File-based Storage for Integrity Reports

1 Introduction

This new feature introduces the possibility to save IRs on file instead of storing them on DB. To obtain them remotely, the component `HisWebService` has been extended with a new web service, called `HisDownloadReportService`.

1.1 Advantages of IRs on file

- It's easier to backup IRs if they are stored on filesystem than if they are placed on DB.
- Reports can be easily shared between components of OAT (if they are deployed in different nodes) by using a distributed filesystem.
- The use of DB is not recommended if you want to include more information in IR (e.g. measurements of BIOS, MBR, kernel, binaries, ...), because of the larger size of the report (we obtained IRs of about 100 kilobytes).
- The feature is easy to use and can be easily turned on/off by setting a property in the `OAT.properties` configuration file (DB-based storage is still the default choice).

1.2 Advantages of new web service

- It guarantees the remote accessibility of the IR. With the present architecture it means that the Portal can still fetch reports even if it is located in a machine different from the one that is hosting the Appraiser. As a future work, when the support for analysis of IRs by external tools will be added, these tools can fetch IRs from this web service.

2 OAT Components Changes

In order to introduce the possibility of storing IR on file, some of the original OAT components have been extended with new features.

2.0.1 New properties (Appraiser)

Two new properties have been added to the Appraiser property file (`OAT.properties`):

- **`IR_DIR`.** Absolute path of the directory where the IRs will be placed; if it's not set (default) IRs are stored on DB.

- **IR_DIGEST_METHOD.** Digest method used to check the integrity of the IR; "SHA-256" is its default value.

2.0.2 New class `HisReportIO` (Appraiser)

It has been introduced a new class (`HisReportIO`) that contains functions for writing and reading IRs:

- **`String writeIR(long reportId, String reportString).`** It receives the value of field `id` in table `audit_log` and the content of the IR to be written; it returns the string to be written in field `report` of table `audit_log`.

  If the property `IR_DIR` is set, `writeIR()` writes `reportString` on a file whose name is obtained concatenating the prefix "report_log_" and the received `reportId`. Then the function uses the digest method read from property `IR_DIGEST_METHOD` to compute the hash of `reportString` and returns a string whose syntax is as follows:

  report_digest:*digest_method*:*digest_value(hex)*

  If the property `IR_DIR` is not set the function returns the unmodified content of `reportString`, that's actually the value to be written on DB.

- **`String readIR(long reportId, String reportString).`** It receives the value of fields `id` and `report` in table `audit_log`; it returns the content of the integrity report.

  If the report is stored in a file, the syntax of `reportString` is the one described above. In this case `readIR()` retrieves the report through `reportId`, computes the hash and compares its value with the one obtained from received `reportString`. If they match it returns the content of report or, otherwise, the null value.

  If the report is stored on DB, it returns the content of the IR, obtained through a query (as it is done in the current version of OAT).

2.0.3 New service `HisDownloadReportService` (`HisWebService`)

The new web service `HisDownloadReportService` contains only one function:

- **`String fetchReport(long reportId).`** It receives the id of IR to be fetched and returns the content of requested report.

The Portal page `report.php` has been updated in order to obtain the IR content from web service instead of DB.

2.0.4 New element `url` in attestation result

In order to add to the attestation request a link to the evaluated IR, two modifications have been done on OAT:

- **Element `url` in attestation result.** It contains the URL of the Portal page showing the IR related to the request.

- **Property `portal_address` (in `OpenAttestationWebServices.properties`).** It contains the address of the machine running the Portal; its default value is the name of the host that receives `PollHosts` requests.

3 Modified Architecture of IR Storage

3.1 Writing process

Figure 1 describes the updated writing process, that uses the new feature introduced. It consists of subsequent steps:

1. The Host Agent sends the IR using the method `postIntegrityReport()` in `HisWebService`.
2. The IR is sent to the function `submitReport()` of the Appraiser, that creates a new entry in table `audit_log`.
3. The Appraiser calls the new function `writeIR()`, which eventually writes on filesystem and returns the string to be written on DB.
4. The Appraiser updates the previously created `audit_log` entry, writing in field `report` the output of function `writeIR()`.

