Quick Guide to Security with Denkh HTML Reporter

Scott Auge sauge@amduus.com scott_auge@yahoo.com

Table of Contents

Web Server Security

Web server security means that level of communication between the user's browser and the web server (as well those programs behind the web server such as HTML Reporter.)

Using SSL sessions (https)

Setting up the web server to use the HTTPS protocol will cause all communications between the web server and the browser to be encrypted. In short, the web browser makes a connection to the web server which sends the web browser some information to aid it in encrypting information between the server and it's self. From then on, all requests to the web server and replies back from the web server will be encrypted, and thereby secure.

This means that anyone on a machine relaying packets between the server and the browser will not be able to view the HTTP messages going back and forth between the browser and the server. On some networks, packets are "broadcasted" out to everyone on the network (who then usually ignore them – unless they are trouble makers.) An example of this is Ethernet. This form of encrypting will also foil these types of attacks upon your privacy and security.

Setting up SSL is dependent on the type of web server you are using, and is outside the scope of this document. See your web server documentation on how to install and configure SSL on your web server.

Using Web Server Security Settings

Most web servers have a setting whereby it will prompt for a user name and password before allowing a web page or set of web pages to be displayed. If you want to control the number of people who can access the application, you may want to use this service.

Note this is different from the user login and prompt from the HTML Reporter should you decide to use the sections below in "Denkh HTML Reporter Security." If you plan on having no security in regards to who can look at which reports, the web server's login service should work just fine. You will just not want to use the @secure() macro in your reports.

The settings for this functionality is dependent on your web server, and is outside the scope of this document. See your web server documentation on how to install and configure user login control via web server.

