SHARE 2001 Technical Conference

RACF PKIServ SPE - Demonstration

March 1, 2001 - Session 1744

Jim Sweeny
tel: (845) 435-7453
email: jsweeny@us.ibm.com

Agenda

Overview

Demo

- Screen Cam
- Implementation details

Overview - Description

- A web based Certificate Authority application that utilizes RACF to generate and deliver certificates to end users.
- Provided as an SPE on OS/390 V2R10
  - RACF APAR OW45211, PTF UW74164
  - SAF APAR OW45212, PTF UW74113
- Roughly Equivalent to the IBM HTTP Server's CaServlet

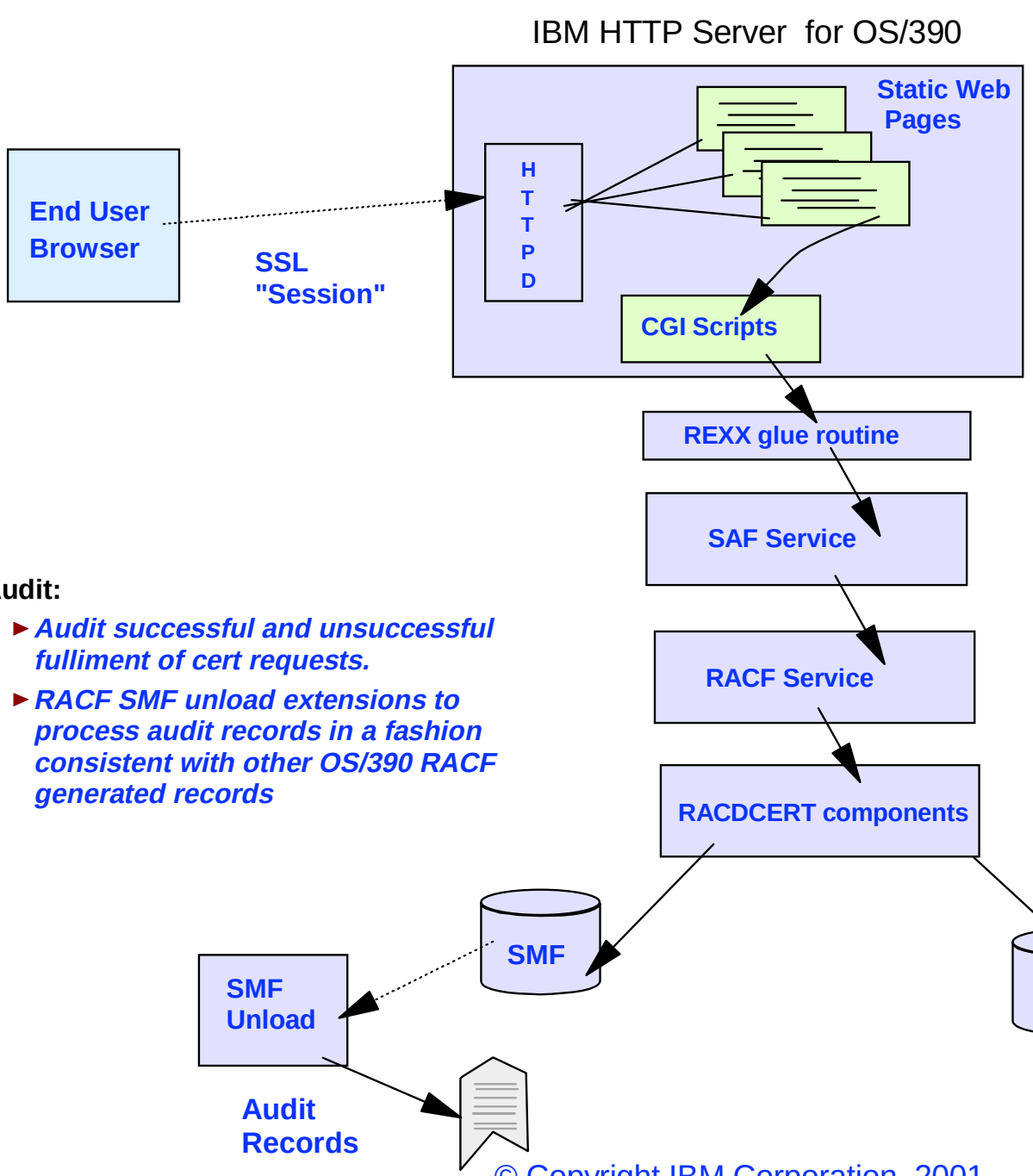
Overview - PKI Services SPE Contents

- New SAF Callable Service R_PKIServ (IRRSPX00)
  - Affects BCP component which requires a co-requisite SAF APAR/PTF
- RACF Glue Routine for IRRRPX00 (IRRRPXGL)
  - Allows REXX program to call new SAF callable service
- SMF Unload - Support two new record types for GENCERT and EXPORT
- Sample materials
  - Downloadable from RACF webpage - http://www.s390.ibm.com/products/racf/webca.html
    - HTML pages and certificate templates contained in a conf file
    - Connector REXX execs (CGIs) to process the above
- Documentation
  - PTF documentation shipped as a samplib part, IRR45211
    - Has RACF pub updates only (SAG, Callables Services, etc)
  - Downloadable from RACF webpage - http://www.s390.ibm.com/products/racf/webca.html
    - Install and use information including pointers to Web Server admin/doc for setup considerations supplied by ITSO security redbook.

Overview - Setup / Usage Concepts

