

Secure Work Files

September 22, 2001

GAMS Development Corp

1217 Potomac Street, NW

Washington, DC 20007

Table of Contents

Introduction	3
A First Example	5
Secure Work Files	7
Access Control Commands	7
Advanced use of Access Control.....	8
Limitations and Future Requirements.....	10

Introduction

When models are distributed to users other than the original developers or embedded in applications to be deployed by other developers, issues of privacy, security, data integrity and ownership arise. We may have to hide, protect or purge some parts of the model before it can be released. The information to be protected can be of numeric or symbolic nature. For example:

Privacy. A Social Accounting Matrix supplied by a Statistical Office is required in a general equilibrium model to be used by the Ministry of Finance. The data from the statistical office needs to be protected for obvious privacy reasons and the model experiments are used to evaluate policy options that are highly confidential. Most of the model structure is public, most of the data however is private and model results need to be transformed in such a way as to prohibit the discovery of the original data.

Security. Components of a model contain proprietary information that describes mathematically a chemical reaction. The associated algebra and some of the data are considered of strategic importance and need to be hidden completely. The final model however, will be used at different locations around the world.

Integrity. Data integrity safeguards are needed to assure the proper functioning of a model. Certain data and symbolic information needs to be protected from accidental changes that would compromise the operation of the model.

To address these issues, access control at a symbol level and secure restart files have been added to the GAMS system.

Access Control. The access to GAMS symbols like sets, variables, parameters and equations can be changed once with the compile time commands \$purge, \$hide, \$protect and \$expose. \$Purge will remove any information associated with this symbol. \$Hide will make the symbol and all its information invisible. \$Protect prevents changes to information. \$Expose will revert the symbol to its original state.

Secure Restart Files. The GAMS licensing mechanism can be used to save a secure model in a secure work file. A secure work file¹ behaves like any other work file but is locked to a specific users license file. A privacy license, the license file of the target users, is required to create a secure work file. The content of a secure work file is disguised and protected against unauthorized access via the GAMS license mechanism.

¹ Work files are used to save and restart the state of a GAMS program. Depending on the context, we refer to those files as work files, save files or restart files.

A special license is required to set the access controls and to create a corresponding secure work file. Reporting features have been added to allow audits and traces during generation and use of secure work files.

A First Example

The model TRANSPORT from the GAMS model library will be used to illustrate the creation and deployment of a secure work file. Assume we want to distribute this model but have concerns about proprietary formulations and data. In addition we would like to protect the user from making unintentional modifications to the model. We assume that the objective function and the supply constraints are to be hidden from other users and only the demand figures can be changed. Data that is not needed any more will be purged as well. This will be demonstrated below using the command line interface to GAMS². First we will copy the model from the GAMS model library, run the model and save a normal work file:

```
> gamslib transport
> gams transport s=t1
```

We continue to enter access control commands in a file called *t2.gms* and create a secure work file with the name *t2*³:

```
> type t2.gms

$eolcom //
$protect all           // make all symbol read only
$purge  d f           // remove items d and f
$hide    cost supply a // make objective invisible
$expose  transport b  // allow changes to b

> gams t2 r=t1 s=t2 plicense=target

GAMS Rev 124 Copyright (C) 1987-2001 GAMS Development...
Licensee: Source User Name
          Source Company Name
*** Creating a Secure Restart File for:
***          Target User Name
***          Target Company Name
--- Starting continued compilation
--- T2.GMS(6) 1 Mb
--- Starting execution
*** Status: Normal completion
```

The access control commands are activated by the use of the privacy GAMS license option PLICENSE. This option specifies the name of the target user license file. The save/restart files *t2* can only be read with the target license file.

² When using the GAMS/IDE interface, the GAMS parameters are entered and maintained in the text window just right to the *Run GAMS (F9)* button.

³ Work files consist of several files with a common name stem. By default, work files have the extension *g0?*.

The three lines starting with ‘***’ are a recap of the content of the target license file. From now on, the source and the target licensees are ‘burned into’ this file and all its descendants. We are ready to send the restart file to the target user or system.

