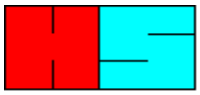


Handout zum Workshop

Einleitung

Eine AS/400 / iSeries ist im lokalen Netz sowie im Internet stets der Gefahr ausgesetzt, dass auf die enthaltenen Daten Personen zugreifen, die hierzu nicht berechtigt sind.
Um dieser Gefahr vorzubeugen, sollte man alle offenen Tore schliessen, und an die benötigten Tore eine Wache aufstellen, die eine „Gesichtskontrolle“ für alle Anwender durchführt.

Im Folgenden finden Sie eine Auswahl an Möglichkeiten, die in erster Linie als Denkanstösse und Richtungsweiser dienen sollen.

Keines der gezeigten Programmbeispiele soll als Lehrstück für Programmierung oder die beste Lösung darstellen. Der geneigte Leser wird im Laufe der Zeit während der Experimente feststellen, dass IBM viele Möglichkeiten geschaffen hat, aus der iSeries einen „Fels in der Brandung“ zu machen.

Bitte beachten Sie, dass einige der gezeigten Programme dem Nutzer einen Zugriff auf gewünschte Daten verbieten. Wenn Sie beim Programmieren einen Fehler machen, kann es sein, dass niemand mehr per ODBC, FTP oder andere Zugangswege an seine Daten kommt. Es wird daher empfohlen, auf einem Produktivsystem mit grösster Vorsicht vorzugehen, oder – wenn vorhanden – ein Testsystem zu verwenden.

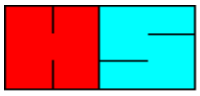
Holger Scherer, im Februar 2004

Analysieren Sie Ihre Benutzer und deren Profile!

Beispiel-CL-Programm `ANZUSRPRF`:

Dieses kleine Programm gibt Informationen über alle Benutzerprofile in eine Datei in `QTEMP` aus und ruft ein verarbeitendes RPG-Programm auf.

```
PGM

DLTF        FILE(QTEMP/USRPRF)
MONMSG      MSGID(CPF0000)

DSPUSRPRF   USRPRF(*ALL) OUTPUT(*OUTFILE) +
              OUTFILE(QTEMP/USRPRF)

CALL        PGM(ANZUSRRPG)
MONMSG      MSGID(CPF0000)

ENDPGM
```



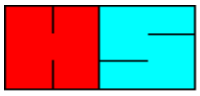
Beispiel-RPG-Programm ANZUSRRPG:

In diesem RPG-Programm können wir viele interessante Prüfungen und Auswertungen mit den Daten aus den Benutzerprofil-Informationen durchführen. Zur besseren Übersicht wurden hier keine APIs verwendet, so dass auch Einsteiger in die Thematik sich schnell einarbeiten können.

```
FMT F   FDateiname+IPEASFSlän+LSlän+AIE/AEinh.Schlüsselwörter+++++++++++++++++
0001.00 FUSRPRF    IF   E             DISK
0002.00 FQPRINT    O    F   80        PRINTER
0003.00
FMT D   DName++++++++++ETDsVon++++Bi/L+++IDG.Schlüsselwörter+++++++++++++++++
0004.00 DMeld             S              80A
0005.00 DDruckzeile       S              80A
0006.00
FMT CX  CL0N01Faktor1+++++++Opcode&ExtErweiterter-Faktor2++++++++++++++++++++++
0007.00 C                   DOW       %EOF(USRPRF) = *OFF
0008.00 C                   READ      USRPRF

         * Nur Benutzer prüfen, die nicht mit Q anfangen (Systemprofile)
0009.00 C                   IF        %SUBST(UPUPRF : 1 : 1) <> 'Q'
0010.00 C
0011.00  * *** Prüfe auf nicht USRCLS(*USER)
0012.00 C                   IF        UPUSCL <> '*USER'
         * (Somit sehen wir priviligierte Benutzer)
0013.00 C                   EVAL      Druckzeile = UPUPRF + ' ist nicht *USER'
0014.00 C                   EXCEPT    Zeile
0015.00 C                   ENDIF
0016.00  * (Ganz interessant: wer hat Rechte auf alle Objekte?)
0017.00  * *** Prüfe auf SPCAUT(*ALLOBJ)
0018.00 C                   IF        %SCAN('ALLOBJ' : UPSPAU) > 0
0019.00 C                   EVAL      Druckzeile = UPUPRF + ' hat *ALLUSR'
0020.00 C                   EXCEPT    Zeile
0021.00 C                   ENDIF
0022.00
0023.00 C                   ENDIF
0024.00 C                   ENDDO
0025.00
0026.00 C                   EVAL      *INlr = *ON
0027.00
          * (Eine ganz einfache Druckausgabe)
0028.00 OQPRINT    E            Zeile
0029.00 O                       Druckzeile          80
```

Hier ist viel Platz für eigene Ideen und weitere Prüfungen, was die Parameter des Benutzerprofils anbelangt. Sei es Plattenplatzbelegung oder Aktivität. Dieses Programm dient nur als erste Idee und erhebt gewiss keinen Anspruch auf Vollständigkeit oder gutes Design :-)

Die Ausgabe von DSPUSRPRF bietet bereits eine Vielzahl an Informationen, die Sie auswerten können. Schauen Sie sich die Feldbeschreibungen mit DSPFFD an, oder analysieren Sie die vorhandenen Daten mit SQL oder Query.



