The Progress Electronic Magazine

In this issue:

Did you sign up to receive this E-Zine? Send email to scott_auge@yahoo.com to subscribe! It's free!

Though intended for users of the tools provided by Progress Software Corporation, this document is NOT a product of Progress Software Corporation.

Main Article:

Safe Hyperlinks referencing records on the internet

Written by Scott Auge scott_auge@yahoo.com

Often we are in need to find records based on a hyperlink provided on the internet.

The hyperlink can include the key information to look up the record on click of the link. This information might be data within the record to help with a unique find, or by using a string representation of the RowID of the record.

There is a problem with this method however. Users with bad intentions, or mere curiosity can save the web page onto their hard drive. The purpose of this is to allow editing of the page so that new values can be placed into the hyperlink. Doing so, they may stumble upon data that they have no business looking at.

Hence, your website has been breached to expose information to people who do not have the proper authorization to see it.

A way to remedy the problem is to introduce a special field into the table that will have records that need to be identified on the internet. This special field's value should not be consecutive in nature, unique to identify the record, and should be difficult to guess at.

The solution is to develop a means of creating unique random strings. Below is a program that will accept an argument as to how many letters should be output, as well the actual output of a random string. The larger the random string, the more dispersed the data will be and thus harder to guess at. However, large data will slow down transaction speed between the browser and the application server. As well, the data might burden disk space.

```
/*
 * RandomString.p
 * Written by Scott Auge
 * Create a random string of the alphabet and digits of the length specified
 * by the caller.
 * Note that the -rand parameter is very important with this function when
 * used on the web.  Else a new number will not be created until all the agents
 * have processed this function.
 *
 */

DEF INPUT PARAMETER pLength   AS INTEGER NO-UNDO.
DEF OUTPUT PARAMETER pString  AS CHARACTER NO-UNDO.

DEF VAR lAlphabet AS CHARACTER NO-UNDO.
DEF VAR i         AS INTEGER NO-UNDO.

ASSIGN
lAlphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"
i = 0
pString = "".

DO WHILE i < pLength:

  ASSIGN
  i = i + 1
  pString = pString + SUBSTRING(lAlphabet, RANDOM(1,LENGTH(lAlphabet)), 1).

END.
```

Once we have a random value, we are not really guaranteed that value is unique to the system. It is very probable, based on the size of the value, that it is unique, but this is not mathematically guaranteed. To do so, we need to add to the string, a value that is from a unique source.

The following code incorporates a call to RandomString.p with a number that is passed into it as an argument. The function then returns the string with the number.

```
/* MakeSeq.i
 * Written by Scott Auge
 * Function that will create a unique sequence ID for the records created
 * within the system.  Reason for this instead of just an integer is so that
 * this ID can be placed on to the web in an URL and not worry about someone
 * fiddling with the values on a saved web page to access other records in
 * the database.
 *
 */

FUNCTION MakeSeq RETURNS CHARACTER (INPUT Seq AS INTEGER):

  DEF VAR i AS CHARACTER NO-UNDO.

  RUN api/web/RandomString.p (INPUT 10, OUTPUT i).

  ASSIGN i = i + STRING (Seq).

  RETURN i.

END. /* FUNCTION MakeID() */
```

This function should receive as it's input a number from a database sequence that is used for that table. This is why the function does not reference a sequence it's self. Else, we would be burning up sequences for multiple tables, when we could have a sequence for each table. This allows the number of records to be marked this way to increase within the scope of the database.

Lets look at an implementation of this idea.

First we will look at the creation of the record:

```
CREATE CreditCard.

ASSIGN
CreditCard.CreditCardID = MakeID(NEXT-VALUE(CreditCardID)).

ASSIGN
CreditCard.Number = GET-VALUE("CCNumber")
CreditCard.ExpireMonth = GET-VALUE("CCMnthExp")
CreditCard.ExpireYear = GET-VALUE("CCYearExp")
CreditCard.Type = GET-VALUE("CCType").
```

Note that the code has a call MakeID() on a separate ASSIGN statement. Under certain conditions, having the call to the function will cause a interesting compile error.

Next we look at E4GL code that would generate the link that one would use to choose the credit card they wish to use.

```
<!--WSS
FOR EACH CreditCard NO-LOCK
WHERE CreditCard.Owner = lOwner:
-->
<a
href="http://www.yourdomain.com/online/example/ChooseCard.html?CCID=` CreditCard.CreditCardID`">`CreditCard.Type`</a>
<!--WSS
END. /* FOR EACH CreditCard */
-->
```

The above code allows the user to choose which credit card to use, should they have more than one card within the system.

As one could probably tell, having a value on the link above that can be easily manipulated to represent another user's card would be a "bad thing." The attacker could attempt to change the number at the end of the string which is consecutively assigned, but then they would need to guess at the random "salt" portion of the string that would accompany that false number.

Once the user clicks the link, we need to see how to use that value to find the appropriate record. This is quite simple actually, as shown below:

```
FIND CreditCard NO-LOCK
WHERE CreditCard.CreditCardID = GET-VALUE("CCID").
```

One might want to include some error messages back at the user should the CCID not relate to any credit card records. As well, you may wish to log the IP and user login associated with the badly structured data, as it appears someone may have been fiddling with it.

About the author: Scott Auge is the founder of Amduus Information Works. He has been programming in the Progress environment since 1994. His work have included E-Business initiatives and focuses on web applications on UNIX platforms.
scott_auge@yahoo.com

Product Updates and Releases:

Do you have a new product or release of software – let me know and it will be included in the latest E-Zine.

Jobs Available:

Do you have a job opening available? Let me know and it will be included in the latest E-Zine. Price is $10.00 per listing per issue.

Contractors/Consulting Companies Available:

If you do work in the Progress world – let me know and I will be able to include you! Price is $10.00 per listing per issue.

Amduus Information Works
4506 Carlyle Court Suite 712
Santa Clara, CA 95054
408-980-8447
scott_auge@yahoo.com

Creation of modules and products for re-sale. Internet/Intranet programming for E-Business in the marketing/manufacturing/service/law enforcement industries.

Jay A. Martin
DataChase, LLC
www.data-chase.com
info@data-chase.com

Providing contract programming and writing services.

Community Announcements:

A place to announce Progress User Groups, Open Source Exchanges, etc. No charge!

Publishing Information:

Scott Auge publishes this document. I can be reached at scott_auge@yahoo.com.

This electronic magazine is to be published once a month.

Article Submission Information:

Currently, payment for an article is not possible. But – articles CAN enhance your prestige and name recognition – and these things can translate into money!

Please submit your article in Word or text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Upcoming Articles:

Saving states in stateless web applications – May 2001

Using Parameter records for easier configuration of your application – June 2001

Sending and receiving email with your progress application – July 2001