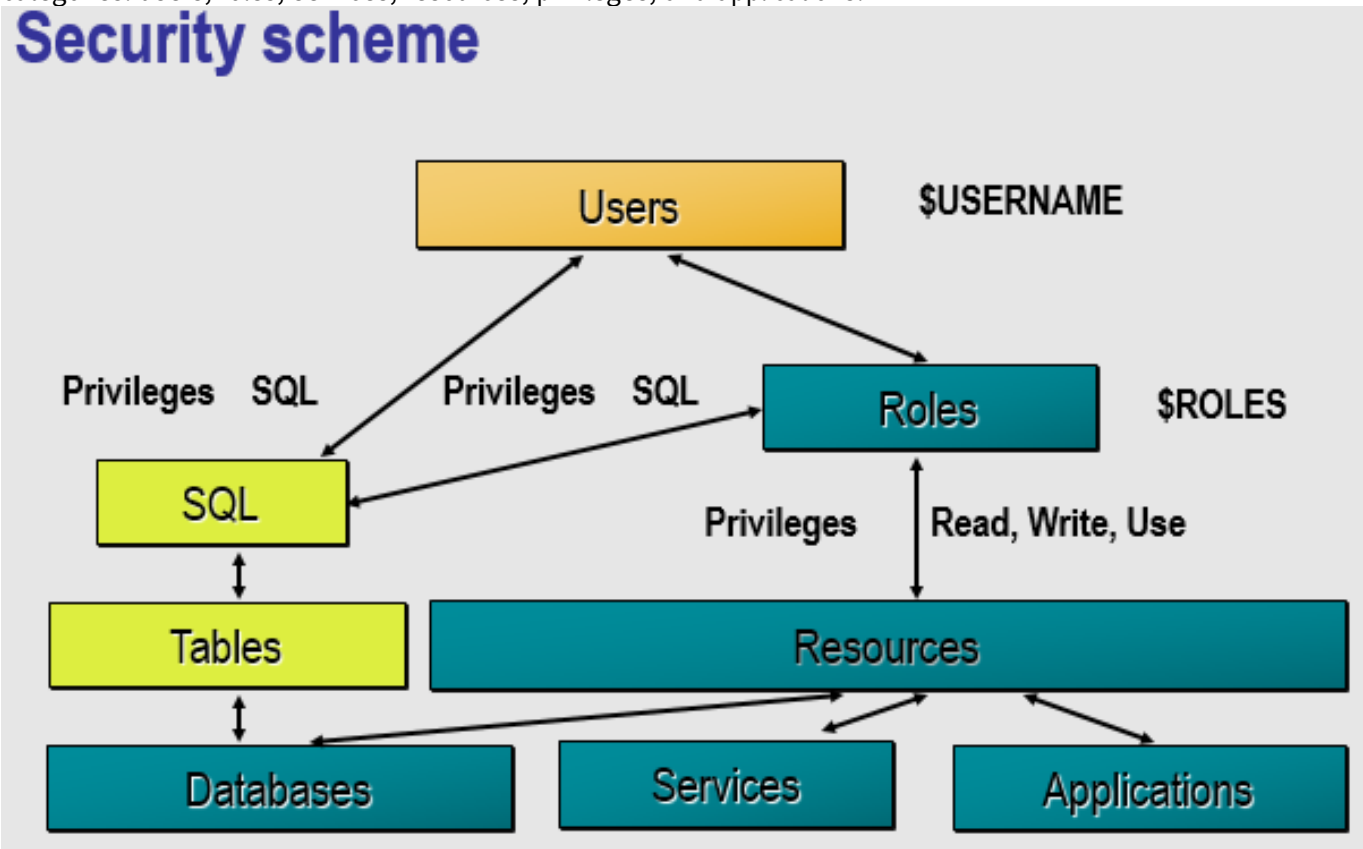
Article
Sergey Mikhailenko · Jun 2, 2020 8m read