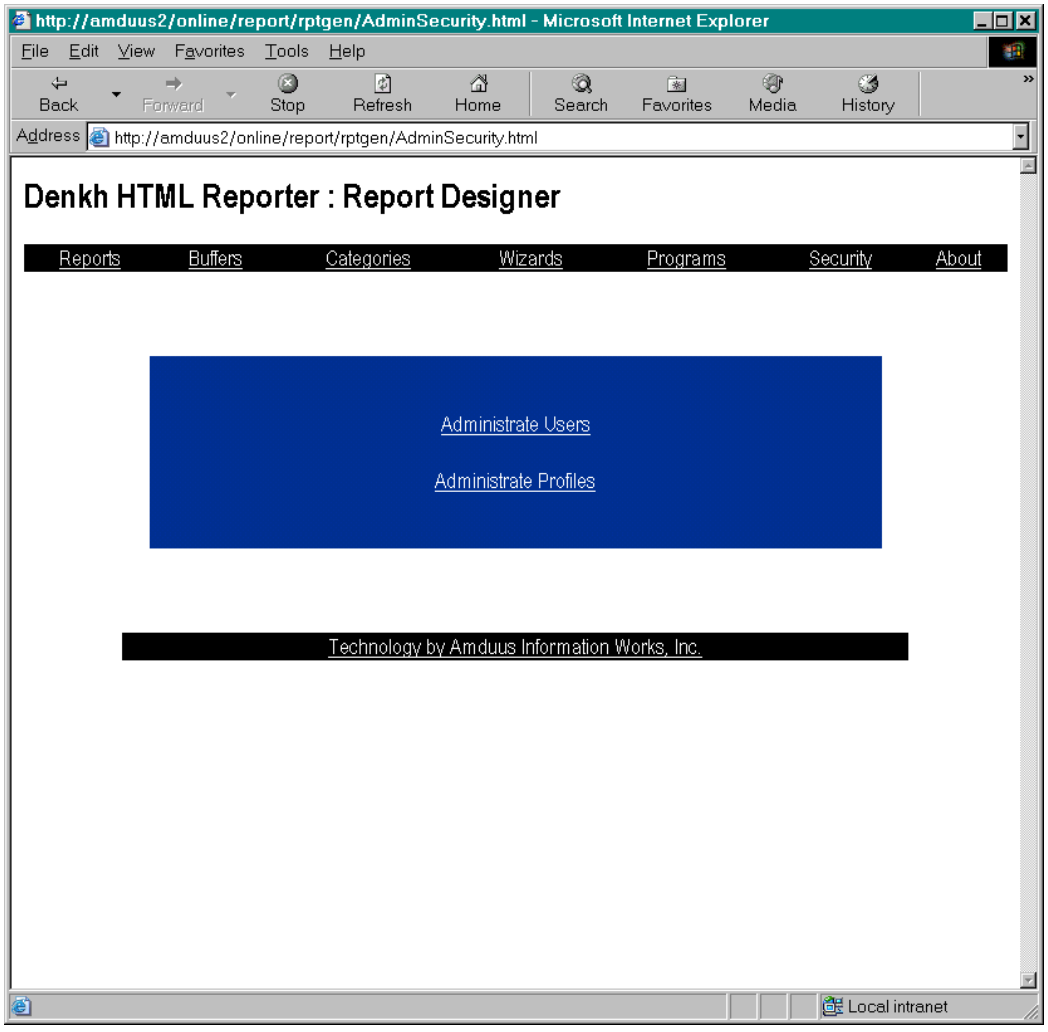
Denkh HTML Reporter Security

Because the web server does not know about the reports or how Denkh HTML Reporter works, there is a layer of security built into the application it's self.

You should use SSL with Denkh HTML Reporter Security, but not the Web Server's user authentication services (this would cause two logins to be required.)

General

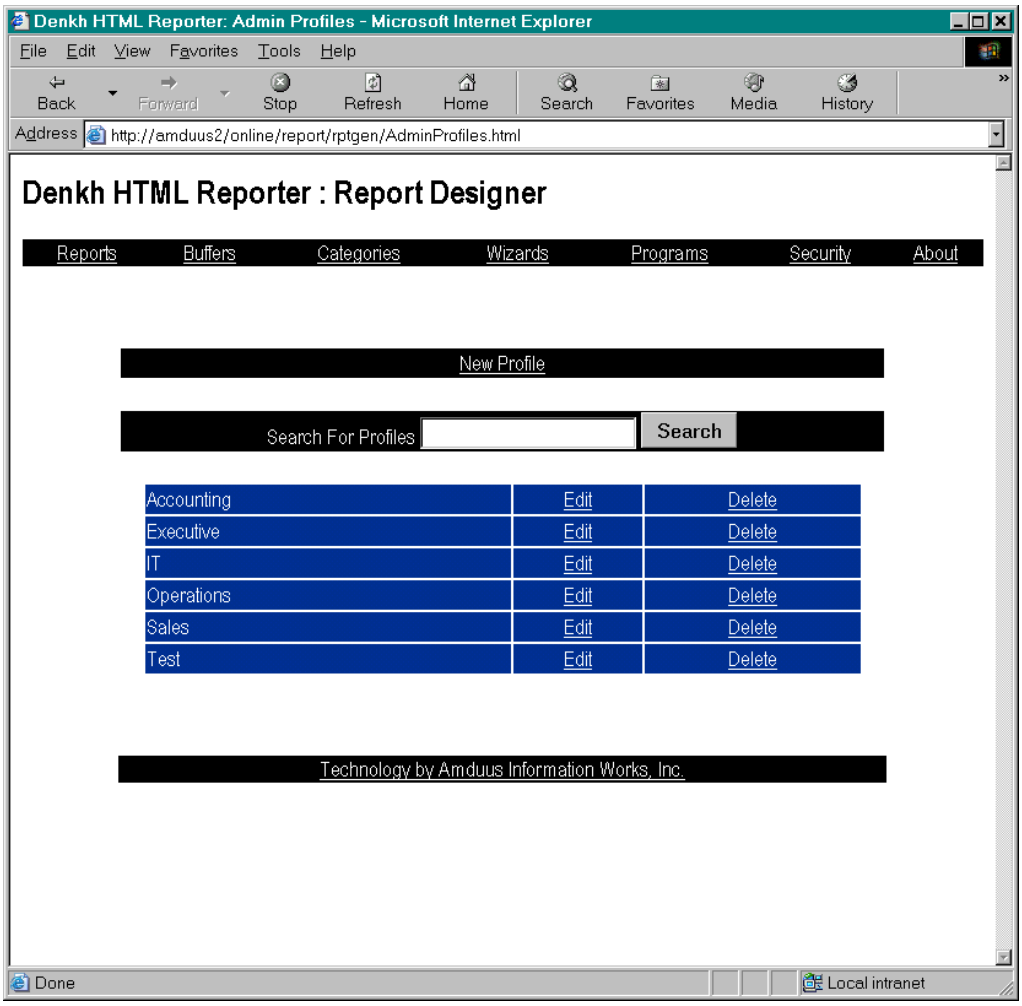
Under the Report Designer is a subsection called Security on the menu bar. Upon pressing this hyperlink, you will be presented with a screen like this:



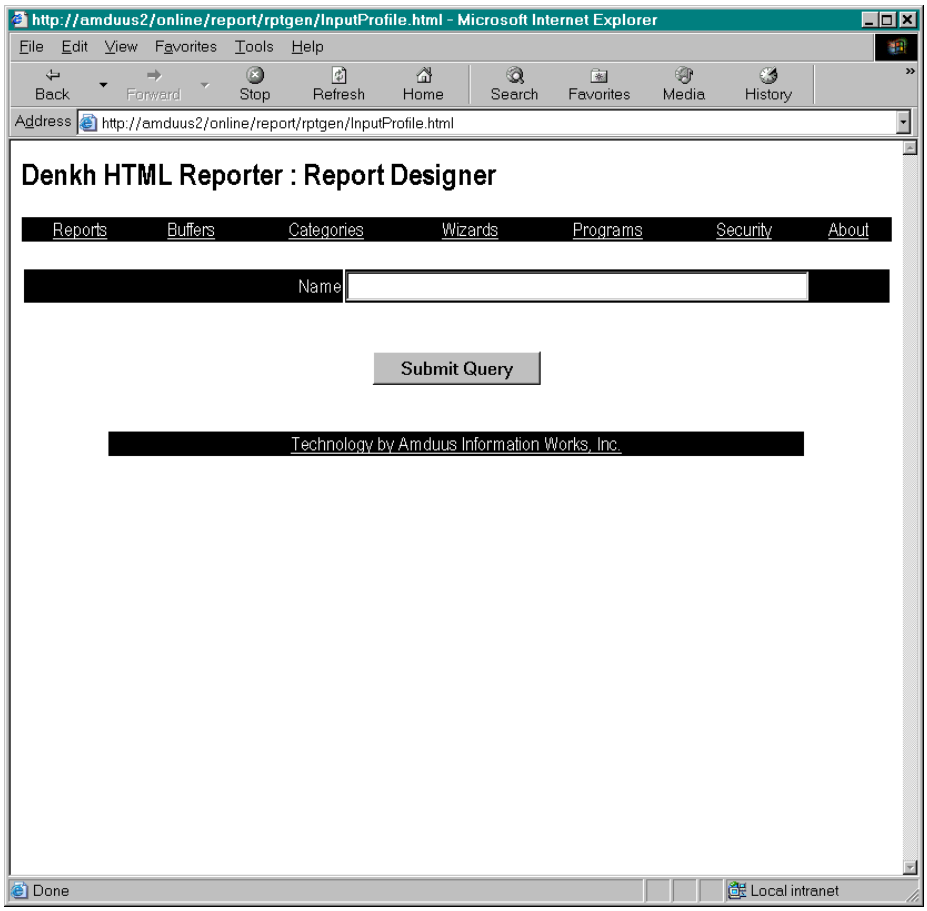
The application has a concept of users and profiles. Users are individuals who would be using the application. Profiles are groups of users. Use of the @secure() macro will allow reports to be viewed only be user's who are members of certain profiles.