Ein erstes Experiment können Sie auch mit folgender CL-Routine durchführen:

```
0001.00 PGM
0002.00               DCLF       FILE(QTEMP/USER)
0003.00
0004.00               DSPUSRPRF  USRPRF(*ALL) OUTPUT(*OUTFILE) +
0005.00                            OUTFILE(QTEMP/USER)
0006.00
0007.00  LOOP:        RCVF
0008.00               MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(ENDE))
0009.00               IF         COND(&UPUSCL *NE '*USER') THEN(DO)
0010.00               SNDMSG MSG('Benutzer ' *CAT &UPUPRF *CAT ' ist  +
0011.00                in Klasse ' *CAT &UPUSCL) TOUSR(QSECOFR)
0012.00                ENDDO
0013.00               GOTO       CMDLBL(LOOP)
0014.00
0015.00  ENDE:        ENDPGM
```

Vor dem Umwandeln bitte per Hand den Befehl aus Zeilen 4 und 5 eingeben, damit der Compiler die Struktur der temporären Datei kennt.

Erstellen Sie ein Audit-Journal, wenn Ihre Maschine schnell ist

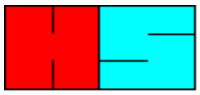
Die AS/400 / iSeries bietet Ihnen die Möglichkeit, alle Tätigkeiten an Objekten, sicherheitsrelevante Daten und Vorgänge aufzuzeichnen. Hierfür gibt es Systemwerte, die bestimmte Journale steuern. Je nach Einstellung der Protokollierung fallen *sehr viele Sätze* an, was den Plattenplatz und die CPU belastet!

Journalempfänger erstellen: (zur Aufnahme der Protokollsätze)

```
CRTJRNRCV JRNRCV(QGPL/QAUDRCV) THRESHOLD(10000) AUT(*EXCLUDE)
TEXT('Empfänger für Systemjournal')
```

Journal erstellen:

```
CRTJRN JRN(QSYS/QAUDJRN) JRNRCV(QGPL/QAUDRCV) MNGRCV(*SYSTEM)
```



Systemwerte anpassen (nach Bedarf):

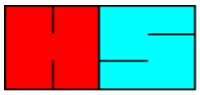
QAUDCTL : *OBJAUD oder *AUDLVL sowie *NOQTEMP

*OBJAUD besagt, dass Sie die zu überwachenden Objekte selbst mit dem CL-Kommando CHGOBJAUT bestimmen.
*AUDLVL besagt, dass alle Objekte auf Ereignisse überwacht werden, die im folgenden Systemwert angegeben sind.
*NOQTEMP besagt, dass fast alle Objekte in QTEMP nicht protokolliert werden.

QAUDLVL : *AUTFAIL *PGMFAIL *SAVRST
*SECURITY *SERVICE

*AUTFAIL protokolliert alle Berechtigungsfehler wie Zugriffsfehler bei Anmeldung oder einer Berechtigungsprüfung, sowie Fehler bei der Benutzeranmeldung.
*PGMFAIL prüft Fehler bei Programmausführung, wie geblockte Anweisungen generell, Domänenfehler (wichtig bei Wechsel auf Security Level 50!)
SAVRST Wenn Ihre Benutzer RST verwenden dürfen, können Sie hiermit kontrollieren, ob Programme zurück gespeichert werden, die die Berechtigungen des Objekteigners übernehmen, ob sich Eigner bei Objekten ändern, ob Jobbeschreibungen auf spezielle Benutzerprofile eingerichtet sind oder ob Benutzerprofile zurückgespeichert werden.
*SECURITY dies protokolliert alle sicherheitsrelevanten Aktionen: Ändern von Objekt-Berechtigungen, Eigner-Änderungen, Änderungen an einem Programm, um das Eignerrecht zu übernehmen, Zurücksetzen von DST-Kennwörtern etc.
*SERVICE protokolliert spezielle Service-Befehle und -APIs.

Mit diesen Überwachungseinstellungen werden auf einem aktiven System schon sehr viele Aktionen überwacht, und das resultierende Journal wird schnell gross und unübersichtlich.



Journal überwachen!

Dies ist die aufwändigste Arbeit. Ein AUDJRN kann sehr schnell sehr gross werden!