- System Programmer installs PTF
- RACF Administrator creates RACF profiles to protect the callable service and RACDCERT functions
- RACF Administrator also creates a CA certificate and private key to use for signing
- Web Administrator configures the IBM HTTP webserver and customizes the web pages for the PKISERV application
- Web users point their browsers to the main URL either directly or via a link from elsewhere
  - May request certificates for their browsers or other servers
  - Always prompted for userid/password before request is submitted
- REXX CGIs process the request info and invoke the new R_PKIServ Callable Service For GENCERT
- Callable Service invokes RACF's RACDCERT GENCERT functionality
  - If successful, certificate is created on the spot and a transaction ID is returned
- User follows revtrieval web pages to pickup certificate
- REXX CGIs process the transaction ID and invoke the new R_PKIServ Callable Service For EXPORT
- Returned web page either install certificate into the user's browser or allows the user to save the certificate elsewhere.
- Certificate remains in RACF mapped to the user's userid

Overview - Structure



- **"Sample" Web Pages & Scripts:**
  - *Shipped as part of RACF's sample materials -- down loadable from RACF product related web site.*
    - *Manual install -- Doc provides guidance and install/planning info.*
  - *Allows for customer tailoring to suite needs.*
  - *Same model being followed as was introduced with the cert auto - registration SPE*

