
Subject: [PATCH v2 3/5] SUNRPC: create GSS auth cache per network namespace
Posted by Stanislav Kinsbursky on Thu, 19 Jan 2012 17:42:37 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch makes GSS auth cache details allocated and registered per network namespace context.

Thus with this patch rsi_cache and rsc_cache contents for network namespace "X" are controlled from proc file system mount for the same network namespace "X".

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
---  
include/linux/sunrpc/svcauth_gss.h |  2  
net/sunrpc/auth_gss/auth_gss.c   | 21 +++++  
net/sunrpc/auth_gss/svcauth_gss.c | 165 ++++++-----  
net/sunrpc/cache.c              |  2  
net/sunrpc/netns.h              |  2  
net/sunrpc/sunrpc_syms.c        |  1  
6 files changed, 143 insertions(+), 50 deletions(-)
```

```
diff --git a/include/linux/sunrpc/svcauth_gss.h b/include/linux/sunrpc/svcauth_gss.h  
index 83bbee3..7c32daa 100644  
--- a/include/linux/sunrpc/svcauth_gss.h  
+++ b/include/linux/sunrpc/svcauth_gss.h  
@@ -18,6 +18,8 @@
```

```
int gss_svc_init(void);  
void gss_svc_shutdown(void);  
+int gss_svc_init_net(struct net *net);  
+void gss_svc_shutdown_net(struct net *net);  
int svcauth_gss_register_pseudoflavor(u32 pseudoflavor, char * name);  
u32 svcauth_gss_flavor(struct auth_domain *dom);  
char *svc_gss_principal(struct svc_rqst *);  
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c  
index 9ef7fa3..5e02207 100644  
--- a/net/sunrpc/auth_gss/auth_gss.c  
+++ b/net/sunrpc/auth_gss/auth_gss.c  
@@ -1662,6 +1662,21 @@ static const struct rpc_pipe_ops gss_upcall_ops_v1 = {  
.release_pipe = gss_pipe_release,  
};  
  
+static __net_init int rpcsec_gss_init_net(struct net *net)  
+{  
+ return gss_svc_init_net(net);  
+}  
+  
+static __net_exit void rpcsec_gss_exit_net(struct net *net)  
+{
```

```

+ gss_svc_shutdown(net);
+}
+
+static struct pernet_operations rpcsec_gss_net_ops = {
+ .init = rpcsec_gss_init_net,
+ .exit = rpcsec_gss_exit_net,
+};
+
/*
 * Initialize RPCSEC_GSS module
 */
@@ -1675,8 +1690,13 @@ static int __init init_rpcsec_gss(void)
err = gss_svc_init();
if (err)
goto out_unregister;
+ err = register_pernet_subsys(&rpcsec_gss_net_ops);
+ if (err)
+ goto out_svc_exit;
rpc_init_wait_queue(&pipe_version_rpc_waitqueue, "gss pipe version");
return 0;
+out_svc_exit:
+ gss_svc_shutdown();
out_unregister:
rpcauth_unregister(&authgss_ops);
out:
@@ -1685,6 +1705,7 @@ out:

static void __exit exit_rpcsec_gss(void)
{
+ unregister_pernet_subsys(&rpcsec_gss_net_ops);
gss_svc_shutdown();
rpcauth_unregister(&authgss_ops);
rcu_barrier(); /* Wait for completion of call_rcu()'s */
diff --git a/net/sunrpc/auth_gss/svcauth_gss.c b/net/sunrpc/auth_gss/svcauth_gss.c
index 8d0f7d3..1600cfb 100644
--- a/net/sunrpc/auth_gss/svcauth_gss.c
+++ b/net/sunrpc/auth_gss/svcauth_gss.c
@@ -48,6 +48,8 @@ 
#include <linux/sunrpc/svcauth_gss.h>
#include <linux/sunrpc/cache.h>