The target user can now run the model with new data, add new GAMS statements, and make new save/restart files. The only restrictions are that some of the symbols are hidden and that this model can only be executed using the target license file. For example, the target user may want to half the demand and compare the original solution with the new one. We will call this program *t3.gms* and it will be executed on the target system:

```
> type t3.gms

parameter rep summary report;
rep(i,j,'base') = x.l(i,j);
b(j) = b(j)*0.5; solve transport minimizing z using lp;
rep(i,j,'half') = x.l(i,j);
display rep;

> gams t3 r=t2

GAMS Rev 124 Copyright (C) 1987-2001 GAMS Development...
Licensee: Target User Name
          Target User Company
*** Restarting from a Secure Restart File created by:
***          Source User Name
***          Source Company Name
--- Starting continued compilation
--- T3.GMS(5) 1 Mb
...

```

Note that the originator/owner of the secure work file is mentioned by name on the log file. A similar message is contained in the listing file:

```
> type t3.lst
...
EXECUTION TIME          =          0.000 SECONDS    1.1 Mb    WIN201-124

**** Secure Save/Restart File Source:
          Source User Name
          Source Company Name
**** Secure Save/Restart File Target:
          Target User Name
          Target User Company
...

```

A more detailed inspection of the listing file will show that the hidden variables and equations do not appear in the usual equation/variable listings and the solution print. The hidden items can only be accessed via a public (exposed) model and a solve statement.

In the following two sections we will describe secure work files and the access control commands in more detail.

Secure Work Files

Secure Work Files control access to symbolic and numeric information and can only be read by a specific GAMS user. The initial creation or additions to access control requires a special GAMS license. Saving Secure Work Files without new access controls does not require a special GAMS license. The Creation or addition of access control is signaled by the use of the GAMS parameter `PLICENSE`, which gives the name of a privacy license file. The shortcut '`PLICENSE=LICENSE`' sets the privacy license to the current license file. This is convenient when experimenting with access controls.

When a secure work file is written the first time, the first and second lines of the current license file and the privacy license file are inserted into the work file. This information cannot be changed any more and the original source and the intended target users are locked into the work file.

A secure work file can be used just like any other work file and new work files can be derived from secure file. However, their use is restricted to the 'target' user specified with the `PLICENSE` parameter. The target user can, if licensed, add access controls to an existing secure file by using the `PLICENSE=LICENSE` parameter but cannot change the original information about source and target users.

Secure work files can be tested on any GAMS system by specifying a non-default license file with the `LICENSE=target` parameter.

Access Control Commands

There are four Access Control Commands (ACC) that are processed during the compilation phase. These commands can be inserted anywhere and are processed in chronological order and have the following syntax:

```
$acc ident1 ident2 ...  
$acc ALL
```

Where *acc* is one of the four ACC's:

<code>PURGE</code>	remove the objects and all data associated
<code>HIDE</code>	hide the objects but allow them to be used in model calculations
<code>PROTECT</code>	the objects cannot be modified but used in model calculations
<code>EXPOSE</code>	removes all restrictions

The keyword `ALL` applies the ACC to all identifiers defined up to this point in the GAMS source code. ACC's can be changed and redefined within the same GAMS program. Identifiers inherited from a restart file cannot be changed, however.

Advanced use of Access Control

We will again use the transport model to show how to hide input data and results from the target user. The target user is only allowed to view percentage changes from an unknown base case. In addition to the original model we will introduce a data initialization and a report model.

First we will define a new model to calculate input data. The previous parameter *c* is now the variable *newc* and the model *getc* does the calculations:

```
$include trnsport.gms
variable newc(i,j)    new transport data;
equation defnewc(i,j) definition of new transport data;
model getc            compute new transport data / defnewc /;
defnewc(i,j).. newc(i,j) =e= f*d(i,j)/1000;
solve getc using cns;
```