`DSPJRN QAUDJRN`
Anzeige aller Sätze im Journal (mit Parametern den Zeitraum eingrenzen). Dies dient dem ersten Stöbern in den Einträgen, bei einem System unter Last gibt es aber sehr viele Einträge, auch das Blättern belastet das System!

Wenn Sie nun wie oben beschrieben das Journal betrachten, sehen Sie mindestens zwei Einträge:

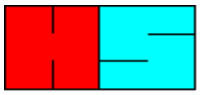
```
                          Journaleinträge anzeigen

Journal  . . . . . :   QAUDJRN         Bibliothek . . . . . :   QSYS

Auswahl eingeben und Eingabetaste drücken.
  5=Gesamten Eintrag anzeigen


Ausw.    Folge   Code  Art   Objekt      Bibliothek  Job         Uhrzeit
  _          1   J     PR                            PCHOLLE1    14:27:59
  _          2   T     SV                            PCHOLLE1    14:29:35
```

Mit der Auswahl für lassen Sie sich die beiden Einträge anzeigen. Im folgenden Bildschirm wird Code und Art genauer beschrieben. Interessant ist der zweite Eintrag:

```
                          Journaleintrag anzeigen

Objekt . . . . . :                   Bibliothek . . . . . :
Teildatei  . . . . . :
Unvollständige Daten :   Nein            Minim. Eintragsdaten :   Nein
Folge  . . . . . :   2
Code . . . . . :   T  - Prüfprotokolleintrag
Art  . . . . . :   SV - Änderung an Systemwert

            Eintragsspez. Daten
Spalte      *...+....1....+....2....+....3....+....4....+....5
00001      'AQAUDCTL   *AUDLVL   *NOQTEMP                    '
```

Sie sehen hier sofort, dass der Systemwert `QAUDCTL` verändert wurde. Mit der Taste **`F10`** sehen Sie weitere Details zu diesem Eintrag, unter anderem auch Job und Username.

`DSPJRN JRN(QAUDJRN) RCVRNG(*CURCHAIN) JRNCDE((T)) ENTTYP(CD)`
Zeigt nur ausgeführte Command-Zeilen (Typ `CD`, Eintragscode `T`) aus dem aktuellen Journalempfänger an . Zum besseren Analysieren diesen Befehl mit dem Parameter `OUTPUT (*PRINT)` laufen lassen, am besten im Batch!
(Siehe unten: Sie müssen für die Benutzer das Protokollieren der Commands erst einschalten, bevor Sie diese Einträge im Journal sehen)



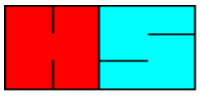
Journal archivieren

Wenn Sie Ihr Journal archivieren (Stichwort: Audits), kann man das zum Beispiel so lösen:

```
DSPJRN      JRN(QAUDJRN) RCVRNG(*CURCHAIN) +
              FROMTIME(&DATUM 000000) TOTIME(&DATUM +
              235959) JRNCDE(*ALL) ENTTYP(*ALL) +
              OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4) +
              OUTFILE(AUDIT/AUDJRN) OUTMBR(*FIRST *ADD) +
              ENTDTALEN(*CALC)
```

Rufen Sie diesen Befehl in einem Nachtjob auf (Zeiten anpassen!) So können Sie später weitere Auswertungen auf die Daten fahren, falls es zu Nachforschungen kommt.
In diesem Beispiel wird in einem CL-Programm die Variable `&DATUM` verwendet, um das anzuzeigende Datum zu bestimmen.

Mit dieser Datei können Sie schon sehr viele Auswertungen erstellen, nur Ihre Phantasie setzt den Möglichkeiten Grenzen.
Auch das obige Beispiel sollte möglichst im Batch laufen (evtl. für den Nachtjob das Protokollieren durch Ändern der Systemwerte ausschalten, um die Geschwindigkeit zu erhöhen).



Eintragsarten im Systemjournal (Eintragscode T)

SV	Änderung an einem Systemwert
CD	Ausgeführter CL-Befehl / Command (sehr interessant!)
ZR	Lesezugriff auf ein Systemobjekt (sollte ausgeklammert werden, da dies das Journal unnötig aufbläht!)
PW	Anmeldeinformationen ungültig
SF	Spooldatei – allgemeine Tätigkeit
AF	Berechtigungsfehler (Authority failure)
CA	Berechtigungsänderung (change authority)
OW	Änderung eines Eigners an einem Objekt (owner)

Bitte beachten: bei der Anzeige von Zugriffen auf IFS-Objekte wird auf einen Pointer verwiesen, der in einem Programm aufgelöst werden muss. Hierzu wird zu späterem Zeitpunkt ein RPG-Programm veröffentlicht.