Open Exchange

Increasing the Security of the InterSystems IRIS DBMS

When you first start working with InterSystems IRIS, it's a common practice to install a system with only a minimum level of security. You have to enter passwords fewer times and this makes it easier to work with development services and web applications when you're first getting acquainted. And, sometimes, minimal security is more convenient for deploying a developed project or solution.
And yet there comes a moment when you need to move your project out of development, into an Internet environment that's very likely hostile, and it needs to be tested with the maximum security settings (that is, completely locked down) before being deployed to production. And that's what we'll discuss in this article.
For more complete coverage of DBMS security issues in InterSystems Caché, Ensemble, and IRIS, you may want to read my other article, Recommendations on installing the InterSystems Caché DBMS for a production environment.
The security system in InterSystems IRIS is based on the concept of applying different security settings for different categories: users, roles, services, resources, privileges, and applications.



Users can be assigned roles. Users and roles can have privileges on resources — databases, services, and applications — with varying read, write, and use rights. Users and roles can also have SQL privileges on the SQL tables located in databases.

How Security Levels Differ

When installing InterSystems IRIS, you can choose the security level: Minimal, Normal, or Locked Down. The levels differ in the degree of user engagement, the available roles and services, and in the configuration of

authentication methods for services and applications. For more information, read the Preparing for InterSystems Security section of the Preparing to Install InterSystems IRIS guide.
In the documentation you'll find the tables shown below, which show the security settings for each level. You can change the settings in the system management portal interface.

Initial User Security Settings

Security Setting
Password Pattern
Inactive Limit
Enable _SYSTEM User
Roles assigned to UnknownUser

Initial Service Properties

Service Property
Use Permission is Public
Requires Authentication
Enabled Services

Initial Enabled Settings for Services

Service
%Service_Bindings

Service

*%Service_CSP

%Service_CacheDirect

%Service_CallIn

%Service_ComPort

%Service_Console

%Service_ECP

%Service_MSMActivate

%Service_Monitor

%Service_Shadow

%Service_Telnet

%Service_Terminal

%Service_WebLink

*For InterSystems IRIS, %Service_CSP applies %Service_WebGateway.
The services used are slightly different for different operating systems.

How You Can Improve Security

For each enabled service, you need to choose the appropriate authentication methods: unauthenticated, password, Kerberos, or delegated.
You also need to disable web applications that aren't used in the system. And for web applications that are enabled, you need to select the correct authentication method: authenticated, password, Kerberos, delegated, login, or cookie.
Of course, the administrator chooses the security settings for each project and solution so the project can function according to the customer's requirements. And this is always a balance between keeping the system convenient enough that users can actually get their work done, while also secure enough to keep intruders at bay. As you know, the most secure system is a disabled system.
If you encounter a need to manually increase the security level of your system more than once, this is a sure sign you need to write a software module to solve these problems.
In fact, InterSystems Open Exchange has a lockdown program that can help you improve security. You'll find the source code for the program in the repository on the InterSystems isc-apptools-lockdown page.
Here's what the LockDown program does.

First, it changes passwords for preinstalled users:

- Admin,
- CSPSystem,
- IAM,
- SuperUser,
- UnknownUser,
- _Ensemble,
- _SYSTEM.

Second, it disables all services except:

- %%service_web gateway
- %service_console
- %service_login
- %service_terminal

Next, it sets password protection for all web applications, including:

- /csp/ensdemo
- /csp/samples
- /csp/user
- /isc/studio/usertemplates
- /csp/docbook
- /csp/documatic
- /isc/studio/rules
- /isc/studio/templates

Finally, it sets system-wide security parameters including:

- Password complexity "8.32 ANP"
- Limit on user inactivity of 90 days
- Audit and all security-relevant events
  You can install the LockDown program on your system by downloading LockDown.cls from GitHub. Then, in terminal mode, enter the following:

```
USER>zn "%SYS"
```

```
%SYS>do $system.OBJ.Load("/home/irisusr/LockDown.cls","ck")
```

Or you can install it using the ZPM batch manager from the public register with the following commands:

```
USER>zn "%SYS"
%SYS> zpm "install isc-apptools-lockdown"
```