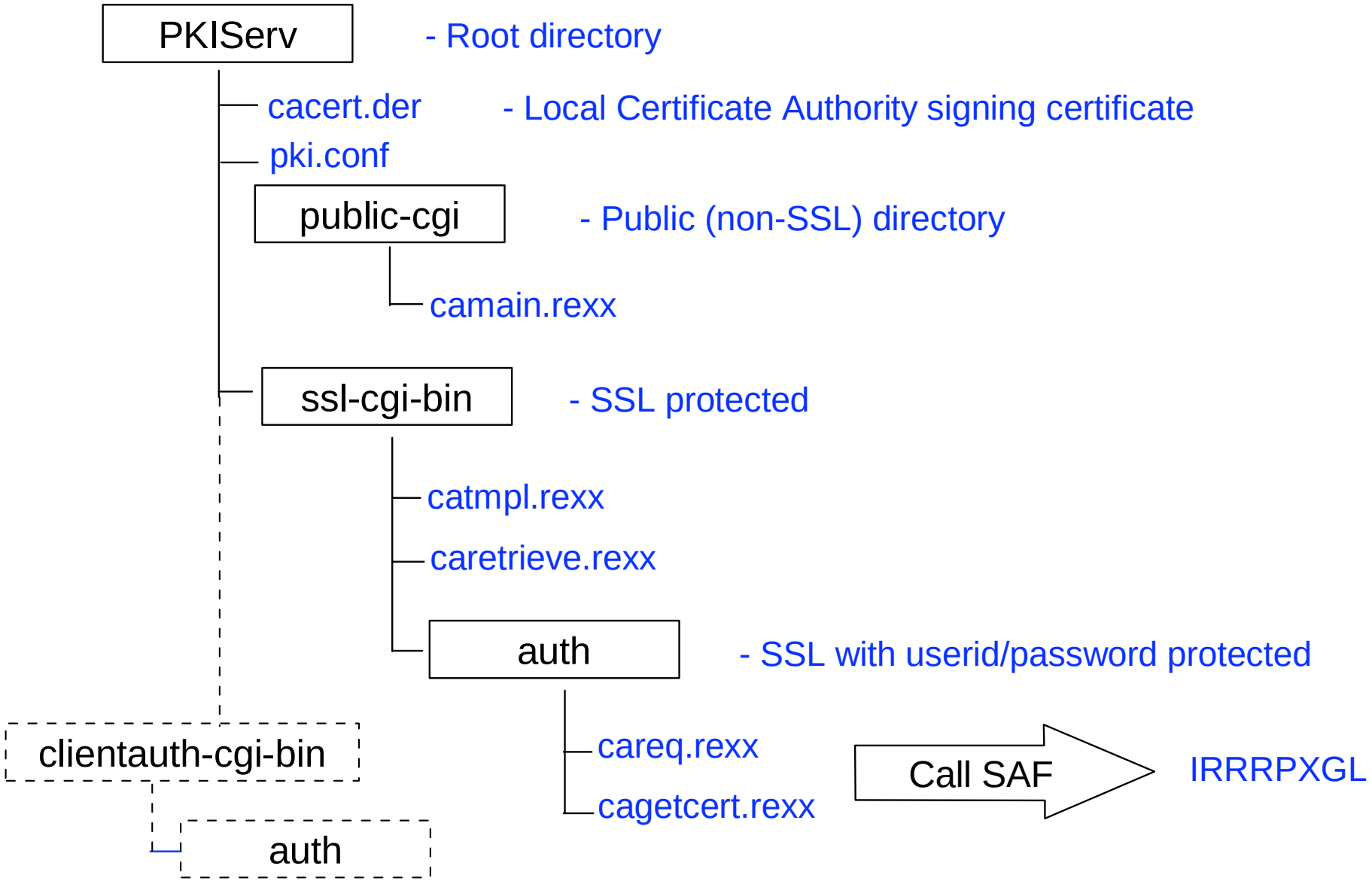
- **SAF Service: PKIServ SAF callable service**
  - *RACF provides support for this new service. Limited number of function code(s) which are directly related to the fulfillment of a given cert request.*
  - *Implementation of this service will re-use RACDCERT "primitives"*

- **Audit:**
  - *Audit successful and unsuccessful fulliment of cert requests.*
  - *RACF SMF unload extensions to process audit records in a fashion consistent with other OS/390 RACF generated records*

OS/390 and/or OS/390 Security Server Component

New Function -- Shipped as samples

Webserver Directory Structure



The Conf File - Certificate Templates

- **pki.conf - The conf file contains a mixture of true HTML and HTML like tags. The main tags divide the file into sections, APPLICATION, TEMPLATE, and INSERT, where APPLICATION and TEMPLATE may contain various subsections, named fields, and substitution variables:**
  - **<APPLICATION> ... </APPLICATION> - Lists the certificate templates to be supported by a given application**
    - **<CONTENT> ... </CONTENT> - Contains HTML for the main page. The certificate templates supported are identified by named fields**
  - **<TEMPLATE> ... </TEMPLATE> - Describes the fields that make up one certificate template**
    - **<CONTENT> ... </CONTENT> - Contains the HTML for the second page. Lists the fields modifiable by the user**
    - **<APPL> ... </APPL> - Lists the fields to be supplied by the application. For PKISERV, UserId is the only supported field**
    - **<CONSTANT> ... </CONSTANT> - Lists the hardcoded certificate field values**
    - **<SUCCESSCONTENT> ... </SUCCESSCONTENT> - Contains the HTML to be presented if the request was successful**
    - **<FAILURECONTENT> ... </FAILURECONTENT> - Contains the HTML to be presented if the request was not successful**
    - **<RETRIEVECONTENT> ... </RETRIEVECONTENT> - Contains the HTML to be presented to enable retrieving the certificate**
    - **<RETURNCERT> ... </RETURNCERT> - Contains the HTML to be presented once the certificate has been retrieved.**