Profiles

By clicking on the Administrate Profiles hyperlink, you will be presented with a screen similar to the one below:



This screen presents currently defined profiles in your application. Your profiles may not exist yet, or may be different based on the organization of your company. From here, one can create a new profile by clicking on the New Profile hyperlink:

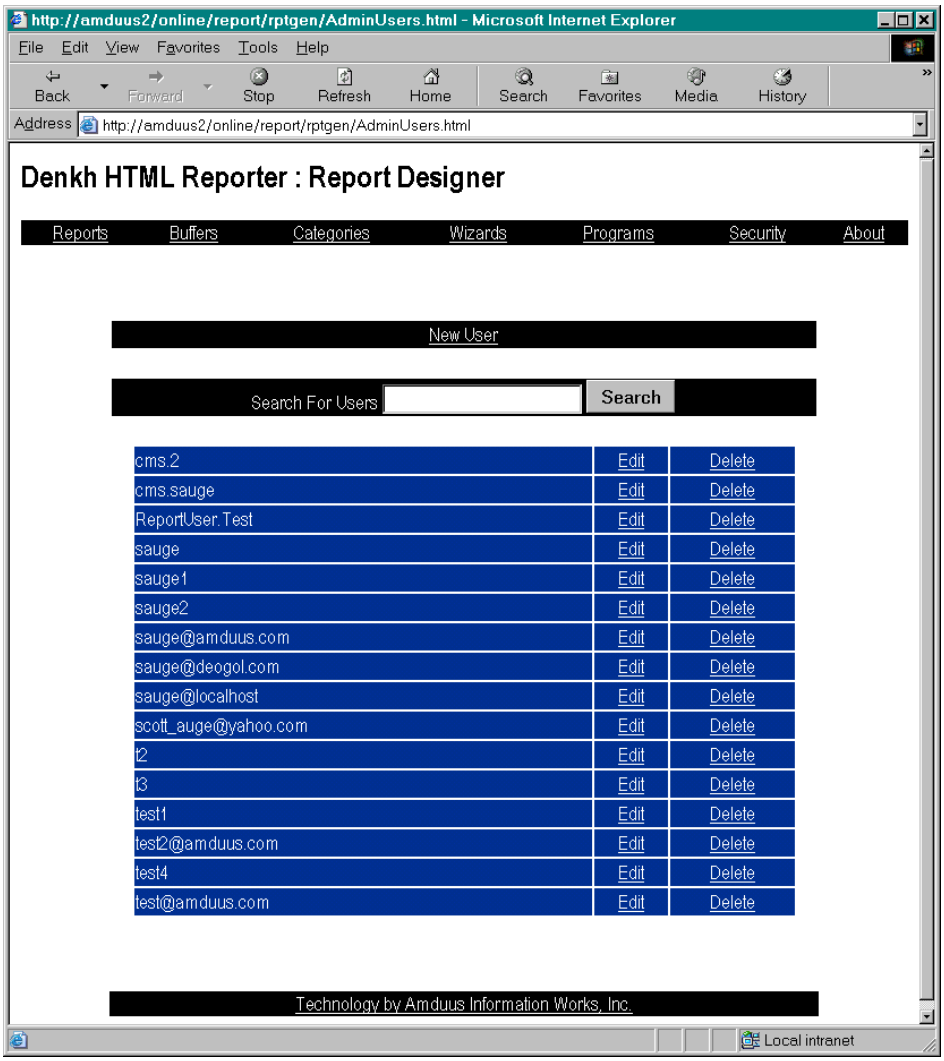


To create a new profile, merely enter a name for the profile that you would use in the @secure() macro. The name should not have (or) or , characters. Spaces are not recommended, but can be used.

