Final project report for "Perform security review of LCFG profile access mechanism" (555)

Project description

Project 402 "Review Security of LCFG Profile Access - transport mechanism" delivered a more secure delivery mechanism for LCFG profiles. This project will perform a security review of the new mechanism to give us additional confidence that the original project met its goals.

Original project pages

- Computing projects page
- Home page
- Final report

More details on project work

From Stephen:

> *There are two parts to the "Perform security review of LCFG profile access mechanism" project.*
>
> *The first part is to verify that the apache config correctly restricts access to the LCFG profiles so that they are only accessible to the relevant lcfg client, users with the /root or /admin principal or in some cases the nagios server. I guess that comes down to reviewing the apache config files and verifying the directory/file permissions on the server offer sufficient protection.*
>
> *The second part is to to review the code in the LCFG::Client::ProfileFetcher module and the helper modules in the LCFG::Client::Fetch name space. It needs checking for programming errors, incorrect use of modules like LWP (i.e. does it do the https/certificate stuff properly?). Also anything that's unclear or badly documented needs flagging up so we can improve maintainability. All the code is in LCFG-Client-Perl in the lcfg source subversion repository.*

Work done

The work undertaken was along the lines as suggested by Stephen, as quoted above.

After initially looking at the client code, I decided to spend some time refamiliarising myself with the workings of the LCFG client. This wasn't strictly necessary for the purposes of this project, but helps to provide context to the code being examined.

Apache configuration

The apache configuration for profile access is contained in the file `/etc/httpd/conf.d/lcfgssl.conf` on the LCFG servers. The key part of configuration is this:

```
AccessFileName .sslaccess
...
# The profiles directory is the *only* place where user-controlled
# override files (.sslaccess in this case) are permitted.

<Directory "${lcfgsslroot}/profiles">
   Options +Indexes
   AllowOverride AuthConfig Limit
</Directory>
```

For each LCFG host profile, there is a `.sslaccess` file which configures access, e.g. for my machine, bolt:

```
[vega]root: cat /var/lcfg/conf/server/web/profiles/inf.ed.ac.uk/bolt/.sslaccess

AuthType GSSAPI
AuthName "lcfg@bolt.inf.ed.ac.uk"
GssapiBasicAuth Off
GssapiBasicAuthMech krb5
GssapiSSLonly On

GssapiCredStore "keytab:/etc/httpd.keytab"


AuthGroupFile "/etc/httpd/lcfg.groups.d/general"

<RequireAny>
  Require user lcfg/bolt.inf.ed.ac.uk@INF.ED.AC.UK
  Require group lcfgadmins
</RequireAny>

# eof
[vega]root:
```

This file is built every time a profile is processed. This is done by the MakeAccessFiles subroutine in `mkxprof`, according to profile resources, e.g. from the output of qxprof profile:

```
auth=http ssl
auth_tmpl_ssl=apache_gssapi.tt
auth_val_ssl_groupfile=/etc/httpd/lcfg.groups.d/general
auth_vars_ssl=groupfile
```

The template file (`apache_gssapi.tt`) is shipped by the LCFG-Compiler package.

The `/etc/httpd/lcfg.groups.d/general` file currently defines two access groups: (1) lcfgadmins (all C[S]O '/admin' and '/root' principals) and (2) nagios (host-based principals for nagios servers). These groups are built by the apacheconf component from resources currently defined in `<live/lcfg-slave-server.h>`

All of this defines that, for an inf.ed.ac.uk host profile, access is limited to:

- SSL/TLS connections only
- the 'lcfg/<host>.inf.ed.ac.uk' principal of the machine
- sysadmin '/root' and '/admin' principals
- nagios machine principals (if configured for nagios)

The access control is managed via the configuration describe above, using the mod_auth_gssapi apache module.

Access to other (network device) profiles

Note that access to profiles that are not under the 'inf.ed.ac.uk' directory currently allow access for anyone within the 'inf.ed.ac.uk' domain. It is assumed that this is a deliberate policy decision. These are 'at.net.inf.ed.ac.uk', 'f.net.inf.ed.ac.uk', 'kb.net.inf.ed.ac.uk', 'bayes.net.inf.ed.ac.uk', 'net.inf.ed.ac.uk'). It appears that most, perhaps all, of these profiles are for network switches.

Testing access control

I confirmed that I could only access a machine's profile using the principals for which access was intended. Attempts to download a machine's profile using other principals correctly generated a 401 Unauthorized http status code.