Define Your CA Certificate in RACF

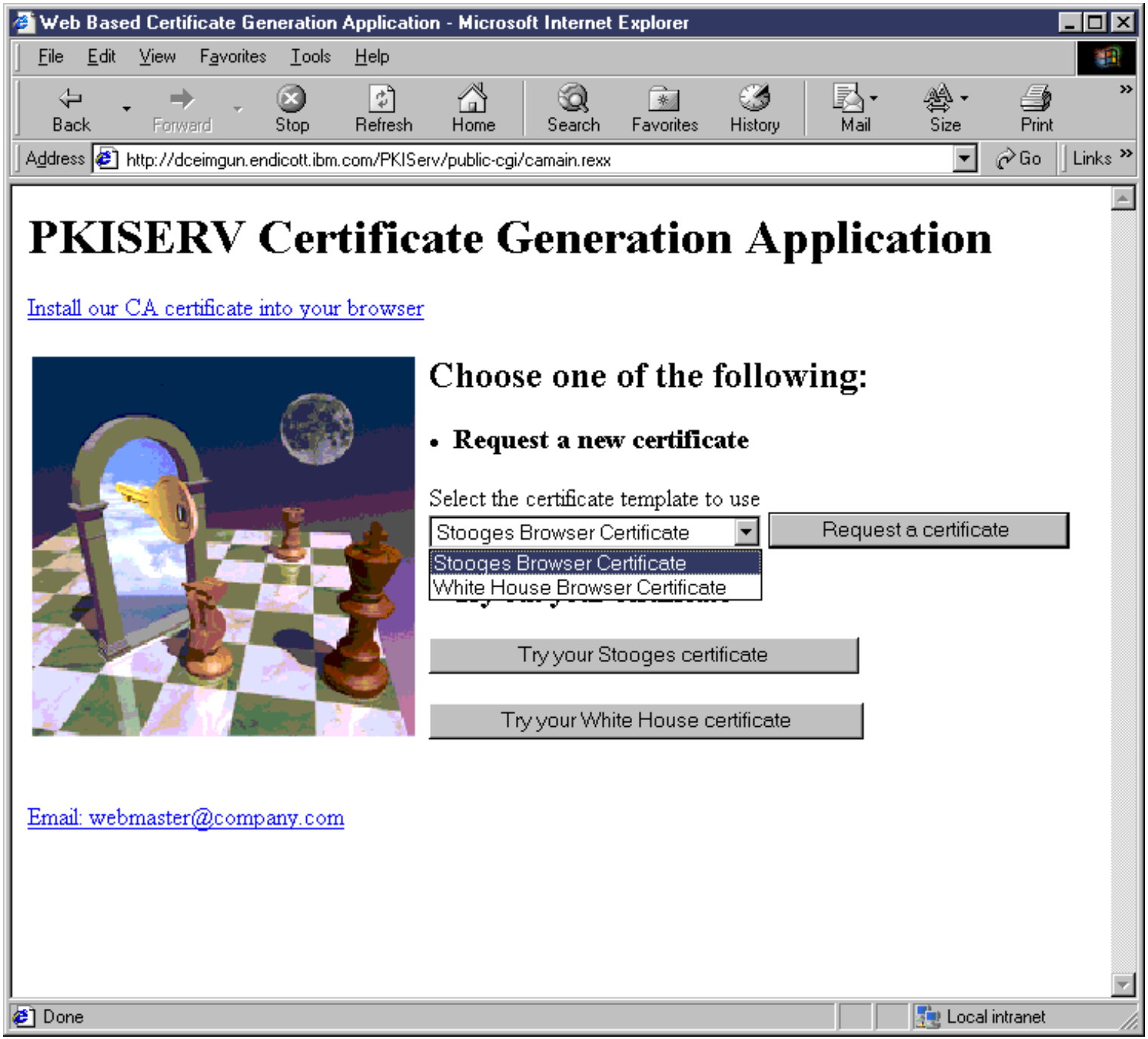
RACDCERT GENCERT CERTAUTH WITHLABEL('Local SAF CA')
SUBJECTSDN(OU('Jim''s RACF CA') O('IBM') C('US'))
NOTBEFORE(DATE(2000/01/01)) NOTAFTER(DATE(2011/04/30))