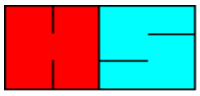
Bei Systemen mit hoher Last dürfte folgendes helfen, um die Zugriffe auf alle Objekte in QSYS *nicht* zu protokollieren (hier sollten sowieso nur Leserechte bestehen):

```
CHGOBJAUD OBJ(QSYS/*ALL) OBJTYPE(*ALL) OBJAUD(*NONE)
```

Alle Benutzer zum Audit anpassen:

```
PGM
             DCLF       FILE(QTEMP/USRPRF)
             DLTF       FILE(QTEMP/USRPRF)
             MONMSG     MSGID(CPF0000)
             DSPUSRPRF  USRPRF(*ALL) OUTPUT(*OUTFILE) +
                          OUTFILE(QTEMP/USRPRF)

LESE:        RCVF
             MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(ENDE))
             CHGUSRAUD  USRPRF(&UPUPRF) AUDLVL(*CMD *SECURITY *SAVRST)
             GOTO       CMDLBL(LESE)
ENDE:        SNDPGMMSG  MSG('Fertig!')
ENDPGM
```

Obiges Beispiel ist etwas „hart“, da hier für jeden Benutzer eingestellt wird, dass Befehle Protokolliert werden, sowie alle `SAVE` und `RESTORE` Tätigkeiten, sowie Aufrufe von System-APIs mit Sicherheitsfunktionen. Für die Systemprofile (mit `Q` beginnend) dürfte das nur in den wenigsten Fällen Sinn machen. Überwachen Sie statt dessen lieber einzelne Benutzerprofile, oder spezielle Profile, wenn Sie Probleme mit einer Software nach der Erhöhung von `QSECURITY` haben.

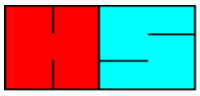


Exitpoints benutzen!

Exitpoints bieten für viele Serveraktionen einen Einstiegspunkt, bei dem ein Prüfprogramm *vor* Ausführung der angeforderten Aktion weitere Berechtigungsprüfungen vornehmen kann. Mit diesen Exit-Points und den dort hinterlegten Programmen kann mittels `WRKREGINF` gearbeitet werden.
Ein kleines Beispiel: (der Einfachheit halber entnommen dem IBM Information Center)

QIBM_QZDA_SQL1 Exit-Point für ODBC-SQL Interface 1

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                Beispiel Exit-Programm für ODBC                          */
/*                                                                         */
/*    Ausgangspunkt-Name      : QIBM_QZDA_SQL1                             */
/*    Programmname            : LOGSQL                                     */
/*    Beschreibung            : Dieses Programm lehnt bestimmte Benutzer   */
/*                              bei einer ODBC-Abfrage ab.                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
PGM PARM(&STATUS &REQUEST)

/*  Deklaration Programm-Parameter                                         */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  DCL VAR(&STATUS) TYPE(*CHAR) LEN(1)  /* Anzeiger: Anfrage erlaubt/n.E. */
  DCL VAR(&REQUEST) TYPE(*CHAR) LEN(2000) /* Parameter: Anfrage          */

/*  Einzelne Werte der Parameterstruktur                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  DCL VAR(&USER) TYPE(*CHAR) LEN(10)  /* Aufruf von welchem Userprofil?  */
  DCL VAR(&SRVID) TYPE(*CHAR) LEN(10) /* Servertyp (immer: „*SQL“)  )    */
  DCL VAR(&FORMAT) TYPE(*CHAR) LEN(8) /* Parameterformat („ZDAI0100“)    */
  DCL VAR(&FUNC) TYPE(*CHAR) LEN(4)   /* ausgef. Funktion (immer 0)      */
  DCL VAR(&SQL)  TYPE(*CHAR) LEN(512) /* Inhalt SQL Anfrage              */

/*  Entnehme die einzelnen Parameter aus der Struktur                      */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  CHGVAR VAR(&USER)   VALUE(%SST(&REQUEST  1 10))
  CHGVAR VAR(&SRVID)  VALUE(%SST(&REQUEST 11 10))
  CHGVAR VAR(&FORMAT) VALUE(%SST(&REQUEST 21  8))
  CHGVAR VAR(&FUNC)   VALUE(%SST(&REQUEST 28  4))
  CHGVAR VAR(&SQL)    VALUE(%SST(&REQUEST 96 512))

/*  Hauptprogramm, hier ist Platz für Experimente!                         */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  /* zunächst erlauben wir mal generell jedem alles...                    */
  CHGVAR VAR(&STATUS) VALUE('1')

  /* Wenn das Benutzerprofil = GAST ist, wird die Anfrage nicht gestattet*/
  IF COND(&USER *EQ 'GAST') THEN( CHGVAR VAR(&STATUS) VALUE('0') )