To delete a profile, merely click the associated Delete hyperlink next to the profile. When you delete a profile, you do NOT delete the users associated with that profile. You do NOT delete any reports associated with that profile by the @secure() macro either.

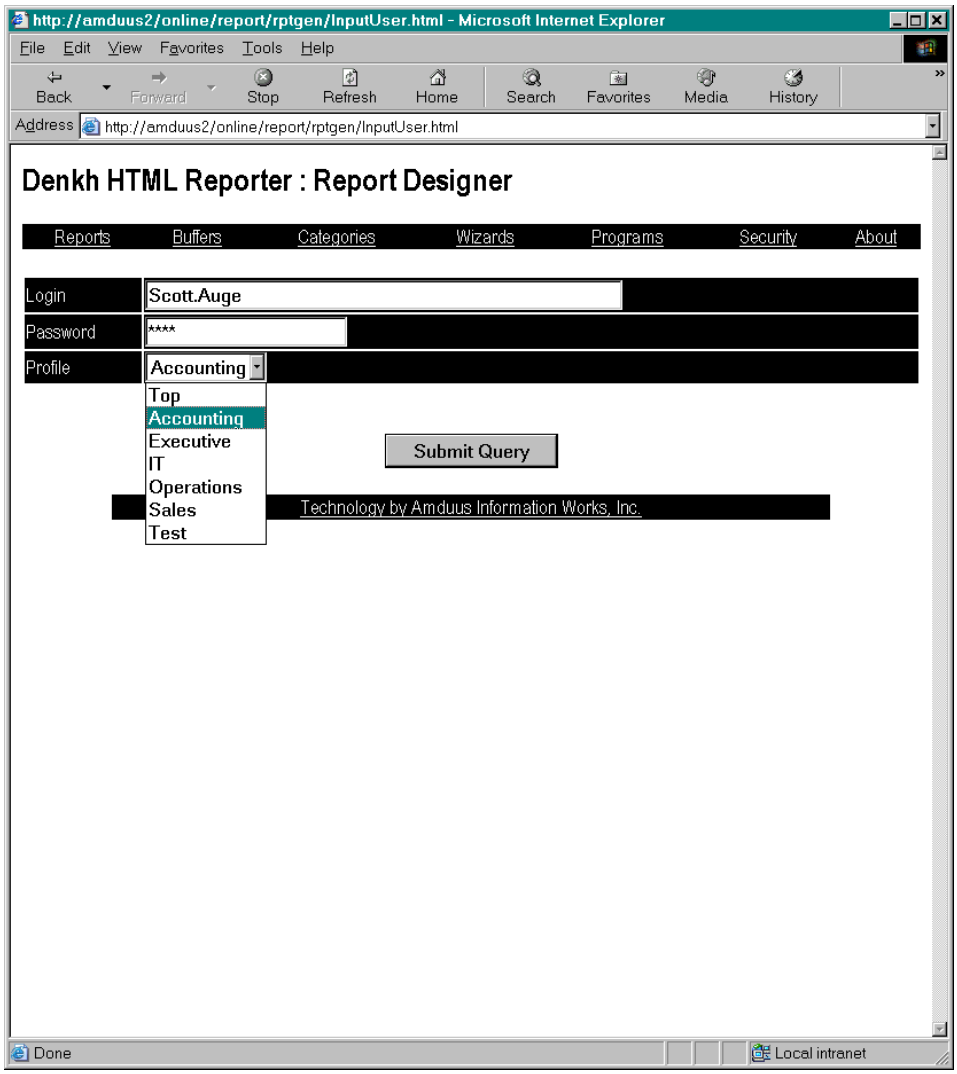
Users

By clicking on the Administrate Users hyperlink, you will be presented with a screen similar to the one below:



With this screen you can create users and modify or delete existing users.

