SAT

A user-centred view of security for the NT file system, Registry and Active Directory

(SAT Prototype + Future Work & Future Plans)

http://www.fim.uni-linz.ac.at/sat/

(MSR Security Workshop 2000-11-27, Cambridge, UK)

A general view of permissions/rights

- The Access Matrix

	Object1	*Object2*	*Object3*	*Object4*	*Object5*	*Object6*	...
User1	-	-	change	change	-	-	...
User2	-	-	-	-	read	-	...
...	...	...	...	...	...	...	...
Group1	change	change	-	-	change	-	...
Group2	read	read	exec	exec	-	read	...
...	...	...	...	...	...	...	...
	Computer1				*Computer2*		...

The problem [1]

- Capability vectors versus access control lists

- Capability vectors:
  permissions/rights are stored as a row of the access matrix
  - example in W2000: right to access RAS, group membership, ...

+ Easy to answer: What are the dedicated user's rights?

- Difficult to answer: Who has rights on a dedicated object ????

The problem [2]

- Capability vectors versus access control lists

- Access control lists (ACLs):
  permissions are stored as a column of the access matrix
  - examples in W2000: NTFS directories and files, Registry keys, objects in the Active Directory

+ Easy to answer: Which users/groups have dedicated object access?

- Difficult to answer: What are all the objects a dedicated user has access to?

The problem gets even worse...

- Inspection of multiple rows necessary (because of group memberships)
- Large access matrix
- Distribution of access information
- Set of results in most cases very large
- ...
- Consequence: standard tools are often of limited use only

How operating systems manage inheritance

- Create time inheritance
  copies the ACL of the parent to its own ACL when the object is created
- Run time inheritance
  evaluates permissions of an object at runtime by traversing the tree from the object up until it finds appropriate permissions
- Automatic propagation of permissions
  whenever permissions of the parent are changed, permissions of underlying objects with enabled inheritance are adapted accordingly

Goal of the SAT project

- Design and evaluate visualisation of hierarchically structured persistent objects
- Deal with “ad hominum” permissions and effective permissions
- Cover the different concepts of inheritance in a single approach
- Prototype for Windows NT / Windows 2000
- Special attention to NTFS, Registry, ADS, ...

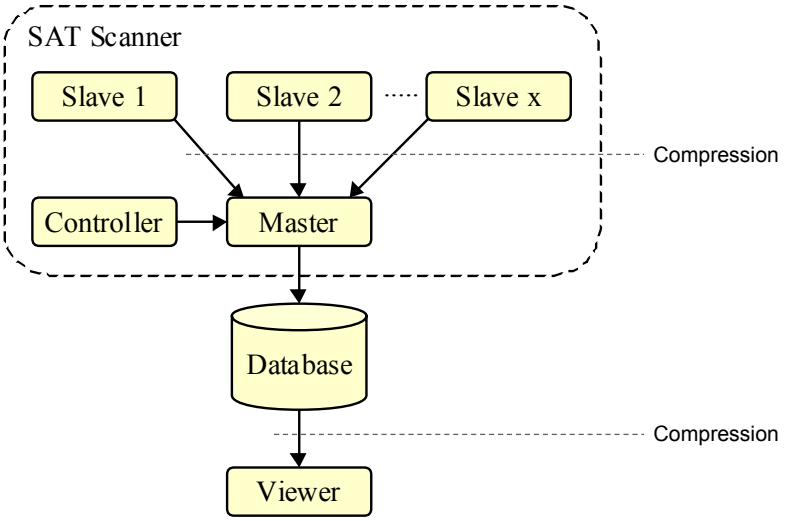
Proposed solution

- Generate a user-oriented view from the already given object-oriented view of security
- Use compression algorithms to make the result short, concise and meaningful.
  Make the compression level selectable according to requirements of different administration tasks
- Offer a viewer with simple GUI and an interface to a report generator

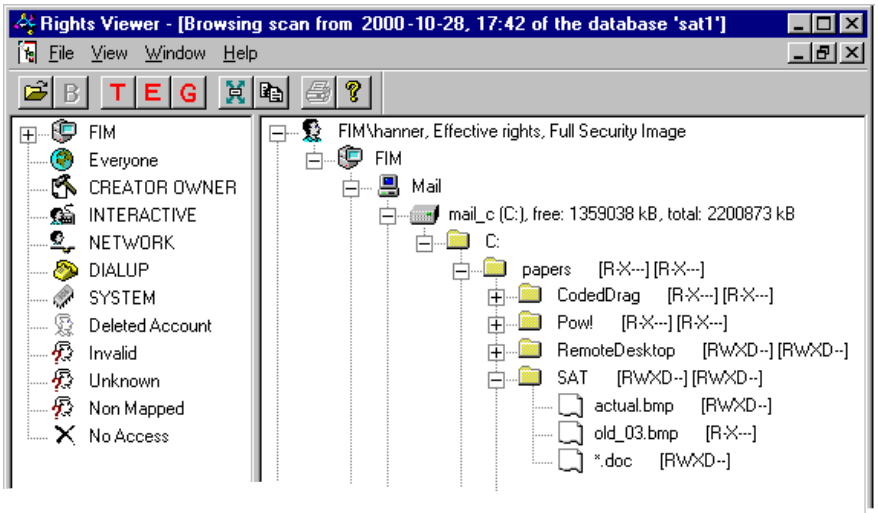
Meaning of “compression”

