**Project From Moonshot to Mars**

**Research Report**

**JaNET Moonshot**

*Remy Bien, Ruben Bras, Marvin Hiemstra,  
Sebastiaan Groot, Wouter Miltenburg, Koen Veelenturf*

29 May 2013,  
Version 0.1



Table of contents

[JaNET LiveCD 4](#_Toc356982603)

[Difference in code 4](#_Toc356982604)

[Network traffic 4](#_Toc356982605)

[EAP/TTLS GSS 5](#_Toc356982606)

[Scenario 1 5](#_Toc356982607)

[Scenario 2 5](#_Toc356982608)

[Interesting files 6](#_Toc356982609)

[Unlocatable files 6](#_Toc356982610)

[Points of attention 6](#_Toc356982611)

[Changed files OpenSSH 7](#_Toc356982612)

[Conclusion 7](#_Toc356982613)

[Port Janet CentOS 8](#_Toc356982614)

[Port Results 8](#_Toc356982615)

[Known Issues 8](#_Toc356982616)

[How it works? 9](#_Toc356982617)

**Introduction**

This document contains the results of the research of JaNET’s Moonshot project. The document is requested by the client in order to create an insight into JaNET’s progress on their project Moonshot.

In the chapter JaNET LiveCD the applied modification that JaNET has done to their Moonshot environment and a description of which parts are usable for our own version of the Moonshot project. The “Port JaNET CentOS” chapter will focus on the know issues and if a port to CentOS is possible. Furthermore, the way JaNET’s moonshot projects works will be described in the chapter how it works.

# JaNET LiveCD

## Difference in code

JaNET needed to make some changes in the source code of the SSH daemon, in order to make federated login possible. They created some extra files in the source code in order to make it work, and they did make some adjustment to the current source code. In the rest of the chapter we will highlight some of those changes and some of the new files.

We tried to compare JaNET’s source code to the official OpenSSH 5.9p1 source from OpenBSD. Bij comparing the different sources we were able to find out the differences between them.

One of the interesting differences in the source code is located in the “auth2-gss.c” file. The file itself exists on every OpenSSH installation, but JaNET made some changes. They added two extra classes of code for GSS key exchange, named: “userauth\_gsskeyex”, “gssapi\_set\_username(Authctxt \*authctxt)”. And they added a new authentication method: “Authmethod method\_gsskeyex”. The first new part of code is the new userauth mechanism for the GSSAPI key exchange. When a user wants to login, it will eventually return “authenticated”. This piece of code will eventually call the gssapi\_set\_username. This function will set the GSSAPI context for the username and password to the PAM module authctxt, if the option USE\_PAM is enabled.

Another file where some major adjustments have been made is the file “ssh-gss.h”. The original file contains a part for MIT Kerberos. In the version of JaNET’s source code, this part is gone. A lot of new code has been added to the file. This is especially for the SSH / GSSAPI key exchange.

## Network traffic

This chapter describes the observed network traffic between the SSH client, identity card application, SSH server and radius server on the Moonshot Pilot DVD.

The SSH client first sets up a TCP connection with the ssh server, after which they complete the SSH key exchange. Seemingly directly after the key exchange, the SSH client opens the moonshot identity card application in which the user selects an identity to “send”. At this point, either the identity card application or the SSH client itself, connects to the freeradius server and initiates an EAP-TTLS inner tunnel. Once the secure tunnel is set up, authentication against the radius server is performed and an Access-Accept is returned with a vendor-specific attribute containing a SAML assertion. This is used to finalize the authentication and the SSH server gives the user (in this case steve) the generic “moonshot” account on the server.

## EAP/TTLS GSS

### Scenario 1

In this chapter we will describe how a connection is made using gss-server and gss-client according to the Janet online how-to. (https://community.ja.net/groups/moonshot/article/moonshot-pilot-release-1-dvd) We will check if this solution uses proven technologies like GSS and EAP-TTLS to make secure authentication possible.

Test scenario 1:

Janet moonshot pilot DVD

Hostname: debian

Used commands:

gss-server -verbose gss@debian (in pts/0)

gss-client -mech ‘{1.3.6.1.5.5.15.1.1.18}’ debian gss@debian “message” (in pts/1)

freeradius -X (in pts/2, freeradius must be halted if this isn’t the case you can execute /etc/init.d/freeradius stop)

**Conclusion:**

In the radius output you will see that there is a EAP-TTLS tunnel where all information is securely transported and handled. In the output that radius produces you can deduce that the local realm is first transported without a username. Radius will then check if the realm exists in its configuration and if so, it will set-up a EAP-TTLS connection with the client. When the connection is set-up the username will be send from the client to the radius server. There it will authenticate the user.

