IMA Template Management Mechanism

Introduction

The original `ima` template is fixed length, containing the filedata hash and pathname. The filedata hash is limited to 20 bytes (md5/sha1). The pathname is a null terminated string, limited to 255 characters. To overcome these limitations and to add additional file metadata, it is necessary to extend the current version of IMA by defining additional templates. For example, information that could be possibly reported are the inode UID/GID or the LSM labels either of the inode and of the process that is accessing it.

However, the main problem to introduce this feature is that, each time a new template is defined, the functions that generate and display the measurements list would include the code for handling a new format and, thus, would significantly grow over the time.

The proposed solution solves this problem by separating the template management from the remaining IMA code. The core of this solution is the definition of two new data structures: a template descriptor, to determine which information should be included in the measurement list; a template field, to generate and display data of a given type.

Managing templates with these structures is very simple. To support a new data type, developers define the field identifier and implement two functions, init() and show(), respectively to generate and display measurement entries. Defining a new template descriptor requires specifying the template format (a string of field identifiers separated by the `|` character) through the `ima_template_fmt` kernel command line parameter. At boot time, IMA initializes the chosen template descriptor by translating the format into an array of template fields structures taken from the set of the supported ones.

After the initialization step, IMA will call `ima_alloc_init_template()` (new function defined within the patches for the new template management mechanism) to generate a new measurement entry by using the template descriptor chosen through the kernel configuration or through the newly introduced `ima_template` and `ima_template_fmt` kernel command line parameters. It is during this phase that the advantages of the new architecture are clearly shown: the latter function will not contain specific code to handle a given template but, instead, it simply calls the `init()` method of the template fields associated to the chosen template descriptor and store the result (pointer to allocated data and data length) in the measurement entry structure.

The same mechanism is employed to display measurements entries. The functions `ima[_ascii]_measurements_show()` retrieve, for each entry, the template descriptor used to produce that entry and call the show() method for each item of the array of template fields structures.

Supported Template Fields and Descriptors

In the following, there is the list of supported template fields (`'<identifier>': description`), that can be used to define new template descriptors by adding their identifier to the format string (support for more data types will be added later):

- 'd': the digest of the event (i.e. the digest of a measured file), calculated with the SHA1 or MD5 hash algorithm;
- 'n': the name of the event (i.e. the file name), with size up to 255 bytes;
- 'd-ng': the digest of the event, calculated with an arbitrary hash algorithm (field format: [<hash algo>:]digest, where the digest prefix is shown only if the hash algorithm is not SHA1 or MD5);
- 'd-modsig': the digest of the event without the appended modsig;
- 'n-ng': the name of the event, without size limitations;
- 'sig': the file signature, or the EVM portable signature if the file signature is not found;
- 'modsig' the appended file signature;
- 'buf': the buffer data that was used to generate the hash without size limitations;
- 'evmsig': the EVM portable signature;
- 'iuid': the inode UID;
- 'igid': the inode GID;
- 'imode': the inode mode;
- 'xattrnames': a list of xattr names (separated by `|`), only if the xattr is present;
- 'xattrlengths': a list of xattr lengths (u32), only if the xattr is present;
- 'xattrvalues': a list of xattr values;

Below, there is the list of defined template descriptors:

- "ima": its format is `d|n`;
- "ima-ng" (default): its format is `d-ng|n-ng`;
- "ima-sig": its format is `d-ng|n-ng|sig`;
- "ima-buf": its format is `d-ng|n-ng|buf`;
- "ima-modsig": its format is `d-ng|n-ng|sig|d-modsig|modsig`;
- "evm-sig": its format is `d-ng|n-ng|evmsig|xattrnames|xattrlengths|xattrvalues|iuid|igid|imode`;

Use

To specify the template descriptor to be used to generate measurement entries, currently the following methods are supported:

- select a template descriptor among those supported in the kernel configuration (`ima-ng` is the default choice);
- specify a template descriptor name from the kernel command line through the `ima_template=` parameter;
- register a new template descriptor with custom format through the kernel command line parameter `ima_template_fmt=`.