Multilevel Security Example

Simon Foley

January 25, 2016

A Document indexing system

A multilevel secure system offers a document indexing system with the operations:

- ☐ assign(n,p,s): give the document (file) located at path p the name n.
- ☐ view(n,s): view the contents of the document with name n.

where s is the subject requesting the operation. Note that document names are unique across the system and are in addition to the name (path) given to the file containing the document. A table is used to store the name-path relationship, for example:

Name	Path
ExamPaper	/home/store/a
Attendance	/home/store/b
LectureNotes	/home/store/c

The system is used to index staff and student documents whereby staff are permitted to access all documents. Students are permitted to access most documents, however, there are certain sensitive documents, such as exam papers, that students may not view.

Information Flow Policy, clearances and classifications

- The information flow policy has a set of classifications $\{staff, student\}$ with an ordering relation defined as $student \leq staff$.
- Each row/document with name n in the table has a classification *security-level*(n), for example,

Name	Path	security-level(Name)
ExamPaper	/home/store/a	staff
Attendance	/home/store/b	staff
LectureNotes	/home/store/c	student

  security-level(n) gives the classification of the information in the given **row** of the table; the classification *security-level*(p) of the file, at path p on the MLS file system, containing the document may be different.
- Alice is cleared to classification $student$ which means she can read and write LectureNotes, but cannot read ExamPaper or Attendance.
  Bob has clearance $staff$ which means he can create a process (subject) running at $student$ to read/write lecture notes and can create another process running at $student$ to read/write exam papers and attendance data.

Secure state changing operations

Operations that describe the behavior of the above operations, while ensuring that the BLP axioms/multilevel security is preserved.

Secure state changing operations

Operations that describe the behavior of the above operations, while ensuring that the BLP axioms/multilevel security is preserved.

```
assign(n,p,s){
    if document with name n exists in table then return null;
    if security-level(p) $\leq$ security-level(s) then
        add (n,p,security-level(s)) to table;
}
```

Secure state changing operations

Operations that describe the behavior of the above operations, while ensuring that the BLP axioms/multilevel security is preserved.

```
assign(n,p,s){
    if document with name n exists in table then return null;
    if $\textit{security-level}(p) \leq \textit{security-level}(s)$ then
        add (n,p,$\textit{security-level}(s)$) to table;
}

view(n,s){
    retrieve entry (n,p,$\textit{security-level}(n)$) for document with name n from table;
    if no entry found then return null;
    if security-level(n) $\leq$ security-level(s) then
        return p;
    else
        return null;
}
```

Secure state changing operations

Using the Bell-LaPadula axioms, let $\leq$ denote the sensitivity ordering between security levels and assume that every file with path p, and subject s, has security level, *security-level*(p) and *security-level*(s), respectively.

Need to recognize that creating a document-path relationship is in itself a piece of information that is at the level of the subject s carrying out the assignment and we must therefore extend the table to include a security-level for this relationship.

Trojan Horse

Document names are unique in the table; a Trojan-Horse can signal one bit of information about the sensitive exam paper to a student.

Trojan Horse

Document names are unique in the table; a Trojan-Horse can signal one bit of information about the sensitive exam paper to a student.

A covert channel exists since a low-level user can test for existence of a high level name-path by attempting to assign that name to another document.

1. A Trojan-Horse executing in Bob's process (classified at $staff$) wishes to signal a bit of ($staff$) information to Alice's process classified at $student$.
2. The Trojan-Horse assigns/inserts a new document with agreed name 1111 meaning 1, or does no insertion meaning 0. For example, assign$(1111, /home/store/somefile, processBob)$.
3. At a later (agreed) point in time the collaborator Alice (a student process with classification $student$) then attempts to assign the document name 1111 to some file: assign$(1111, /home/store/somefile, processAlice)$ If the insertion fails Alice deduces 1, or 0 otherwise.

The covert channel can be removed by allowing duplicate names for different security levels. In the example above, the unclassified user would be allowed insert a secret tuple with key 1111.