  /* Weiteres */
  /* ...      */

  SNDMSG MSG(&SQL) TOMSGQ(QGPL/LOGSQL)     /* Protokollieren */
EXIT:
ENDPGM
```



Vor dem Aufruf dieses Beispielprogrammes LOGSQL erstellen Sie eine eigene Nachrichtenwarteschlange, die alle Protokollierungen empfangen soll:

```
CRTMSGQ MSGQ(QGPL/LOGSQL) TEXT('Protokoll SQL')
```

Wandeln Sie nun das CL-Programm um. Exit-Programme sollten nach Möglichkeit dem Benutzer QSECOFR gehören, unter Benutzerprofil *OWNER laufen, und niemand sonst darf Rechte an diesem Objekt haben. Spätestens wenn Sie in einem Exitprogramm eine Protokollierung in eine Datei einbauen, die sonst niemand auf dem System sehen darf, müssen Sie so vorgehen:

```
CHGOBJOWN OBJ(QGPL/LOGSQL) OBJTYPE(*PGM) NEWOWN(QSECOFR)

CHGPGM PGM(QGPL/LOGSQL) USRPRF(*OWNER)

CHGAUT OBJ('/qsys.lib/qgpl.lib/logsql.pgm') USER(*PUBLIC) DTAAUT
(*EXCLUDE) OBJAUT(*NONE)
```

Nicht vergessen dürfen wir, dass der Benutzer `QUSER` (unter dessen Profil der SQL-Server läuft) das Programm ausführen darf:

```
CHGAUT OBJ('/qsys.lib/qgpl.lib/logsql.pgm') USER(QUSER) DTAAUT(*RX)
```

Nun können Sie mit `WRKREGINF` das Programm an den Exitpoint hängen. Mit Auswahl 8 vor den Exitpoint-Namen sehen Sie eventuell vorhandene Programme. Fügen Sie das neue Programm hinzu. Wenn Sie mit dem Operations Navigator arbeiten, werden Sie schon staunen...

Zugriff auf ODBC über Authorisierungsliste

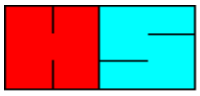
Wenn Sie nur eine gewisse Personengruppe für ODBC berechtigen wollen, erstellen Sie eine Authorisierungsliste:

```
CRTAUTL AUTL(ODBCUSER)
```

Bearbeiten Sie die Liste:

```
WRKAUTL ODBCUSER
```

Fügen Sie die gewünschten User mit der Berechtigung `*USE` hinzu.



Prüfen Sie dann in obigen Programm nach der Stelle `/* Weiteres */` wie folgt:

```
RTVAUTLE   AUTL(ODBCUSER) USER(&USER) USE(&AUTH)
IF         COND(&AUTH *NE '*USE') THEN(DO)
CHGVAR     VAR(&STATUS) VALUE('0')
```

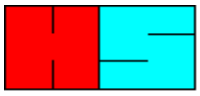
(`&AUTH` ist eine Variable und muss als `*CHAR 10` definiert sein)

```
 DCL        VAR(&AUTH) TYPE(*CHAR) LEN(10)
```

Hier wird geprüft, ob das Benutzerprofil auf die angegebene Authorisierungsliste das `*USE` Recht hat. Wenn nicht, wird vom Exit-Programm der Wert "0" zurückgegeben und die Anfrage somit verweigert.

Hinweis:

Ganz neugierige Administratoren prüfen den Inhalt der SQL-Anfrage.
Definieren Sie hierzu den Parameter &REQUEST auf z.B. 2000 Zeichen. Sie finden dort ab Stelle 96 die SQL-Anfrage. Man könnte nun abfragen, ob dort "DELETE" oder "ALTER" oder "UPDATE" vorkommt und entsprechend handeln. :-)



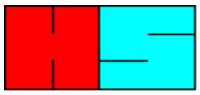
FTP-Zugriffe beobachten und steuern

Noch ein gutes Beispiel für ein Exitprogramm ist der Exit-Punkt QIBM_QTMF_SERVER_REQ.

Das hierfür nötige Programm ist schon etwas aufwändiger, da ein FTP-Server verschiedene Funktionalitäten bieten kann. Zunächst das Programm:

Zunächst viele Variablendeklarationen, da der Exit-Point viele Informationen übergibt und die Struktur der Server-Anfrage auch sehr aussagekräftig ist: (FTPLOGS)

```
PGM         PARM(&APPIDB &OPIDB &USRPRF &IPADDR &IPLENB +
                 &OPINFO &OPLENB &ALLOWOPB)

DCL         VAR(&APPIDB)    TYPE(*CHAR) LEN(4)
DCL         VAR(&OPIDB)     TYPE(*CHAR) LEN(4)
DCL         VAR(&USRPRF)    TYPE(*CHAR) LEN(10)
DCL         VAR(&IPADDR)    TYPE(*CHAR) LEN(15)
DCL         VAR(&IPLENB)    TYPE(*CHAR) LEN(4)
DCL         VAR(&OPINFO)    TYPE(*CHAR) LEN(999)
DCL         VAR(&OPLENB)    TYPE(*CHAR) LEN(4)
DCL         VAR(&ALLOWOPB) TYPE(*CHAR) LEN(4)