- Not the standard meaning!
- The goal: we want to make the output (or the browser list) as compact and as precise as possible. Nevertheless it has to remain fully readable.
- Compression in the sense of this presentation e.g. a simple directory listing
  - file1.htm, file2.htm, file3.htm, file4.html, file5.html
  - simple: 3 *.htm, 2 *.html or better: 5 <IE>
- e.g. do not display subdirectories the user does not have access to

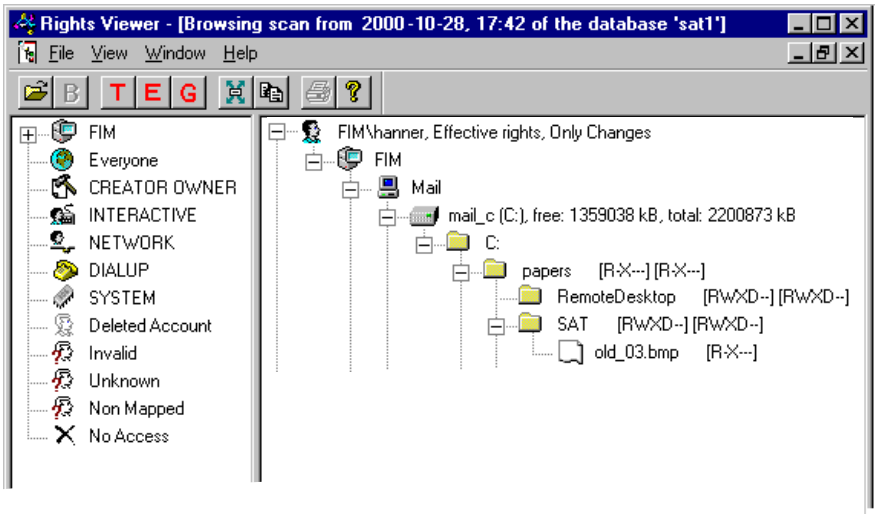
SAT prototype system architecture



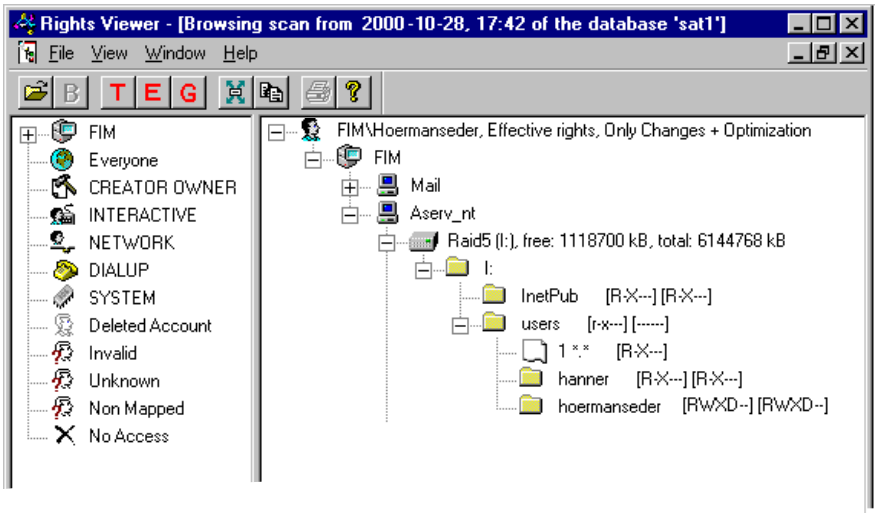
SAT prototype simplified screendump



SAT prototype simplified screendump



SAT prototype
simplified screendump



SAT prototype
file name compression

0 all files are displayed including full filenames

```
 v1000.doc   [RWXD--]
 v1001.doc   [RWXD--]
actual.bmp   [RWXD--]
old_03.bmp   [R-X---]
```

2 all files with same file extension and same permissions are grouped together

```
  2 *.doc   [RWXD--]
  1 *.bmp   [R-X---]
  1 *.bmp   [RWXD--]
```

3 all files with same permissions are grouped together

```
  1 *.*     [R-X---]
  3 *.*     [RWXD--]
```

Expanding the prototype: new file name compression

- All files with same associated application are combined

```
 page1.html [R-X---]
 page2.htm  [R-X---]
 demo1.ppt  [RWXD--]
 demo2.pps  [RWXD--]
pptapp.ppa  [RWXD--]
```

- Result:

```
2 <IE>          [R-X---]
3 <PowerPoint>  [RWXD--]
```


Expanding the prototype: compressing permissions [1]

NTFS 5 standard permissions

Full control, Modify, Read & Execute, Read, Write

comprise 13 advanced permissions:

Traverse folder / Execute file, Create files / Write data, Create folders / Append data, Write attributes, Write extended attributes, Delete subfolders and files, Delete, Change permissions, Take Ownership, ...

