```
**********************************************************
   12641 Tue Jan 20 23:18:28 2015
new/usr/src/uts/common/fs/nfs/nfs_acl_xdr.c
5548 Attempt to read ACLs from Illumos NFS client is toxic
**********************************************************
______unchanged_portion_omitted_

 86 /*
 87  * Serialize and de-serialize access control attributes
 88  */
 89 bool_t
 90 xdr_secattr(XDR *xdrs, vsecattr_t *objp)
 91 {
 92         uint_t count = 0;
 93         uint_t dfacount = 0;
 89         uint_t count;

 95         if (!xdr_u_int(xdrs, &objp->vsa_mask)) {
 91         if (!xdr_u_int(xdrs, &objp->vsa_mask))
 96                 return (FALSE);
 97         }

 99         /*
100          * Refuse request if we do not understand it completely.
101          * There should be at least one valid bit set in the mask and
102          * none of the unknown bits set.
103          */
104         if ((objp->vsa_mask &
105                 (VSA_ACL | VSA_ACLCNT | VSA_DFACL | VSA_DFACLCNT)) == 0) {
 93         if (!xdr_int(xdrs, &objp->vsa_aclcnt))
106             return (FALSE);
107         }
108         if ((objp->vsa_mask &
109                 ~(VSA_ACL | VSA_ACLCNT | VSA_DFACL | VSA_DFACLCNT)) != 0) {
110             return (FALSE);
111         }

113         if (!xdr_int(xdrs, &objp->vsa_aclcnt)) {
114                 return (FALSE);
115         }
116         if (objp->vsa_aclentp != NULL) {
 95         if (objp->vsa_aclentp != NULL)
117                 count = (uint_t)objp->vsa_aclcnt;
118         }

 97         else
 98                 count = 0;
120         if (!xdr_array(xdrs, (char **)&objp->vsa_aclentp, &count,
121             NFS_ACL_MAX_ENTRIES, sizeof (aclent_t), (xdrproc_t)xdr_aclent)) {
100         NFS_ACL_MAX_ENTRIES, sizeof (aclent_t), (xdrproc_t)xdr_aclent))
122                 return (FALSE);
123         }

125         if (count != 0 && count != (uint_t)objp->vsa_aclcnt) {
126                 /*
127                  * Assign the actual array size to vsa_aclcnt before
128                  * aborting on error
129                  */
130                 objp->vsa_aclcnt = (int)count;
131                 return (FALSE);
132         }

134         /*
135          * For VSA_ACL the count should be zero or there should
136          * be array attached.
137          */
```



```
138         if ((objp->vsa_mask & VSA_ACL) != 0) {
139                 if ((objp->vsa_aclcnt != 0) && (objp->vsa_aclentp == NULL)) {
140                         objp->vsa_aclcnt = 0;
110         if (!xdr_int(xdrs, &objp->vsa_dfaclcnt))
141                         return (FALSE);
142                 }
143         }

145         if (!xdr_int(xdrs, &objp->vsa_dfaclcnt)) {
112         if (objp->vsa_dfaclentp != NULL)
113                 count = (uint_t)objp->vsa_dfaclcnt;
114         else
115                 count = 0;
116         if (!xdr_array(xdrs, (char **)&objp->vsa_dfaclentp, &count,
117             NFS_ACL_MAX_ENTRIES, sizeof (aclent_t), (xdrproc_t)xdr_aclent))
146                 return (FALSE);
147         }
148         if (objp->vsa_dfaclentp != NULL) {
149                 dfacount = (uint_t)objp->vsa_dfaclcnt;
150         }

152         if (!xdr_array(xdrs, (char **)&objp->vsa_dfaclentp, &dfacount,
153             NFS_ACL_MAX_ENTRIES, sizeof (aclent_t), (xdrproc_t)xdr_aclent)) {
154                 return (FALSE);
155         }

157         if (dfacount != 0 && dfacount != (uint_t)objp->vsa_dfaclcnt) {
119         if (count != 0 && count != (uint_t)objp->vsa_dfaclcnt) {
158                 /*
159                  * Assign the actual array size to vsa_dfaclcnt before
160                  * aborting on error
161                  */
162                 objp->vsa_dfaclcnt = (int)dfacount;
124                 objp->vsa_dfaclcnt = (int)count;
163                 return (FALSE);
164         }

166         /*
167          * for VSA_DFACL The count should be zero or there should
168          * be array attached
169          */
170         if ((objp->vsa_mask & VSA_DFACL) != 0) {
171                 if ((objp->vsa_dfaclcnt != 0) &&
172                     (objp->vsa_dfaclentp == NULL)) {
173                         objp->vsa_dfaclcnt = 0;
174                         return (FALSE);
175                 }
176         }


179         return (TRUE);
180 }
______unchanged_portion_omitted_
```