+#include "../netns.h"
+
#ifndef RPC_DEBUG
#define RPCDBG_FACILITY RPCDBG_AUTH
#endif
@@ -75,10 +77,8 @@ struct rsi {
    int major_status, minor_status;

```

```

};

-static struct cache_head *rsi_table[RSI_HASHMAX];
-static struct cache_detail rsi_cache;
-static struct rsi *rsi_update(struct rsi *new, struct rsi *old);
-static struct rsi *rsi_lookup(struct rsi *item);
+static struct rsi *rsi_update(struct cache_detail *cd, struct rsi *new, struct rsi *old);
+static struct rsi *rsi_lookup(struct cache_detail *cd, struct rsi *item);

static void rsi_free(struct rsi *rsii)
{
@@ -216,7 +216,7 @@ static int rsi_parse(struct cache_detail *cd,
if (dup_to_netobj(&rsii.in_token, buf, len))
goto out;

- rsip = rsi_lookup(&rsii);
+ rsip = rsi_lookup(cd, &rsii);
if (!rsip)
goto out;

@@ -258,21 +258,20 @@ static int rsi_parse(struct cache_detail *cd,
if (dup_to_netobj(&rsii.out_token, buf, len))
goto out;
rsii.h.expiry_time = expiry;
- rsip = rsi_update(&rsii, rsip);
+ rsip = rsi_update(cd, &rsii, rsip);
status = 0;
out:
rsi_free(&rsii);
if (rsip)
- cache_put(&rsip->h, &rsi_cache);
+ cache_put(&rsip->h, cd);
else
status = -ENOMEM;
return status;
}

-static struct cache_detail rsi_cache = {
+static struct cache_detail rsi_cache_template = {
.owner = THIS_MODULE,
.hash_size = RSI_HASHMAX,
- .hash_table = rsi_table,
.name = "auth.rpcsec.init",
.cache_put = rsi_put,
.cache_upcall = rsi_upcall,
@@ -283,24 +282,24 @@ static struct cache_detail rsi_cache = {
.alloc = rsi_alloc,
};

```

```

-static struct rsi *rsi_lookup(struct rsi *item)
+static struct rsi *rsi_lookup(struct cache_detail *cd, struct rsi *item)
{
    struct cache_head *ch;
    int hash = rsi_hash(item);

- ch = sunrpc_cache_lookup(&rsi_cache, &item->h, hash);
+ ch = sunrpc_cache_lookup(cd, &item->h, hash);
if (ch)
    return container_of(ch, struct rsi, h);
else
    return NULL;
}

-static struct rsi *rsi_update(struct rsi *new, struct rsi *old)
+static struct rsi *rsi_update(struct cache_detail *cd, struct rsi *new, struct rsi *old)
{
    struct cache_head *ch;
    int hash = rsi_hash(new);

- ch = sunrpc_cache_update(&rsi_cache, &new->h,
+ ch = sunrpc_cache_update(cd, &new->h,
    &old->h, hash);
if (ch)
    return container_of(ch, struct rsi, h);
@@ -339,10 +338,8 @@ struct rsc {
    char *client_name;
};

-static struct cache_head *rsc_table[RSC_HASHMAX];
-static struct cache_detail rsc_cache;
-static struct rsc *rsc_update(struct rsc *new, struct rsc *old);
-static struct rsc *rsc_lookup(struct rsc *item);
+static struct rsc *rsc_update(struct cache_detail *cd, struct rsc *new, struct rsc *old);
+static struct rsc *rsc_lookup(struct cache_detail *cd, struct rsc *item);

static void rsc_free(struct rsc *rsci)
{
@@ -444,7 +441,7 @@ static int rsc_parse(struct cache_detail *cd,
if (expiry == 0)
    goto out;

- rscp = rsc_lookup(&rsci);
+ rscp = rsc_lookup(cd, &rsci);
if (!rscp)
    goto out;

```