RACDCERT LIST(LABEL('Local SAF CA')) CERTAUTH

```
Digital certificate information for CERTAUTH:

  Label: Local SAF CA
  Certificate ID: 2QiJmZmDhZmjgdOWg4GTQOLBxkDDwUBA
  Status: TRUST
  Start Date: 2000/01/01 00:00:00
  End Date:   2011/04/30 23:59:59
  Serial Number:
       >00A8<
  Issuer's Name:
       >OU=Jim's RACF CA.O=IBM.C=US<
  Subject's Name:
       >OU=Jim's RACF CA.O=IBM.C=US<
  Key Usage: CERTSIGN
  Private Key Type: Non-ICSF
  Private Key Size: 1024
  Ring Associations:
  *** No rings associated ***
```

Demo - Stooges and The White House



Sample Application

```
<APPLICATION NAME=PKISERV>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Application </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>PKISERV Certificate Generation Application</H1>
<A HREF="/PKIServ/cacert.der">Install our CA certificate into your browser </A>
<p>
<table border=0><tr><td>
<IMG SRC="/PKIServ/chess.gif" BORDER=0 WIDTH=250 HEIGHT=250><BR><BR>
</td><td></td><td><H2>Choose one of the following:</H2>
<li><h3>Request a new certificate</h3>
<FORM name=mainform METHOD=GET ACTION="/PKIServ/ssl-cgi/catmpl.rexx">
Select the certificate template to use<br>
<SELECT NAME="Template">
 %%Stooges Browser Certificate%%
     <OPTION>Stooges Browser Certificate
 %%White House Browser Certificate%%
     <OPTION>White House Browser Certificate
</SELECT>
<INPUT TYPE="submit" VALUE=" Request a certificate  ">
</FORM>
<li><h3>Try out your certificate</h3>
<FORM name=admform METHOD=GET ACTION="/PKIServ/clientauth-cgi/auth/stooge.rexx">
<INPUT TYPE="submit" VALUE="    Try your Stooges certificate    ">
</FORM>
<FORM name=admform METHOD=GET ACTION="/PKIServ/clientauth-cgi/auth/whitehouse.rexx">
<INPUT TYPE="submit" VALUE="Try your White House certificate">
</FORM>
</td></tr></table>
<p> %%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
</APPLICATION>
```