A mapping example:

Read = List folder / Read data & Read attributes & Read extended attributes & Read permissions

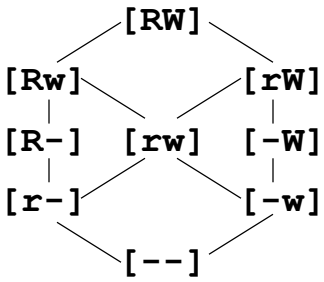
Expanding the prototype: compressing permissions [2]

R = List folder / Read data
& Read attributes
& Read extended attributes
& Read permissions

r = List folder / Read data
| Read attributes
| Read extended attributes
| Read permissions

e.g. only Read and Write

```
[RW]  [rW]  [-W]
[Rw]  [rw]  [-w]
[R-]  [r-]  [--]
```



Expanding the prototype: Windows 2000 inheritance

- standard inheritance

```
\dir\          [RW]
\dir\file1     [RW]
```

(file is not displayed in most cases)

- inheritance + permission W / + deny all

```
\dir\          [Rw]
\dir\file1     [RW]  +=[-W]     =[Rw]+[-W]
\dir\file2     [--]  -=[RW]     =[Rw]-[RW]
```

- no (dynamic) inheritance

```
\dir\          [R-]
\dir\file3 =[RW]
```

Expanding the prototype: display Registry inheritance

C = current key
S = subkeys

0 1 x	
C--	This key only (= do not inherit)
CSS	This key and subkeys
-SS	Subkeys only
CS-	This key and subkeys & propagate inheritable permissions one level only
-S-	Subkeys only & propagate inheritable permissions one level only

Expanding the prototype: display NTFS inheritance

C = current directory
S = subdirectories

F = files
B = both (F+S)

0 1 x	
C--	This folder only (= do not inherit)
CB*B*	This folder, subfolders and files (default)
CS*S*	This folder and subfolders
CF*F*	This folder and files
-B*F*	Subfolders and files only
-S*S*	Subforders only
-F*F*	Files only

Expanding the prototype: maximum permission colours

- red:
  equal and higher permissions in this directory
- yellow:
  equal and lower per-mission in this directory
- green:
  same security in this directory

Expanding the prototype: dealing with ADS complexity

- Large number of different classes
  => difficult to display inheritance
- Large set of permissions
- Extended rights

+ Same concepts as for NTFS and Registry
+ Concentrate on generic / standard permissions
+ Special treatment of objects with default permissions (simply omit them?)
+ Check if only “delegate control (for common tasks)” has been used to modify permissions

Expanding the prototype: compressing AD-permissions

R = Generic Read
= B_X & B_Y & B_Z & ...

r = part of Generic Read
= B_X | B_Y | B_Z | ... |
has read permission on at least one attribute of the object

- Treat other generic permissions the same way
- Combine "Extended Rights" into one (user/group has extended rights on object)

Expanding the prototype: compressing AD-permissions

- We do not want to specify the precise permissions (because there are too much of them, especially if we include R/W on every attribute and all extended rights)
- Show precise permissions with tooltips
- Plan to list samples and their security
  - number of objects with same security
  - object class
  - name of one of these objects
  - already compressed security

Project status

- Prototype (NTFS NT4) available at http://www.fim.uni-linz.ac.at/sat
  - permissions for a user/group and
  - effective permissions for a user/group, including all group memberships Top-down:
  - analysis of frequent cases of ACL inheritance in NTFS, ADS and Registry
  - design new “compression” algorithms

Further plans

- Future work:
  - development of basic modules for the new application: access to ADS, practice with MMC, access to SQL-server, ...
  - design of new “compression” algorithms
- Long-term future (e.g.):
  - inclusion of computers and shares in the analysis
  - compare permissions of different users/groups
  - show permissions of all members of a group or OU (in the form of union and intersection)
  - support automatic comparison of successive scans and mark security-relevant differences

The SAT team at FIM, University Linz, Austria

- Jörg R. Mühlbacher (head of FIM)
- Kurt Hanner (SAT prototype)
- Johann Murauer (SecSim)
- Rudolf Hörmanseder

- Michael Achleitner
- Thomas Helml
- Christoph Kofler
- Gerald Zarda