```

@@ -506,22 +503,21 @@ static int rsc_parse(struct cache_detail *cd,
}

rsci.h.expiry_time = expiry;
- rscp = rsc_update(&rsci, rscp);
+ rscp = rsc_update(cd, &rsci, rscp);
status = 0;
out:
    gss_mech_put(gm);
    rsc_free(&rsci);
    if (rscp)
- cache_put(&rscp->h, &rsc_cache);
+ cache_put(&rscp->h, cd);
    else
        status = -ENOMEM;
    return status;
}

-static struct cache_detail rsc_cache = {
+static struct cache_detail rsc_cache_template = {
    .owner = THIS_MODULE,
    .hash_size = RSC_HASHMAX,
- .hash_table = rsc_table,
    .name = "auth.rpcsec.context",
    .cache_put = rsc_put,
    .cache_parse = rsc_parse,
@@ -531,24 +527,24 @@ static struct cache_detail rsc_cache = {
    .alloc = rsc_alloc,
};

-static struct rsc *rsc_lookup(struct rsc *item)
+static struct rsc *rsc_lookup(struct cache_detail *cd, struct rsc *item)
{
    struct cache_head *ch;
    int hash = rsc_hash(item);

- ch = sunrpc_cache_lookup(&rsc_cache, &item->h, hash);
+ ch = sunrpc_cache_lookup(cd, &item->h, hash);
    if (ch)
        return container_of(ch, struct rsc, h);
    else
        return NULL;
}

-static struct rsc *rsc_update(struct rsc *new, struct rsc *old)
+static struct rsc *rsc_update(struct cache_detail *cd, struct rsc *new, struct rsc *old)
{
    struct cache_head *ch;

```

```

int hash = rsc_hash(new);

- ch = sunrpc_cache_update(&rsc_cache, &new->h,
+ ch = sunrpc_cache_update(cd, &new->h,
    &old->h, hash);
if (ch)
    return container_of(ch, struct rsc, h);
@@ -558,7 +554,7 @@ static struct rsc *rsc_update(struct rsc *new, struct rsc *old)

static struct rsc *
-gss_svc_searchbyctx(struct xdr_netobj *handle)
+gss_svc_searchbyctx(struct cache_detail *cd, struct xdr_netobj *handle)
{
    struct rsc rsci;
    struct rsc *found;
@@ -566,11 +562,11 @@ gss_svc_searchbyctx(struct xdr_netobj *handle)
    memset(&rsci, 0, sizeof(rsci));
    if (dup_to_netobj(&rsci.handle, handle->data, handle->len))
        return NULL;
- found = rsc_lookup(&rsci);
+ found = rsc_lookup(cd, &rsci);
    rsc_free(&rsci);
    if (!found)
        return NULL;
- if (cache_check(&rsc_cache, &found->h, NULL))
+ if (cache_check(cd, &found->h, NULL))
    return NULL;
    return found;
}
@@ -968,20 +964,20 @@ svcauth_gss_set_client(struct svc_rqst *rqstp)
}

static inline int
-gss_write_init_verf(struct svc_rqst *rqstp, struct rsi *rsip)
+gss_write_init_verf(struct cache_detail *cd, struct svc_rqst *rqstp, struct rsi *rsip)
{
    struct rsc *rsci;
    int      rc;

    if (rsip->major_status != GSS_S_COMPLETE)
        return gss_write_null_verf(rqstp);
- rsci = gss_svc_searchbyctx(&rsip->out_handle);
+ rsci = gss_svc_searchbyctx(cd, &rsip->out_handle);
    if (rsci == NULL) {
        rsip->major_status = GSS_S_NO_CONTEXT;
        return gss_write_null_verf(rqstp);
    }
}

```

```

rc = gss_write_verf(rqstp, rsci->mechctx, GSS_SEQ_WIN);
- cache_put(&rsci->h, &rsc_cache);
+ cache_put(&rsci->h, cd);
    return rc;
}

@@ -1000,6 +996,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
    struct xdr_netobj tmpobj;
    struct rsi *rsip, rsikey;
    int ret;
+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