The SELECT creates the template choice dropdown

Trying the certificate invokes SSL client authentication through 2nd webserver

Sample Template - Stooges

```
TEMPLATE NAME=Stooges Browser Certificate>
CONTENT>
HTML><HEAD><TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE></HEAD>
BODY>
H1> [tmplname] </H1>
H2>Choose one of the following:</H2>
p><ul><h3><li>Request a New Certificate</h3>
FORM NAME="CertReq" METHOD=POST ACTION="/PKIServ/ssl-cgi-bin/auth/careq.rexx">
INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
%%PublicKey[browsertype]%%
INPUT TYPE="reset" VALUE="Clear">
/FORM>
p><H3><li>Pick Up a Previously Issued Certificate</H3>
FORM NAME="PUCert" METHOD=GET ACTION="/PKIServ/ssl-cgi-bin/caretrieve.rexx">
INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
INPUT TYPE="submit" VALUE="Retrieve your certificate">
/FORM>
/ul><p><A HREF="mailto:webmaster@company.com">Email: webmaster@company.com</A>
/BODY></HTML>
/CONTENT>
APPL>
%%UserId%%
/APPL>
CONSTANT>
%%KeyUsage=handshake%%
%%NotAfter=365%%
%%OrgUnit=Stooge template certificate%%
%%Org=Stooge Certs Inc%%
%%Locality=Poughkeepsie%%
%%StateProv=New York%%
%%Country=US%%
%%SignWith=SAF:CERTAUTH/Local SAF CA%%
%%CommonName=%%
/CONSTANT>
.. more sections ...
/TEMPLATE>
```

Public key created by the browser

CGI provides the User ID

Remaining info hardcoded

Sample Template - White House

```
<TEMPLATE NAME=White House Browser Certificate>
<CONTENT>
<HTML><HEAD><TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE></HEAD>
<BODY>
<H1> [tmplname] </H1>
<H2>Choose one of the following:</H2>
<p><ul><h3><li>Request a New Certificate</h3>
<FORM NAME="CertReq" METHOD=POST ACTION="/PKIServ/ssl-cgi-bin/auth/careq.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
 %%PublicKey[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p><H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM NAME="PUCert" METHOD=GET ACTION="/PKIServ/ssl-cgi-bin/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul><p><A HREF="mailto:webmaster@company.com">Email: webmaster@company.com</A>
</BODY></HTML>
</CONTENT>
<APPL>
 %%UserId%%
</APPL>
<CONSTANT>
 %%KeyUsage=handshake%%
 %%NotAfter=365%%
 %%Org=Federal Government%%
 %%Locality=Washington%%
 %%StateProv=DC%%
 %%Country=US%%
 %%SignWith=SAF:CERTAUTH/Local SAF CA%%
 %%CommonName=%%
</CONSTANT>
... more sections ...
</TEMPLATE>
```

Public key created by the browser

CGI provides the User ID

Remaining info hardcoded.
OrgUnit not specified

Add EXIT logic to careq.rexx

```
csrc= 0
if env._PKISERV_EXIT ^= "" then do
  /* There is an installation exit. Build the pre-exit command string */
  ecmd= "'" || env._PKISERV_EXIT || "'"
  do i= 1 to eps.0
     ecmd= ecmd || ' eps.' || i
  end
  address syscall 'pipe p.'   /* make a pipe to catch the exit's output  */
  ecmd= ecmd  "'>/dev/fd' || p.2"
  interpret ecmd /* Make the exit call */
  csrc= rc
  address syscall 'close' p.2         /* close write end of pipe   */
  address mvs 'execio * diskr' p.1 '(stem mep.'
                                      /* read data in pipe     */
  address syscall 'close' p.1          /* close read side too     */
end
if csrc = 0 then do
  /* Exit is allowing the request. Add modifications if any */
  do i= 1 to mep.0
     cmd= cmd || ' mep.' || i
  end
end
if csrc = 8 then do
  /* Exit has denied the request */
  errorinfo.0= mep.0
  errorinfo.0= 1
  errorinfo.1= "Request denied by installation exit. RC =" csrc
  csrc= 99
end
/* Now invoke the command if we should */
if csrc = 0 then do
  interpret cmd
  csrc= rc
end
```

Build EXIT command string from input parms

Invoke EXIT. Catch output in pipe

Success. Add EXIT provided parms

Fail if EXIT says "bad"

Invoke SAF if the EXIT says "good"