To create a user click on the New User hyperlink at the top. You will be presented with a screen similar to below:



In the Login field, you would name the login for a given user. I suggest using the FirstName.LastName profile as shown above. Otherwise you may consider using employee numbers.

The password field will contain a word or phrase that the person logging in will need to know to continue using the application with secure reports. By only the user knowing his/her password means only that user will be able to view secure reports within their profile.

A user can be a member of only one profile. Profiles are discussed in the preceding section.

Using the @secure() macro

To secure a report, you need to use the @secure() macro in the Page Header section of the report. A report without the @secure() macro is viewable by everyone with access to the HTML Reporter program.

The macro has the following syntax:

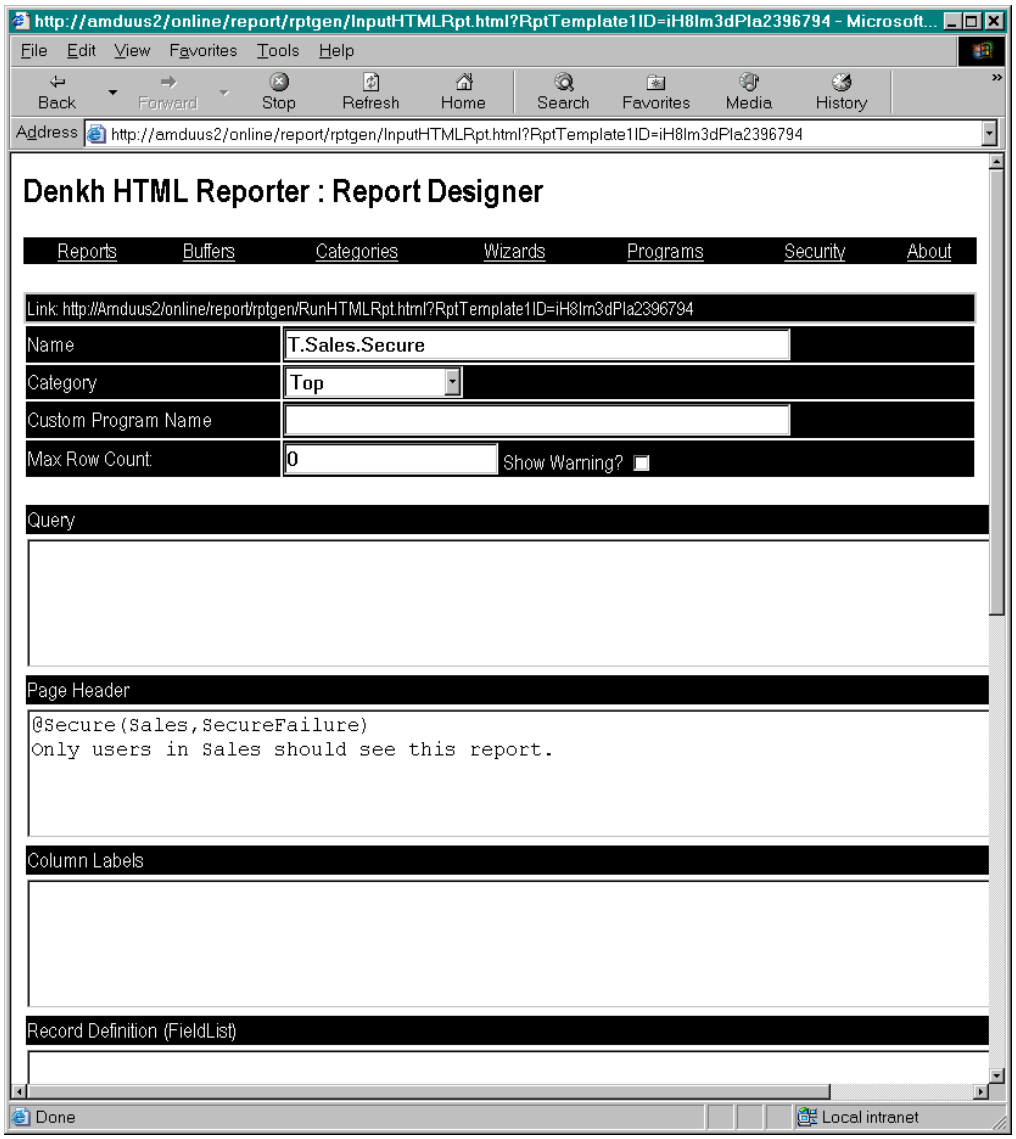
@secure(Profiles,FailReportName)