Performing a Lockdown

Before executing a lockdown, it's strongly recommended that you perform a backup.
The LockDown program must be executed from the %SYS area. If you don't want to change the password for all preinstalled users, leave the first parameter empty.
If you want to keep the ability to edit programs and classes using IRIS Studio, Atelier, or VSCode, don't disable the %Service_Bindings service. To ensure this works, the bindings argument must be set to 1. Here's an example:
do ##class(App.Security.LockDown).Apply("New Password 123",.msg,1)
This module also contains a function that's useful if the system password is compromised and you need a replacement for all preinstalled accounts without performing a lockdown. You can run it as follows:
do ##class(App.Security.LockDown).Change Password("New Password 123",
"Admin,CSPSystem,IAM,SuperUser,Unknown User, _Ensemble,_SYSTEM")
Most likely, after performing the lockdown, your application or project will stop working. To fix it, you'll need to restore some security settings to their original state. This can be done either via the management portal interface (security section) or programmatically.

Changing Your Security Settings After Lockdown

After lockdown, if your web applications used authentication methods other than passwords, you'll need to enable them.
I suggest running the software module zpm-registry-test-deployment, which has an example of using LockDown for the ZPM-registry project.
The code that follows is applied at the end of the installation. The project was installed on IRIS with a minimal level of security. Here's what the code had to do:
- Change passwords for all preinstalled users.
- Disable all services not used by this project.
- Enable password protection for all applications on the system, except web applications /registry (which allows unauthorized users to get a list of packages in the registry).
- Create a new user with privileges to publish new packages in the registry. This user must have write rights to the project tables in the IRISAPP database.

Create a new user:

```
set tSC= ##class(App.Security.LockDown).CreateUser(pUsername, "%DB_"_Namespace, pPass
word, "ZMP registry user",Namespace)
If $$$ISERR(tSC) quit tSC
write !,"Create user "_pUsername
```

Add privileges for a new and unauthorized user:

```
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "1,ZPM.Package", "s
", "UnknownUser")
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "1,ZPM.Package", "s
", pUsername)
```

```
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "1,ZPM.Package_depe
ndencies", "s", pUsername)
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "1,ZPM_Analytics.Ev
ent", "s", pUsername)
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "9,ZPM.Package_Exte
nt", "e", pUsername)
set tSC=##class(App.Security.LockDown).addSQLPrivilege(Namespace, "9,ZPM_Analytics.Ev
ent_Extent", "e", pUsername)
If $$$ISERR(tSC) quit tSC
write  !,"Add privileges "
```

Run the LockDown program:

```
set tSC= ##class(App.Security.LockDown).Apply(NewPassSys)
If $$$ISERR(tSC) quit tSC

Change the settings for the web app so that an unknown user can log in:
set prop("AutheEnabled")=96
set tSC=##class(Security.Applications).Modify("/registry",.prop)
If $$$ISERR(tSC) quit tSC
write !,"Modify /registry "

Change the settings for the %service_terminal service, changing the authorization met
hod to Operating System, Password:
set name="%service_terminal"
set prop("Enabled")=1
set prop("AutheEnabled")=48 ; Operating System,Password
set tSC=##class(Security.Services).Modify(name,.prop)
If $$$ISERR(tSC) quit tSC
write !,"Modify service terminal"
```

Wrapping Up

In this article, I discussed why you might want to increase the security level of your system and how you'd do this programmatically, and I showed an example using the InterSystems LockDown program.
We used a method in which we first closed down everything in the system (that is, we set the maximum security level). We then moderated the security by opening the services and applications necessary for the project to function, but only those. I'm sure there are other ways and best practices, and I'd love to hear about them as part of the discussion of this article by the community.

#Beginner #Security #System Administration #Caché #Ensemble #InterSystems IRIS #InterSystems IRIS for Health
Check the related application on InterSystems Open Exchange

Source URL: https://community.intersystems.com/post/increasing-security-intersystems-iris-dbms