The EXIT - A UNIX Executable

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
main(int argc, char * argv[]) {
   int i, rc, needs_protection = 0;
   char user[9];

   user[0]= '\0';
   for (i=1;i<argc;i++) {
      if (strncmp(argv[i],"Template=",9) == 0)
        if (strcmp(argv[i]+9,"White House Browser Certificate") == 0)
          needs_protection = 1;
      if (strncmp(argv[i],"UserId=",7) == 0)
        strcpy(user,argv[i]+7);
   }
   if (needs_protection) {
      /* Is client a former or current President*/
      if (0 == (rc= __check_resource_auth_np(NULL,NULL,
                  user,"FACILITY","USPRESIDENT",__READ_RESOURCE))) {
        /* Is client the current president */
        if (0 == __check_resource_auth_np(NULL,NULL,
                    user,"FACILITY","USPRESIDENT",__UPDATE_RESOURCE))
          printf("OrgUnit=White House Staff\n");
        else
          printf("OrgUnit=White House Alumni\n");
      }
      else /* Not a president */
        exit(8);
   }
}
```

Parms input through argv in name=value form

The "White House Browser Certificate" is the only template protected

User's access is checked via a UNIX syscall. Current President has UPDATE access while former Presidents have only READ. Organizational Unit is set accordingly

Directives - Webserver 1

```
sslmode    on
sslport    443
normalmode  on

Protection PublicUser {
        ServerId        PublicUser
        UserID          PUBLIC
        Mask            Anyone
}
Protect /PKIServ/public-cgi/*  PublicUser
Protect /PKIServ/ssl-cgi-bin/*  PublicUser
Protect /PKIServ/*  PublicUser

Protection AuthenticatedUser {
        ServerId        AuthenticatedUser
        AuthType        Basic
        PasswdFile      %%SAF%%
        UserID          %%CLIENT%%
        Mask            All
}
Protect /PKIServ/ssl-cgi-bin/auth/* AuthenticatedUser

Protection SurrogateUser {
        ServerId        SurrogateUser
        AuthType        Basic
        PasswdFile      %%SAF%%
        UserID          PKISERV
        Mask            All
}
Protect /PKIServ/ssl

Redirect  /PKIServ/ssl-cgi/*  https://<server-domain-name>/PKIServ/ssl-cgi-bin/*
Redirect  /PKIServ/clientauth-cgi/* https://<server-domain-name>:1443/PKIServ/clientauth-cgi-bin/*
```

EXPORT runs under client's User ID

GENCERT runs under PKISERV User ID

Redirect to 2nd webserver for SSL client Authentication

Directives - Webserver 2

```
ssl mode    on
ssl port    1443
normal mode  off

Protection AuthenticatedClient {
        ServerId        AuthenticatedClient
        AuthType        Basic
        UserID          %%CERTIF%%
        SSL_CLIENTAUTH  Client
        Mask            Anyone
}
Protect /PKIServ/clientauth-cgi-bin/auth/* AuthenticatedClient

Protection WhiteHouseClient {
        ServerId        WhiteHouseClient
        AuthType        Basic
        UserID          %%CERTIF%%
        SSL_CLIENTAUTH  Client
        Organization    "Federal Government"
        Mask            Anyone
}
Protect /PKIServ/clientauth-cgi-bin/auth/whitehouse* WhiteHouseClient
```

These CGIs runs under client's User ID as determined by certificate

Ditto except subject's organization must be "Federal Government"

Test login CGI - stooge.rexx