The FailReportName is a report you would define in the Report editor of Report Designer. This allows you to have different screens for different reports. Such screens might have phone numbers of people to contact to gain access to the report, or a reason why the user is not allowed to use the report. HTML Reporter allows this for customizability for your portal environment.

The Profiles section refers to those profiles that may *view* the report. For example, Scott.Auge is user who is a member of the Accounting profile. Should the Profiles section include Accounting as a required profile for viewing, Scott.Auge as well other users in that profile can access the results of the report.

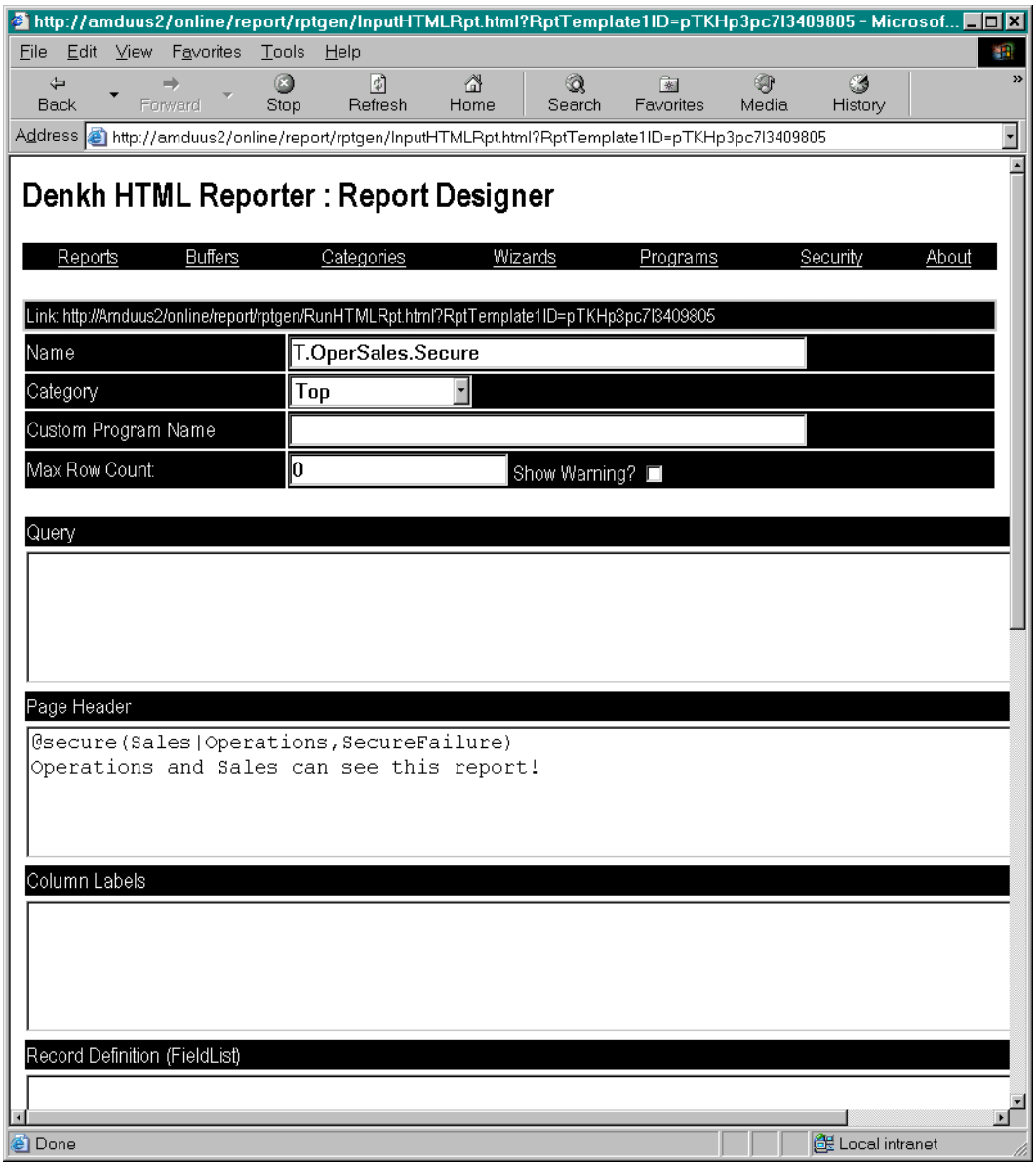
Let's look at some samples:

Here is a report that should be viewable only by users in the Sales profile.



Note that the @secure() macro can ONLY be used in the Page Header section of the report. In this example, note that if the user running the report is not in the Sales profile, the report called SecureFailure will be run instead.

But perhaps you have a need for reports to be accessible to more than one group. Then one would call the @secure() macro with a listing of profiles that may view it's results separated by a | symbol as shown below:



Note how the @secure() macro has Sales|Operations in the profiles section. This means that the report may be accessed by users in the Sales OR the Operations profiles. If a user tries to access the report who is not a member of Sales or Operations would be sent to the SecureFailure report defined by the implementor.

Using @secure() in an embedded report

You can use @secure() in embedded reports. Should the current user not have access to see the report, the embedded report will not be present. If they do, they will be able to see the report.

Using the @sessionid() macro

This section is of interest to programmers who might be using the @prg() macro.

When a user logs into the system, a cookie of random characters is set on their browser called SessionID. This value is matched to a WebState record that notes the login associated with that cookie value.

In order to reach this cookie in your report, you would use the @sessionid() macro. The @sessionid() macro can be used in the page header, page footer, and records section of the report.

To call out to a program that may need to know who is calling it, you would use something like:

```
@prg(helpers/checkuser.p,@sessionid()|@fld(Order.OrderNumber))
```

This would send the session id from the browser into your program. From there you can use the following code to pull up the name of the login associated with that login:

```
RUN state/ReadState.p (INPUT cSessionID,
                       INPUT "Denkh.Identification",
                       INPUT "Login",
                       OUTPUT cLogin).
```

where cSessionID would be a variable you defined and populated as documented in the @prg() macro text. (See HTML Reporter document.) The cLogin varible would contain the name of the user login associated with that sessionid. You can then use the WebState table to store information you may need to pass between the user's report calls.

```
  RUN state/WriteState.p (INPUT cSessionID,
                          INPUT "YourCategory",
                          INPUT "YourValueName",
                          INPUT YourCharacterVarible).
```

Implementation Notes!!!

After defining your reports, users, and profiles, you should let the user's access the system via the Login.html page. It should look similar to this:

```
https://yourhost/cgi-bin/webspeedscript/rptgen/Login.html
```

Since the application is Open Source, you can change the look and feel of the screen easily.

This page will post into the PrcLogin.html screen. At the moment, upon successful login, the user is redirected to the report designer. This is probably something you will NOT want, as they can change reports, profiles, and their own security settings.

To change where the user's go after that, change the following line:

```
ASSIGN cReDirectURL = "AdminHTMLRpt.html".
```

to something more appropriate such as:

```
ASSIGN cReDirectURL = "HTMLRpt.html?RptTemplate1ID=djskdsiowu2392".
```

Where that report that acts as a report listing screen. (This way you can list reports with a report and thereby have that much more customization available to your site's look and feel.)