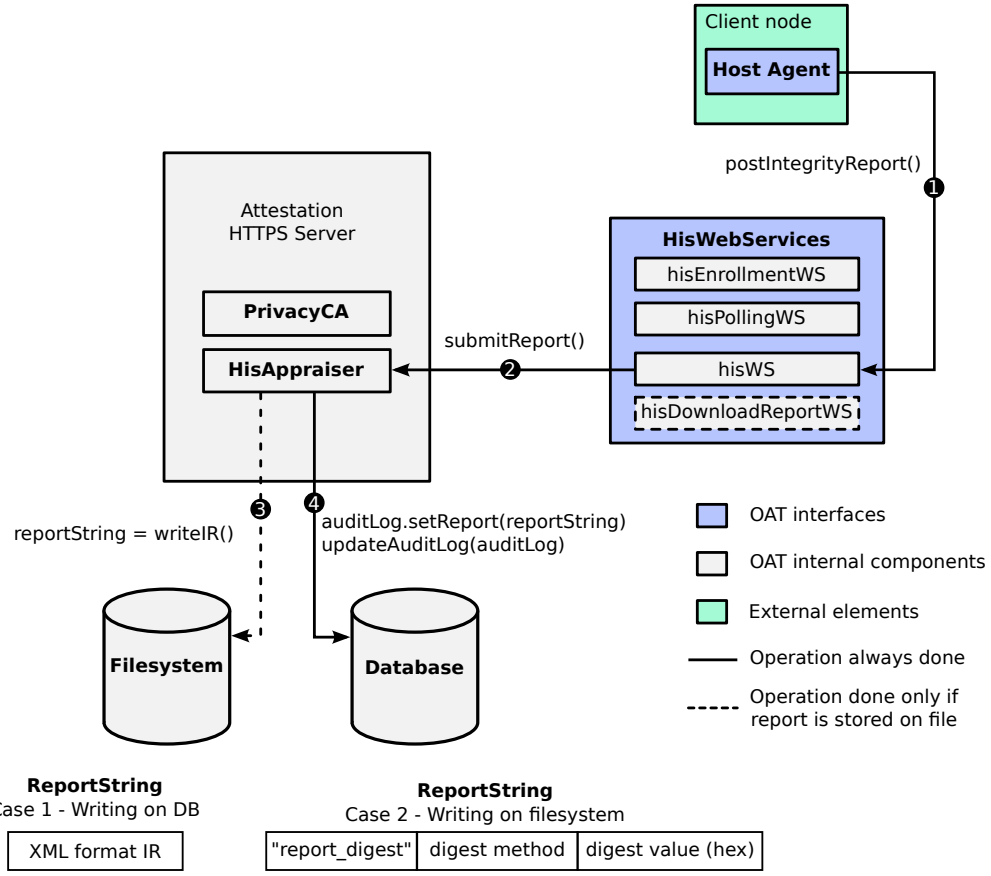

Figure 1: New IR writing process

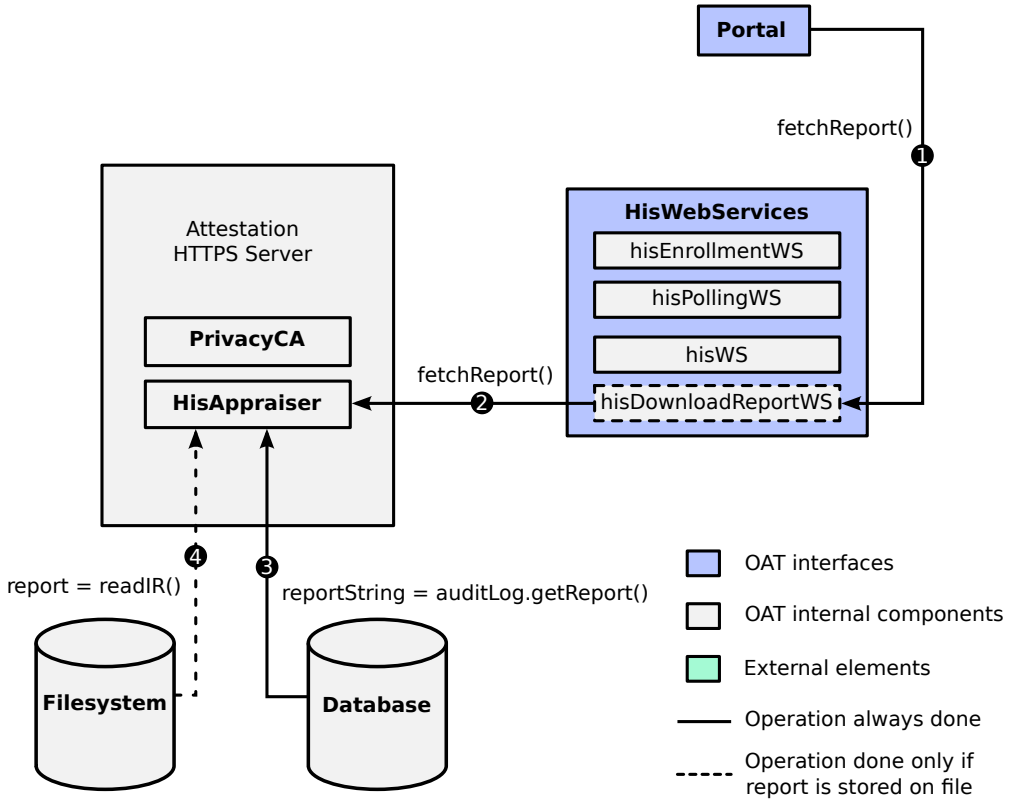

Figure 2: New IR reading process

3.2 Reading process

Figure 2 describes the updated reading process, that uses the new feature introduced. It consists of subsequent steps:

1. The Portal calls the function `fetchReport()` of the new web service `HisDownloadReportService`.

2. `HisDownloadReportService` calls the new function `fetchReport()` of the Appraiser, that checks if the received ID is valid.

3. The Appraiser reads the field `report` of table `audit_log`.

4. The new function `readIR()` receives the value of field `report` and evaluates it to determine from where the report should be fetched.

   If the string begins with "report_digest", `readIR()` retrieves the report from the filesystem, while otherwise it returns directly the content obtained from the DB. Also, in the first case, it determines before returning the result if the file has been corrupted by comparing the digest of read content with that extracted from the string stored in the DB.

4 Evaluation

Figure 3 sums up the result of the compatibility evaluation related to the adoption of the new feature (*OAT v1.7* in figure). The depicted table lists the requirements

from \ to	**OAT v1.6**	**OAT v1.7**
OAT v1.6	DB	DB
OAT v1.7	DB/file*	DB/file

*A conversion tool is needed

Figure 3: Evaluation for the new feature

that must be satisfied to migrate from one version to another (*DB*: report is stored into DB, *file*: report is stored in a file).

Migration from OAT v1.6 to OAT v1.7 is always possible, as the latter supports both the DB and file storage types. On the contrary, the switch from OAT v1.7 to OAT v1.6 would be possible with a conversion tool that overwrites the value of field `report` in `audit_log` table (when it starts with the string "report_string") with the report content read from the filesystem.