Next, we change the objective function of the original model to a more complicated nonlinear function. Furthermore, we will compute a base case value to be used later in the reporting model. Note the reference to *newc.l(i,j)*, since *nexc* is a variable we have to specify that we only want the level value:

```
scalar  beta scale coefficient / 1.1 /;
equation newcost definition of new objective function;
model newtrans / newcost,supply,demand /;
newcost.. z =e= sum((i,j), newc.l(i,j)*x(i,j)**beta);
solve newtrans using nlp minimizing z;
parameter basex(i,j) base values of x;
basex(i,j) = x.l(i,j);
```

Finally we transform the result by using a third model:

```
variable delta(i,j)    percentage change from base values;
equation defdelta(i,j) definition of delta;
model rep / defdelta /;
defdelta(i,j)$basex(i,j)..
    delta(i,j) =e= 100*(x.l(i,j)- basex(i,j))/basex(i,j);
solve rep using cns;
```

We will save the above GAMS code under the name *p1.gms*, execute and make a save/restart file with the name *p1.g0?* as follows:

```
> gams p1 s=p1
```

Now are ready to make some test runs similar to those we expect to be defined by the target user. We will define three scenarios to be solved in a loop and name the file *u1.gms*:

```
set s / one,two,three /;
```

```

parameter sbeta(s) / one 1.25, two 1.5, three 2.0 /
                sf(s) / one 85, two 75, three 50 /;
parameter report summary report;
loop(s,
  beta = sbeta(s);
  f     = sf(s);
  solve getc using cns;
  solve newtrans using nlp minimizing z;
  solve rep using cns;
  report(i,j,s) = delta.l(i,j);
  report('','beta',s) = beta;
  report('','f',s) = f;
  report('obj','z',s) = z.l ) ;
display report;

```

When executing the above GAMS code together with the original transport model from the GAMS model library we will get the following results.

```
> gams u1 r=p1
```

```

----      109 PARAMETER report  summary report

                                one          two          three
seattle .new-york                -4.050         -6.967         -8.083
seattle .chicago               -18.797        -27.202        -31.550
seattle .topeka                  233.958        348.468        404.187
san-diego.new-york                3.605          6.201          7.194
san-diego.chicago                28.138         40.719         47.228
san-diego.topeka                 -15.512        -23.104        -26.799
          .beta                   1.250          1.500          2.000
          .f                       85.000         75.000         50.000
obj          .z                    526.912        1652.963       13988.774

```

Note that all symbols are still completely exposed. We need to add access controls to the model *p1.gms* before we can ship it to the target client. The information to be protected is the original distance matrix and derived information. We start out by hiding everything and then give access to selected parts of the model. We collect the access control information in the file *s1.gms* shown below and save the secure work file under the name *s1.g0?*. Since we are still testing, we use our own license as target user. This will allow us to test the system the same way the target user will use it:

```

$hide all
$expose getc newtrans rep
$expose i j z delta
$expose f beta a b

```

```
> gams s1 r=p1 s=s1 plicense=license
```

To test the new secure file, we run again the problem *u1.gms*. When doing so you will observe that equation, variable and solution listings related to the hidden variables are not

shown any more. Any attempt to reference a hidden variable will cause a compilation error.

```
> gams ul r=s1
```

Before we can ship a secure work file we need a copy of the target user license file. We then will restart again from *p1.gms*, zip the resulting secure files and we are ready to distribute the model:

```
> gams s1 r=p1 plicense=target.txt s=target  
> zip target target.g0?
```

Limitations and Future Requirements

One of the design goals for secure work files has been to minimize the impact on other components of the GAMS system. Solvers used out of a secure environment should work as if called out of a normal environment. This implies that, in principle, certain information could be recovered, if one has knowledge of GAMS solvers internals and is willing to expand considerable programming effort. In this section we will discuss current limitations and possible extensions to the security features.

The following limitations exist:

1. Solvers are not security aware and it would be possible to write a special GAMS solver that extracts information about a specific model instance. Primal and dual values as well as first partial derivatives could be extracted and displayed.
2. The names and explanatory text of all GAMS symbols are retained in the work file and could be accessed by a special GAMS solver.
3. The source and target license files locked to the secure work file cannot be changed. If the target user upgrades the GAMS system and receives a new license file, the secure work file cannot be read any more.