Open Source Kerberos Tools: KNC and Kharon

Roland C. Dowdeswell <elric@imrryr.org>

KNC: Kerberised NetCat

- Like netcat.
- But with Kerberos.
- GSSAPI, really.
- Full privacy and integrity.
- Why not ssh/rsh?

KNC: How do you use it?

NAME

knc -- kerberized netcat

SYNOPSIS

knc -l [-n] [-d] [-a bind_address] [-f] [-c num] port prog [args]
knc -il [-n] [-d] prog [args]
knc -lS path [-n] [-d] [-a bind_address] [-f] [-c num] port
knc -ilS path [-n] [-d]

knc [-d] [-n] service@host port
knc [-d] [-n] -N fd service@host

KNC Example: Client setup

Step 1

ssh-proxy-suid

Step 2

```
#!/usr/pkg/bin/perl

my ($host) = @ARGV;

$ENV{KRB5CCNAME} = "FILE:/tmp/foo.$$";
system("kinit -k host/\$(hostname)");

exec("knc -- host\@$host 2666");
```

Step 3

```
ssh -v -oProxyCommand="/tmp/ssh-proxy-suid %h"  aceroalx
```

KNC Example: SSHD setup

Step 1

```
#!/usr/pkg/bin/perl

exit 1 if $ENV{KNC_CREDS} ne 'host/mournblade@IMRRYR.ORG';

exec {“/usr/sbin/sshd”} (qw{sshd -f /etc/ssh/sshd_config -i});
```

Step 2

```
knc -l 2666 -- /tmp/sshd
```

KNC Example: SSH further ideas

- Integrate into the PAM stack evaluating KNC_CREDS
- Require Kerberos and another form of authentication

KNC: Other use cases

- SSH ``Jump Servers''
- Kharon
- SVN
- RSYNC
- SOAP and other inefficient whatnot

Kharon: What is it?

- Basically OO RPC
- Marshalls and unmarshalls a variety of Perl data structures
- Client side failover
- Server side redirection
- Logging framework
- Entitlements

Kharon: A simple example 1

```
package Example::Eg1;

use strict;
use warnings;

sub new {
        my $self;
        $self->{num} = 1;
        bless($self);
}
sub inc {
        my ($self) = @_;
        $self->{num}++;
        undef;
}
sub query {
        my ($self) = @_;
        return $self->{num};
}

sub exception {
        die "alkjsnckjsanclkjac";
}

1;
```

Kharon: A simple example 2

```
#!/usr/pkg/bin/perl

use Example::Eg1;
use strict;
use warnings;

my $obj = Example::Eg1->new();
my $ret = $obj->query();
print "$ret\n";

$obj->inc();
$obj->inc();
$obj->inc();
$ret = $obj->query();
print "$ret\n";

$obj->complicated();
$obj->exception();

This program will output:
1
4
alkjsnckjsanclkjac at Example/Eg1.pm line 34.
```

Kharon: A simple example 3

```
#!/usr/pkg/bin/perl

use Example::Eg1;
use Kharon::Protocol::ArrayHash;
use Kharon::ProtocolEngineServer;

#
# Instantiate our object:
my $obj = Example::Eg1->new();

#
# Setup Kharon:
my $ahr = Kharon::Protocol::ArrayHash->new(banner => {version =>'2.0' } );
my $pes = Kharon::ProtocolEngineServer->new($ahr);
$pes->Connect();

#
# And now ``Run the Object'', exporting the methods inc, query, exception
# and complicated:

$pes->RunObj(object => $obj, cmds => [ qw/inc query exception complicated/ ]);
```

Kharon: A simple example 4

```
$ ./daemon
220 . {version=2.0}
query
250 . 1
inc
250 . !
inc
250 . !
inc
250 . !
query
250 . 4
complicated
250 - foo
250 . [bar,baz,{wow=!,c=d,a=b}]
quit
220 . bye
```

Kharon: A simple example 5

```
package Example::Client;

use Kharon::Protocol::ArrayHash;
use Kharon::ProtocolEngineClientKnc;
use Kharon::utils qw/mk_array_methods mk_scalar_methods/;
use Example::Eg1;

sub new {
        my ($isa, @servers) = @_;
        my $self;
        my $ahr = Kharon::Protocol::ArrayHash->new(banner =>
            {version=>'2.0'});
        my $pec = Kharon::ProtocolEngineClientKnc->new($ahr);
        $pec->SetServerDefaults({KncService=>'service1',
             PeerPort=>2666});

        $pec->Connect(@servers);
        $self->{pec} = $pec;
        bless($self, $isa);
}

eval mk_scalar_methods('Example::Eg1', qw/inc query complicated exception/);

1;
```

Kharon: A simple example 6

So, we can rewrite our original program just adding:

```
use Example::Client;
```

And then modifying the instantiation of $obj from using Example::Eg1 to Example::Client:

```
my $obj = Example::Client->new('dowdeslx');
```

And now we have a Kerberised, mutually authenticated, encrypted, integral client/server application.

Kharon: Futures

- C implementation
- Other languages
- Other protocols
- LD_PRELOAD
- Better entitlement model

Applications

- krb5_admin
- krb5_keytab
- Proid
- Sudo configuration management
- AFS key management

Where is all of this?

http://oskt.secure-endpoints.com/

Disclaimer: not everything is up there yet...