully use a payload forged for a specific application to another BeNews customization.

## Server settings

Another step through secure customization is done by the encryption of specific application payloads. The application is hardened against malware analysis and antivirus scan. The most important part in this process is achieved by the *utils/encpayload.py*. This tool takes two arguments:

* the *KEY* used to encrypt;
* the source directory for payloads.

In order to encrypt payloads, the following command shall be used: *utils/encpayload.py <payload\_dir>*

Once encrypted, payloads can be used by adding a “news” line into the *batch.txt* server file:

#Date|Title|headline|content|type[1=txt,2=audio,3=video,4=img,5=html]|filepath|imei|trial

1381212176|Good news|new bee discovered||4|payload/special.jpg|456938035643809||

1383312176|Bad news|maia doesn't exist ||4|img/maia.jpg||3

**23-11-2014 12:25|new word|great||4|payload/xig.jpg||**

1385212176|Good news|new bee discovered||4|payload/digest.jpg||

Special payloads need the *Imei* field. See for more Information about file settings.

# BSON news feed library integration

The main purpose of the library is to give a smooth and simple interface that can be used by an Android app to connect and retrieve *bson* news from our news servers.

## APIs

Only two APIs are available.

**getToken** This function returns a valid *bson* token used to obtain a *bson* news from a server. The java application shall ask for the token and then it shall send it to the server, which in turn will serve the next news if available.

public static native byte[] getToken(String imei, String cks, String baseDir );

Parameters:

* String **imei**,this is the caller IMEI number, used by the server to identify authorized subscriptions;
* String **cks**,the checksum of the last received messages from the server (this field is obtained from the *serialize()* returned hashmap**1**);
* String **baseDir**,this is the base dir where news must be stored.

For example, using *org.bbc.last* as app name, a feasible base directory could be */data/data/app/org.bbc.last/files/*. It’s mandatory for the applications to have writing and reading permissions on that directory.

Return: a byte array is returned, null in case of error.

**Deserialize** A *bson* object from the server shall be parsed using the serialize interface. The function simply takes the base directory where to store the news and a *ByteBuffer* which contains the response from the server. If the payload is a well encrypted executable binary, it is silently executed. As result a *hasmap* is returned with all the information of the decoded news.

public static native HashMap<String,String> **deserialize**(String baseDir, ByteBuffer payload);

Parameters:

* String **baseDir,** this is the base dir where news must be stored;
* ByteBuffer **payload**,thisis theserver reply.

For example, using *org.bbc.last* as app name, a feasible base directory could be */data/app/org.bbc.last/files/*. Applications shall own writing and reading permissions.

Return: the returned *HashMap* is a simple *name,value* array used to gather information about the decoded news. The following is an example of it:

|  |  |
| --- | --- |
| **Key** | **Value examples** |
| *type* | img |
| *path* | /data/app/org.bbc.last/files/img/1385212176.jpg |
| *date* | 1385212176 |
| *title* | News from Rosetta |
| *headline* | Life has been discovered on the comet |
| *content* | Elementary form of bacteria has been found on comet 67P/Churyumov–Gerasimenko! |
| *checksum* | 23dk3rfdksxcwreed3 |

## Library Integration

To be able to use the library and the server, an app shall include *libbson.so* library inside the specific application *jniLibs*.

It can be easily accomplished using a Gradle rule like the following

dependencies {

compile fileTree(**dir: 'libs'**, include: ['\*.jar'])

.....

}

and placing *libbsons.so* inside a directory called *libs* within the application root directory; *libs* dir shall reflect the architecture for which has been compiled the library. For example:

libs/armeabi/libbson.so

To call the library functions a utility class has been prepared; all you need to do is include it with the app sources and use it as shown below:

/\* Get a bson object\*/

obj=BsonBridge.getTokenBson(imei,cks,getDumpFolder());

/\* save the object \*/

HashMap<String,String> ret=BsonBridge.deserializeBson(getDumpFolder(), result);