Testing was done using both curl, e.g.

```
curl --negotiate -u : https://lcfg1.inf.ed.ac.uk/profiles/inf.ed.ac.uk/bolt/XML/profile.xml
```

... and via a custom perl script using the LCFG::Client::ProfileFetcher module.

Access/permissions on LCFG server(s)

Shell access to the LCFG file server(s) is determined by `/etc/security/access.conf` and currently looks like this:

```
-:login/deny/pre-2020:ALL
+:@sysmans:ALL
+:@techs:ALL
+:root:LOCAL
-:ALL:ALL
```

The profile directories allow no access for 'other' and allow read/write access for the 'root' user and read access for the 'lcfg' group. The 'lcfg' group currently consists of all C[S]Os and the 'apache' and 'nutrun' users

'nsu' has been removed from the LCFG servers.

Code review/testing

Firstly, an obvious statement: code review is difficult to do well, particularly if you're not familiar with the code in question. I took some time to familiarise myself with the general workings of the LCFG client, specifically around profile fetching.

`rdxprof` uses LCFG::Client::ProfileFetcher to fetch a client's profile. It reads configuration from `/etc/lcfg/rdxprof.yaml`. This is where the configuration for profile fetching is defined:

```
...
fetch:
  authenticators:
    - gssapi
  params:
    gssapi:
      ccache: /var/lcfg/tmp/client/client.krb5ccache
      keytab: /etc/lcfg/client.keytab
...
```

`/etc/lcfg/client.keytab` contains keys for 'lcfg/<host>.inf.ed.ac.uk', as managed by the kerberos component.

To test the LCFG::Client::ProfileFetcher module, I decided the best approach was to use it in my own code, so I wrote a small script to test fetching LCFG profiles. This allowed me to try to do things that the LCFG client doesn't (authenticating as different principals, misconfiguring SSL/TLS, etc.). I confirmed that I could only access the profiles as intended by the server side configuration.

To test the code was dealing with certificates checking correctly, I increased debugging levels from the underlying SSL library. As the list of certificate authorities is configurable, I pointed the module at a personal copy of the trusted certificate authorities and removed the 'DST Root CA X3' certificate (this is the root certificate in the Let's Encrypt trust chain). All behaviour was as expected.

As a general comment, the code is well-written and documented using a modern style of perl, and was relatively easy to understand. The only minor issues I discovered were in inconsistent handling of ssl options, depending on how the code is used. This has been reported to Stephen.

Conclusion/Recommendations

The work done to deliver a more secure mechanism for delivery of client profiles appears to have been successful.

I have a few documentation recommendations:

The entire system for delivery of LCFG profiles is quite complex, with a number of interacting parts. It would be beneficial to have an overview document which brings all of this together.

mkxprof should extract its perldoc to a man page.

The profile (meta-)component is important in a few ways, but I don't think its use is documented. It would be useful if it was.

Total Effort

44 hours (~6 days)

Appendix: hacky script using LCFG::Client::ProfileFetcher

Warning, very hacky, use as example only.

```
#!/usr/bin/perl

# run with test-profile-fetcher , probably as root

use LCFG::Client::ProfileFetcher;
use Data::Dumper;

# set up logging
# will need to create lcfg-log-dir/test-profile-fetcher
# (e.g. /var/log/lcfg/test-profile-fetcher on ubuntu
use LCFG::Client::Log ();
my $logger = LCFG::Client::Log->GetLogger();
$logger->LogComponent('test-profile-fetcher');
$logger->DebugFlags("all");

# uncomment to see certificate negotiation debugging output
#use IO::Socket::SSL qw(debug3);

my $host = shift @ARGV;

my $fetcher = LCFG::Client::ProfileFetcher->new(
  {
   sources => "https://lcfg1.inf.ed.ac.uk/profiles",
   authenticators => "gssapi",
   ssl_opts => { verify_hostname => 1,
   # use something like this to muck around with trusted CAs
   #            SSL_ca_file => '/tmp/ca-bundle.crt' },
               },
   # we can change keytab, ccache to auth as other princ
   # auth_params => { gssapi => { ccache => '/tmp/testing.krb5ccache',
   #                              keytab => '/etc/testing.keytab' } }
  }
);

my $node = $host.'.inf.ed.ac.uk';
# need to mkdir -p this directory first
my $target = '/tmp/profile/xml/' . $node . '.xml';
my $action = 'NOTIF';

$fetcher->fetch_profile($node, $target, $action);
```

-- TobyBlake - 07 Jul 2021

This topic: DICE > FinalProjectReport555
Topic revision: r3 - 07 Jul 2021 - 14:43:12 - TobyBlake