/* Read the verifier; should be NULL: */
*authp = rpc_autherr_badverf;
@@ -1028,17 +1025,17 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
}

/* Perform upcall, or find upcall result: */
- rsip = rsi_lookup(&rsikey);
+ rsip = rsi_lookup(sn->rsi_cache, &rsikey);
    rsi_free(&rsikey);
    if (!rsip)
        return SVC_CLOSE;
- if (cache_check(&rsi_cache, &rsip->h, &rqstp->rq_chandle) < 0)
+ if (cache_check(sn->rsi_cache, &rsip->h, &rqstp->rq_chandle) < 0)
    /* No upcall result: */
    return SVC_CLOSE;

ret = SVC_CLOSE;
/* Got an answer to the upcall; use it: */
- if (gss_write_init_verf(rqstp, rsip))
+ if (gss_write_init_verf(sn->rsc_cache, rqstp, rsip))
    goto out;
    if (resv->iov_len + 4 > PAGE_SIZE)
        goto out;
@@ -1055,7 +1052,7 @@ static int svcauth_gss_handle_init(struct svc_rqst *rqstp,
    ret = SVC_COMPLETE;
out:
- cache_put(&rsip->h, &rsi_cache);
+ cache_put(&rsip->h, sn->rsi_cache);
    return ret;
}

@@ -1079,6 +1076,7 @@ static int svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
    __be32 *rpcstart;
    __be32 *reject_stat = resv->iov_base + resv->iov_len;
    int ret;

```

```

+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

dprintk("RPC:    svcauth_gss: argv->iov_len = %zd\n",
       argv->iov_len);
@@ -1129,7 +1127,7 @@ svcauth_gss_accept(struct svc_rqst *rqstp, __be32 *authp)
case RPC_GSS_PROC_DESTROY:
/* Look up the context, and check the verifier: */
*authp = rpcsec_gsserr_credproblem;
- rsci = gss_svc_searchbyctx(&gc->gc_ctx);
+ rsci = gss_svc_searchbyctx(sn->rsc_cache, &gc->gc_ctx);
if (!rsci)
    goto auth_err;
switch (gss_verify_header(rqstp, rsci, rpcstart, gc, authp)) {
@@ -1209,7 +1207,7 @@ drop:
ret = SVC_DROP;
out:
if (rsci)
- cache_put(&rsci->h, &rsc_cache);
+ cache_put(&rsci->h, sn->rsc_cache);
return ret;
}

@@ -1362,6 +1360,7 @@ svcauth_gss_release(struct svc_rqst *rqstp)
struct rpc_gss_wire_cred *gc = &gsd->clcred;
struct xdr_buf *resbuf = &rqstp->rq_res;
int stat = -EINVAL;
+ struct sunrpc_net *sn = net_generic(rqstp->rq_xprt->xpt_net, sunrpc_net_id);

if (gc->gc_proc != RPC_GSS_PROC_DATA)
    goto out;
@@ -1404,7 +1403,7 @@ out_err:
put_group_info(rqstp->rq_cred.cr_group_info);
rqstp->rq_cred.cr_group_info = NULL;
if (gsd->rsci)
- cache_put(&gsd->rsci->h, &rsc_cache);
+ cache_put(&gsd->rsci->h, sn->rsc_cache);
gsd->rsci = NULL;

return stat;
@@ -1429,30 +1428,96 @@ static struct auth_ops svcauthops_gss = {
.set_client = svcauth_gss_set_client,
};

+static int rsi_cache_create_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;

```