The RADIUS log shows the information sent in the outer- and inner tunnel. The realm will be transported through the outer tunnel to the un-authenticated server. The username will be transported in the inner tunnel, which the RADIUS server will be able to see when the EAP-TTLS tunnel is set up.

When the authentication process is completed a local account named moonshot will be mapped to the user.

### Scenario 2

Janet pilot DVD (virtual machine)

Root radius server

Radius server moonshot.nl

OpenLDAP server moonshot.nl

Radtest (program to test radius network connectivity)

Changing the configuration on the JaNET pilot made it possible to successfully authenticate against another radius server in a radius chain. However, at the moment it’s not possible to securely authenticate against a RADIUS server in a RADIUS chain. One side note, we didn’t make use of the GSS integrated in the JaNET pilot. Further research is needed to figure out if a default radius server can be used in a radius chain using GSS technology inside OpenSSH.

When using Radtest, running all software in debug mode makes it possible to see the user information of the user that wants to login. For security and privacy it’s not desirable that all information is viewable in clear text.

A possible solution to address this problem is using GSS; further research will provide the necessary information to determine if Radius or OpenSSH need changes in order to accomplish this goal.

The possibility to authenticate works but mapping is not done automatically. Mapping might be possible by running a script that combines users to a unique local account.

### Interesting files

In this paragraph we list a number of interesting files:

auth2-gss.c

(Include) monitor\_wrap.h

Line 40 deleted

Added: OM\_uint32 mm\_ssh\_gssapi\_localname / OM\_uint32 mm\_ssh\_gssapi\_sign

(Include) packet.h

Line added: “packet\_connection\_is\_ipv4(void)”

mod\_auth\_kerb

mod\_auth\_gssapi.c

ssh-gss.h

radius\_example.c

eap\_example

kexgsss.c

### Unlocatable files

Here you’ll find a list of files which are not present on the JaNET live CD installation but are mentioned in either the original installations or referred to in the code.

Security/AuthSession.h

### Points of attention

A list of things that caught our attention while looking through the JaNET live CD installation.

/moonshot/moonshot/libeap/radiusexample (radius\_example.c)

Radius configuration build in c. Not yet analysed.

/moonshot/moonshot/libeap/eap\_example (certificates (.pem))

They’ve experimented with certificates as well, unknown how far they’ve got or what the results are.

## Changed files OpenSSH

Janet has changed quite a lot of the OpenSSH directory in version 5.9. They’ve removed the sandbox files and the gss-serv-krb5.c and added files like changeLog.gssapi, kexgsss.c and kexgssc.c. Janet edited a lot of files as well like gss-serv-krb5.c, auth2-gss.c and ssh-gss.h.

For the full overview of changes, please see the appendix.

## Conclusion

Due to the amount of changes JaNET did to their Test Environment and without their documentation, it is hard to pinpoint the essence of their work. We could use some of their work as an example but it takes quite some time to implement it for our purpose. This is why we need to consider if it is smart to put quite some time in it analyzing their work, if not focus on our own solution.

# Port Janet CentOS

The moonshot environment of Janet is a Debian (Wheezy) based environment. The standard operating system for Nikhef is CentOS (6). Therefore the team wanted to install the moonshot libraries and packages onto a CentOS system. Janet did make some RPMs available, but where not up to date for almost two years. The team needed to compile the source code of the moonshot libraries and packages and build it from scratch.

Initially the team thought that this would not be a big problem, but after a while the team discovered several problems with the code in combination with CentOS 6. Compiling the code took much longer than expected. The team reported major bugs to the developers of Janet.