```
/* REXX */
'cgiutils -ct text/html'
env.= ""
do i=1 to __environment.0
  parse var __environment.i varname '=' data
  env.varname = data
end
/* Put these details into stem printablecert */
printablecert.0 = 7
printablecert.1= "<b>Serial #:" env.HTTPS_CLIENT_CERT_SERIAL_NUM"</b>"
printablecert.2=  "<b>CN=" env.HTTPS_CLIENT_CERT_COMMON_NAME"</b>"
printablecert.3=  "<b>OU=" env.HTTPS_CLIENT_CERT_ORG_UNIT"</b>"
printablecert.4=  "<b>O=" env.HTTPS_CLIENT_CERT_ORGANIZATION"</b>"
printablecert.5=  "<b>L=" env.HTTPS_CLIENT_CERT_LOCALITY"</b>"
printablecert.6=  "<b>SP=" env.HTTPS_CLIENT_CERT_STATE_OR_PROVINCE"</b>"
printablecert.7=  "<b>C=" env.HTTPS_CLIENT_CERT_COUNTRY"</b>"
say "<h1>Stooge Information:</h1>"
say "<table border>"say "<tr><td>" printablecert.1 "</td><td rowspan=7>"
if env._BPX_USERID = "MOE" Then
  say '<img src="/PKIServ/clientauth-cgi-bin/test/moe.jpg">'
else if env._BPX_USERID = "LARRY" Then
  say '<img src="/PKIServ/clientauth-cgi-bin/test/larry.jpg">
else if env._BPX_USERID = "CURLY" Then
  say '<img src="/PKIServ/clientauth-cgi-bin/test/curly.jpg">'
else do
  say '<FORM name=admform METHOD=GET ',
'ACTION="http://dceimgun.endicott.ibm.com//PKIServ/clientauth-cgi/auth/whitehouse.rexx">'
  say "<b>----- Client is not a stooge -----</b><p>"
  say '<INPUT TYPE="submit" VALUE="Goto the White House">'
end
say "</td></tr><tr><td>" printablecert.2 "</td></tr>"
say "<tr><td>" printablecert.3 "</td></tr>"
say "<tr><td>" printablecert.4 "</td></tr>"
say "<tr><td>" printablecert.5 "</td></tr>"
say "<tr><td>" printablecert.6 "</td></tr>"
say "<tr><td>" printablecert.7 "</td></tr></table>"
return 0
```

Certificate info contained in environment vars

_BPX_USERID contains userid that certificate maps to

Test login CGI - whitehouse.rexx

```
/* REXX */
'cgiutils -ct text/html'
env.= ""
do i=1 to __environment.0
  parse var __environment.i varname '=' data
  env.varname = data
end
/* Put these details into stem printablecert */
printablecert.0 = 7
printablecert.1= "<b>Serial #:" env.HTTPS_CLIENT_CERT_SERIAL_NUM"</b>"
printablecert.2=  "<b>CN=" env.HTTPS_CLIENT_CERT_COMMON_NAME"</b>"
printablecert.3=  "<b>OU=" env.HTTPS_CLIENT_CERT_ORG_UNIT"</b>"
printablecert.4=  "<b>O=" env.HTTPS_CLIENT_CERT_ORGANIZATION"</b>"
printablecert.5=  "<b>L=" env.HTTPS_CLIENT_CERT_LOCALITY"</b>"
printablecert.6=  "<b>SP=" env.HTTPS_CLIENT_CERT_STATE_OR_PROVINCE"</b>"
printablecert.7=  "<b>C=" env.HTTPS_CLIENT_CERT_COUNTRY"</b>"
say "<h1>White House Information:</h1>"
say "<table border>"
say "<tr><td>" printablecert.1 "</td><td rowspan=7>"
say '<img src="/PKIServ/clientauth-cgi-bin/test/whitehouse.jpg">'
say "</td></tr>"
say "<tr><td>" printablecert.2 "</td></tr>"
say "<tr><td>" printablecert.3 "</td></tr>"
say "<tr><td>" printablecert.4 "</td></tr>"
say "<tr><td>" printablecert.5 "</td></tr>"
say "<tr><td>" printablecert.6 "</td></tr>"
say "<tr><td>" printablecert.7 "</td></tr>"
say "</table>"
return 0
```

Certificate info contained in environment vars

_BPX_USERID not used

References

- RACF Website: http://www.s390.ibm.com/racf/
  - PKISERV - http://www.s390.ibm.com/products/racf/webca.html
- RACF Manuals:
  - RACF Command Language Reference (SC28-1919)
  - RACF Security Administrator's Guide (SC28-1915)
  - RACF Callable Services Guide (SC28-1921)
- IBM HTTP Server Manuals:
  - Planning, Installing, and Using (SC31-8690)
- Other Sources:
  - PKIX
    - http://www.ietf.org/html.charters/pkix-charter.html
  - X.509v3 Certificate Format
    - http://www.entrust.com/s97is.vts (search for 'X.509 Certificate')