```

+
+ cd = cache_create_net(&rsi_cache_template, net);
+ if (IS_ERR(cd))
+ return PTR_ERR(cd);
+ err = cache_register_net(cd, net);
+ if (err) {
+ cache_destroy_net(cd, net);
+ return err;
+ }
+ sn->rsi_cache = cd;
+ return 0;
+}
+
+static void rsi_cache_destroy_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd = sn->rsi_cache;
+
+ sn->rsi_cache = NULL;
+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
+}
+
+static int rsc_cache_create_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd;
+ int err;
+
+ cd = cache_create_net(&rsc_cache_template, net);
+ if (IS_ERR(cd))
+ return PTR_ERR(cd);
+ err = cache_register_net(cd, net);
+ if (err) {
+ cache_destroy_net(cd, net);
+ return err;
+ }
+ sn->rsc_cache = cd;
+ return 0;
+}
+
+static void rsc_cache_destroy_net(struct net *net)
+{
+ struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
+ struct cache_detail *cd = sn->rsc_cache;
+
+ sn->rsc_cache = NULL;

```

```

+ cache_purge(cd);
+ cache_unregister_net(cd, net);
+ cache_destroy_net(cd, net);
+}
+
int
-gss_svc_init(void)
+gss_svc_init(struct net *net)
{
- int rv = svc_auth_register(RPC_AUTH_GSS, &svcauthops_gss);
+ int rv;
+
+ rv = rsc_cache_create_net(net);
if (rv)
    return rv;
- rv = cache_register(&rsc_cache);
+ rv = rsi_cache_create_net(net);
if (rv)
    goto out1;
- rv = cache_register(&rsi_cache);
- if (rv)
-    goto out2;
    return 0;
-out2:
- cache_unregister(&rsc_cache);
out1:
- svc_auth_unregister(RPC_AUTH_GSS);
+ rsc_cache_destroy_net(net);
    return rv;
}

void
+gss_svc_shutdown_net(struct net *net)
+{
+ rsi_cache_destroy_net(net);
+ rsc_cache_destroy_net(net);
+}
+
+int
+gss_svc_init(void)
+{
+ return svc_auth_register(RPC_AUTH_GSS, &svcauthops_gss);
+}
+
+void
gss_svc_shutdown(void)
{
- cache_unregister(&rsc_cache);

```

```

- cache_unregister(&rsi_cache);
  svc_auth_unregister(RPC_AUTH_GSS);
}
diff --git a/net/sunrpc/cache.c b/net/sunrpc/cache.c
index 9ef5926..a450b8a 100644
--- a/net/sunrpc/cache.c
+++ b/net/sunrpc/cache.c
@@ -1643,6 +1643,7 @@ int cache_register_net(struct cache_detail *cd, struct net *net)
    sunrpc_destroy_cache_detail(cd);
    return ret;
}
+EXPORT_SYMBOL_GPL(cache_register_net);

int cache_register(struct cache_detail *cd)
{
@@ -1655,6 +1656,7 @@ void cache_unregister_net(struct cache_detail *cd, struct net *net)
    remove_cache_proc_entries(cd, net);
    sunrpc_destroy_cache_detail(cd);
}
+EXPORT_SYMBOL_GPL(cache_unregister_net);

void cache_unregister(struct cache_detail *cd)
{
diff --git a/net/sunrpc/netns.h b/net/sunrpc/netns.h
index 309f88d..ce7bd44 100644
--- a/net/sunrpc/netns.h
+++ b/net/sunrpc/netns.h
@@ -10,6 +10,8 @@ struct sunrpc_net {
    struct proc_dir_entry *proc_net_rpc;
    struct cache_detail *ip_map_cache;
    struct cache_detail *unix_gid_cache;
+   struct cache_detail *rsc_cache;
+   struct cache_detail *rsi_cache;

    struct super_block *pipefs_sb;
    struct mutex pipefs_sb_lock;
diff --git a/net/sunrpc/sunrpc_syms.c b/net/sunrpc/sunrpc_syms.c
index 38a72a1..d16ac08 100644
--- a/net/sunrpc/sunrpc_syms.c
+++ b/net/sunrpc/sunrpc_syms.c
@@ -25,6 +25,7 @@ 
 #include "netns.h"

int sunrpc_net_id;
+EXPORT_SYMBOL_GPL(sunrpc_net_id);

extern int unix_gid_cache_create(struct net *net);
extern int unix_gid_cache_destroy(struct net *net);

```