DCL         VAR(&APPID)     TYPE(*DEC)  LEN(1)
DCL         VAR(&OPID)      TYPE(*DEC)  LEN(3)
DCL         VAR(&IPLEN)     TYPE(*DEC)  LEN(3)
DCL         VAR(&OPLEN)     TYPE(*DEC)  LEN(3)
DCL         VAR(&ALLOWOP)   TYPE(*DEC)  LEN(1)
DCL         VAR(&IPADDRX)   TYPE(*CHAR) LEN(15)
DCL         VAR(&OPINFOX)   TYPE(*CHAR) LEN(999)

DCL         VAR(&NEVER)     TYPE(*DEC)  LEN(1)  VALUE(-1)
DCL         VAR(&REJECT)    TYPE(*DEC)  LEN(1)  VALUE(0)
DCL         VAR(&ALLOW)     TYPE(*DEC)  LEN(1)  VALUE(1)
DCL         VAR(&ALWAYS)    TYPE(*DEC)  LEN(1)  VALUE(2)

DCL         VAR(&APP)       TYPE(*CHAR) LEN(1)
DCL         VAR(&OP)        TYPE(*CHAR) LEN(4)
DCL         VAR(&TEXT)      TYPE(*CHAR) LEN(256)
DCL         VAR(&D)         TYPE(*CHAR) LEN(30)
DCL         VAR(&DATUM)     TYPE(*CHAR) LEN(6)
DCL         VAR(&ZEIT)      TYPE(*CHAR) LEN(6)
DCL         VAR(&MSGQ) TYPE(*CHAR) LEN(10) VALUE(FTPLOGS)
DCL         VAR(&MSGQLIB)  TYPE(*CHAR) LEN(10) VALUE(QGPL)
 MONMSG      MSGID(CPF0000 MCH0000)

 CHGVAR      VAR(&APPID)     VALUE(%BIN(&APPIDB))
 CHGVAR      VAR(&OPID)      VALUE(%BIN(&OPIDB))
 CHGVAR      VAR(&IPLEN)     VALUE(%BIN(&IPLENB))
 CHGVAR      VAR(&OPLEN)     VALUE(%BIN(&OPLENB))
 CHGVAR      VAR(&APP)       VALUE(&APPID)
 CHGVAR      VAR(&OP)        VALUE(&OPID)
```



Nun prüfen wir an Hand der Operation-ID-Variable, welche Aktion der Nutzer durchführt. Zunächst wird aber zu Dokumentationszwecken die Variable &OP belegt.

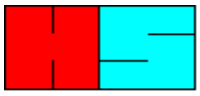
```
IF   COND(&OPID *EQ 0) THEN(CHGVAR VAR(&OP) VALUE(INIT))
IF   COND(&OPID *EQ 1) THEN(CHGVAR VAR(&OP) VALUE(CRTL))
IF   COND(&OPID *EQ 2) THEN(CHGVAR VAR(&OP) VALUE(DLTL))
IF   COND(&OPID *EQ 3) THEN(CHGVAR VAR(&OP) VALUE(CURL))
IF   COND(&OPID *EQ 4) THEN(CHGVAR VAR(&OP) VALUE(LSTL))
IF   COND(&OPID *EQ 5) THEN(CHGVAR VAR(&OP) VALUE(DLTF))
IF   COND(&OPID *EQ 6) THEN(CHGVAR VAR(&OP) VALUE(SNDF))
IF   COND(&OPID *EQ 7) THEN(CHGVAR VAR(&OP) VALUE(RCVF))
IF   COND(&OPID *EQ 8) THEN(CHGVAR VAR(&OP) VALUE(RNMF))
IF   COND(&OPID *EQ 9) THEN(CHGVAR VAR(&OP) VALUE(EXEC))

IF         COND(&IPLEN *GE 1) THEN(DO)
CHGVAR     VAR(&IPADDRX) VALUE(%SST(&IPADDR 1 &IPLEN))
ENDDO
IF         COND(&OPLEN *GE 1) THEN(DO)
CHGVAR     VAR(&OPINFOX) VALUE(%SST(&OPINFO 1 &OPLEN))
ENDDO
```

Hier wird eine Nachricht formatiert, die an eine Nachrichtenwarteschlange (siehe oben, blau) gesendet wird. Alternativ könnte man hier eine Routine einbauen, die diese Informationen in eine Datei schreibt.