## Port Results

The source code of the different Moonshot libraries and packages were made available on the Moonshot repository. The source code contains the following libraries and packages:

* Shibboleth XMLtooling
* Shibboleth OpenSAML2
* Shibboleth Service Provider
* Shibboleth Resolver
* Libradsec library
* Jansson
* Moonshot User Interface
* Moonshot library (including the Moonshot-gss-eap mechanism)
* Trust Router
* Moonshot OpenSSH

The source code is provided by a git repository, located at: http://www.project-moonshot.org/gitweb/

To be able to compile the code, you need a couple of dependencies. In order to compile code, you need the automake, autoconf, g++ and gcc-c++ package. Those are the basic packages. You also need some less basic packages like *boost-devel, log4cpp-devel, xerces-c-devel, libcurl-devel,* and *libtool*. Using only the CentOS repository is not enough. You need the EPEL repository too. This repository contains a lot of extra packages. Here is an overview of the required packages:

* gcc
* boost-devel
* log4cpp-devel
* xerces-c-devel
* automake
* autoconf
* libtool
* libcurl-devel
* libconfuse-devel
* gettext-devel
* gtk2-devel
* shared-mime-info
* desktop-file-utils
* libgee
* libgee-devel
* dbus-glib
* dbus-glib-devel
* openssl-devel
* sqlite-devel (for the trust-router)
* libtalloc-devel (for freeradius-server)

Some of the required packages needed to be compiled by hand. The libevent pacakage, which you can install using yum is not the right version. You need libevent2. The team used the version libevent-2.0.21-stable.

You also need the xml-security-c package by Apache Santuario. This package is, just like libevent, installable using yum, but the yum repostitory does not contain the right version. The team used the last version available, vesion xml-security-c-1.7.0.

After preparing the system, you can clone the source code of Janet using the moonshot git available at: http://www.project-moonshot.org/git/moonshot.git. You need to execute the following commands when you cloned the code onto your system:

*cd moonshot  
git submodule init  
git submodule update*

Janet created a builder script, that automatically compiles the moonshot code in the right order.

## Known Issues

Janet created the code for the libraries and packages especially for Debian Wheezy systems. Compiling the code on CentOS could be very problematic. A couple of important dependencies have a newer version on Debian, which are not available (yet?) on CentOS. Compiling those packages will give a bunch of extra problems. Janet released a couple of new RPMs in early May 2013, but those packages will not work out of the box either.

One of the other package issues was *libevent*. Libevent is available at the yum repository, but not the required version. Moonshot code uses the one of the newest versions of *libevent*, libevent-2.0.21-stable. When you compile this by hand, it will also include the required ssl libraries.

A major issue of the moonshot user interface (on Debian), is that the user interface will only appear once, if you signed in successfully. The team reported this bug to the moonshot-ui developers, and they are working on this problem. Another major bug of the Moonshot UI on CentOS is that it will not run after installing the RPM or after compiling the code. This is caused by a GDBUS problem. This is probably a problem of the *glib* package, due to different versions. The team reported this issue to the developers of the Moonshot UI. When the team tried to compile everything on a new system this did not happen, and the Moonshot UI did appear. However, the SSH client did not work as it should. The Moonshot UI will not appear, like it should. We do not understand why this event happened.

A major difficulty for our team was the fact that Janet did not document a lot about their code and packages. It was very difficult for the team to understand the different kinds of code and the way it works. Understanding the code took a very long time.

The SSH server and/or client is also pretty buggy on CentOS. When you compile the code from the Moonshot repository, you will not be able to use the Moonshot SSH client out of the box. Somehow the team was not able to get it to work. When trying to access the Moonshot SSH server, this error will appear (in debug mode):

…

debug1: SSH2\_MSG\_SERVICE\_REQUEST sent

debug2: service\_accept: ssh-userauth

debug1: SSH2\_MSG\_SERVICE\_ACCEPT received

debug2: key: /home/moonshot/.ssh/id\_rsa ((nil))

debug2: key: /home/moonshot/.ssh/id\_dsa ((nil))

debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password

debug1: Next authentication method: gssapi-keyex

debug1: No valid Key exchange context

debug2: we did not send a packet, disable method

debug1: Next authentication method: gssapi-with-mic

debug1: Unspecified GSS failure. Minor code may provide more information

Credentials cache file '/tmp/krb5cc\_500' not found

debug1: Unspecified GSS failure. Minor code may provide more information

Credentials cache file '/tmp/krb5cc\_500' not found

debug1: Unspecified GSS failure. Minor code may provide more information

debug1: Unspecified GSS failure. Minor code may provide more information

Credentials cache file '/tmp/krb5cc\_500' not found

debug2: we did not send a packet, disable method

debug1: Next authentication method: publickey

debug1: Trying private key: /home/moonshot/.ssh/id\_rsa

debug1: Trying private key: /home/moonshot/.ssh/id\_dsa

debug2: we did not send a packet, disable method

debug1: Next authentication method: password

…

# How it works?

Welke packages vallen er onder de Moonshot packages?

Wat is het GSS\_EAP\_MECH?

Wat is de trust-router?