```
RTVSYSVAL  SYSVAL(QDATE) RTNVAR(&DATUM)
RTVSYSVAL  SYSVAL(QTIME) RTNVAR(&ZEIT)

IF         COND(&OPID *EQ 0) THEN(DO)
CVTDAT     DATE(&DATUM) TOVAR(&D) FROMFMT(*SYSVAL) +
             TOFMT(*SYSVAL) TOSEP(*SYSVAL)
CHGVAR     VAR(&D) VALUE(&D *BCAT %SST(&ZEIT 1 2) *CAT +
             ':' *CAT %SST(&ZEIT 3 2) *CAT ':' *CAT +
             %SST(&ZEIT 5 2))
ENDDO

CHGVAR     VAR(&ALLOWOP) VALUE(&ALLOW)

CHGVAR     VAR(&TEXT) VALUE(&APP *BCAT &IPADDRX *BCAT +
             &OP *BCAT &USRPRF *BCAT &D *BCAT &OPINFOX)

CHGMSGQ    MSGQ(&MSGQLIB/&MSGQ) MSGQFULL(*WRAP)
MONMSG     MSGID(CPF2403) EXEC(DO)
CRTMSGQ    MSGQ(&MSGQLIB/&MSGQ) MSGQFULL(*WRAP)
ENDDO
SNDPGMMSG  MSGID(CPF9897) MSGF(QCPFMSG) MSGDTA(&TEXT) +
             TOMSGQ(&MSGQLIB/&MSGQ)
```

Nachsatz: Anhand `&APPID` kann man auch prüfen, um welche Serverart es sich handelt:

0 = FTP-Client
1 = FTP Server
2 = REXEC Server
3 = TFTP Server

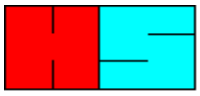


Der Benutzer HACKER wird generell vom Zugriff auf FTP-Funktionen ausgesperrt!

```
    /* sperren */
               IF          COND(&USRPRF *EQ 'HACKER') THEN(DO)
                 CHGVAR VAR(&ALLOW) VALUE(&NEVER)
               SNDMSG      MSG('Verweigert für ' *CAT &USRPRF) +
                             TOMSGQ(&MSGQLIB/&MSGQ)
               ENDDO
               CHGVAR      VAR(&ALLOWOP) VALUE(&ALLOW)

  ENDE:        CHGVAR      VAR(%BIN(&ALLOWOPB)) VALUE(&ALLOWOP)
               ENDPGM
```

Hinweise:

- Sie könnten generell den EXEC-Befehl aussperren, in dem Sie einfach jedesmal, wenn die Operation 9 angefordert wird, der Wert 0 (oder &REJECT, siehe oben grau) zurückgegeben wird. Im groben Fall geben Sie -1 (oder &NEVER) zurück und die Sitzung wird für den Angreifer wertlos.
- Auch hier besteht die Möglichkeit, viel mehr Logik einzubauen, entweder im CL, oder als eigenes RPG-Programm, welches datenbankgestützt Benutzerberechtigungen prüft.
- generell gilt für alle Exit-Programme: Sie wirken erst, wenn der zugehörige Job neu gestartet wird. Für bereits aktive Serverjobs oder Sitzungen wird das Eintragen eines Exitprogramms nicht wirksam! Beenden Sie also entsprechende Serverdienste, siehe Subsysteme QSERVER, QSYSWRK, QUSRWRK.



(Sc)hauen Sie Ihren Anwendern auf die Finger!

•Mit SBMJOB kann ein böser User rekursive Schweinereien anstellen:
Beispielprogramm LOOP:

```
PGM
  GO:     SNDMSG MSG('Hallo!') TOUSR(QSYSOPR)
          SBMJOB CMD(CALL LOOP)
          GOTO CMDLBL(GO)
ENDPGM
```

Ergebnis: Eine schöne Endlosschleife, wenn die JOBQ des Benutzers nicht beschränkt ist, steht die Maschine nach wenigen Minuten, und die Platten sterben einen Heldentod.

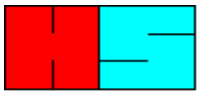
Abhilfe: Jedem Benutzer eine eigene JOBQ zuteilen. Diese JOBQ sollte begrenzt einem Subsystem zugeordnet sein. Bei sehr vielen Usern eine Jobbeschreibung für Usergruppen verwenden. Wichtigstes Ziel: Es sollten nicht viele Batchjobs gleichzeitig möglich sein! Idealerweise pro User max. 1 gleichzeitig. Somit kann auch eine JOBQ nicht in einem Loop gefüllt werden (s.o.)
Kurzbeispiel:

```
CRTUSRPRF  DUMMY
CRTJOBQ QGPL/DUMMY
CRTJOBD JOBD(QGPL/DUMMY) JOBQ(QGPL/DUMMY)
ADDJOBQE SBSD(QBATCH) JOBQ(QGPL/DUMMY) SEQNBR(9999)
CHGUSRPRF USRPRF(DUMMY) JOBD(QGPL/DUMMY)
```

Nun kann Benutzer DUMMY das folgende Programm TEST in QGPL gefahrlos starten:

```
PGM
         SNDMSG MSG(HALLO) TOUSR(QSYSOPR)
         SBMJOB CMD(CALL PGM(QGPL/TEST))
ENDPGM
```

Selbstverständlich sollte DUMMY keine Berechtigungen für andere Subsyteme oder JobQueues oder Jobbeschreibungen haben...



Denial of Service oder: Ausgesperrt!

•Mit ALCOBJ kann man ganz schnell das System schliessen, um seine Ruhe zu haben:

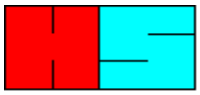
```
ALCOBJ OBJ((QCMD *PGM *EXCL))
```

Sperrt den zentrallen Kommandointerpreter QCMD, keine interaktive Anmeldung mehr möglich!

Abhilfe: ein CL, welches ständig prüft, ob auf QCMD *PGM eine exklusive Sperre existiert. Wenn ja, entsprechenden Job ermitteln und beenden. Sollte endlos in einer Schleife laufen:

```
0001.00 PGM
0002.00               DCLF       FILE(QTEMP/QPDSPOLK)
0003.00               DCL        VAR(&JOBNAM) TYPE(*CHAR) LEN(10)
0004.00               DCL        VAR(&JOBUSR) TYPE(*CHAR) LEN(10)
0005.00               DCL        VAR(&JOBNUM) TYPE(*CHAR) LEN(6)
0006.00               DCL        VAR(&JOBLCK) TYPE(*CHAR) LEN(6)
0008.00               DLTF       FILE(QTEMP/QPDSPOLK)
0009.00               MONMSG     MSGID(CPF0000)
0011.00               CRTPF      FILE(QTEMP/QPDSPOLK) RCDLEN(132)
0012.00               MONMSG     MSGID(CPF0000)
0014.00               WRKOBJLCK  OBJ(QSYS/QCMD) OBJTYPE(*PGM) OUTPUT(*PRINT)
0015.00               CPYSPLF    FILE(QPDSPOLK) TOFILE(QTEMP/QPDSPOLK) +
0016.00                            SPLNBR(*LAST)
0018.00  LOOP:       RCVF
0019.00               MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(ENDE))
0020.00               CHGVAR     VAR(&JOBNAM) VALUE(%SST(&QPDSPOLK 4 10))
0021.00               CHGVAR     VAR(&JOBUSR) VALUE(%SST(&QPDSPOLK 16 10))
0022.00               CHGVAR     VAR(&JOBNUM) VALUE(%SST(&QPDSPOLK 28 6))
0023.00               CHGVAR     VAR(&JOBLCK) VALUE(%SST(&QPDSPOLK 37 6))
0024.00               IF         COND(&JOBLCK *EQ '*EXCL') THEN(DO)
0025.00               SNDMSG     MSG(&JOBLCK) TOUSR(*REQUESTER)
0026.00               ENDJOB     JOB(&JOBNUM/&JOBUSR/&JOBNAM) OPTION(*IMMED) +
0027.00                            LOGLMT(*NOMAX) ADLINTJOBS(*ALL)
0028.00               ENDDO
0029.00               GOTO       CMDLBL(LOOP)
0031.00  ENDE:       ENDPGM
```

Abhilfe2: Spezielles Subsystem, das eine Kopie von QCMD verwendet und für einen speziellen Gerätenamen eingerichtet ist (nicht 100%ig sauber, aber wirksam!). Hier müssen die Leitwegeinträge eines Subsystems geändert werden, so dass diese auf die Kopie von QCMD zeigen. Diese Kopie sollte anders heissen.



•Benutzer geben sich und ihrem Job hohe Priorität
Hier hilft ein (etwas komplexes und langsames) CL-Script, welches a) die aktiven Jobs ermittelt, anhand bestimmter Regeln für jeden Job die Priorität neu setzt und c) noch andere Überwachungstätigkeiten wie Anzahl Jobs pro Userprofil überwachen kann.
Diese Methode per CL ist nicht ganz elegant und braucht etwas CPU-Zeit, aber einfach zu erstellen. Man schlägt hier zwar dem automatischen Prioritätenmanager von IBM ein Schnippchen, aber bei Maschinen, die von einigen Powerusern (Query?) dichtgemacht werden, kann dies helfen.
Die Methode hilft aber nicht, wenn ein CFINT zugeschlagen hat.

Ein Beispiel habe ich hier jetzt nicht wiedergegeben (mehrere 100 Zeilen), kann aber gerne als Denkanstoss bei mir per eMail an **`info@holgerscherer.de`** angefordert werden.