**Project From Moonshot to Mars**

**Research Report**

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

29 May 2013,  
Version 0.1



**Management summary**

Yada Yada Yada

Table of contents

[Janet Moonshot 5](#_Toc353807386)

[Network traffic 5](#_Toc353807387)

[EAP/TTLS GSS 6](#_Toc353807388)

[Overview changed files OpenSSH 8](#_Toc353807389)

[STrace & LTrace 9](#_Toc353807390)

[Conclusion 9](#_Toc353807391)

[Port Janet CentOS 10](#_Toc353807392)

[Port Results 10](#_Toc353807393)

[Configuration 10](#_Toc353807394)

[Known Issues 10](#_Toc353807395)

[FreeRadius Module 11](#_Toc353807396)

[Used Library’s 11](#_Toc353807397)

[Module 11](#_Toc353807398)

[Encryption 11](#_Toc353807399)

[Certificates 11](#_Toc353807400)

[Conclusion 12](#_Toc353807401)

[Appendix 13](#_Toc353807402)

[Output 13](#_Toc353807403)

**Introduction**

Yada yada yada

The protocols that we needed to study were: SSH, GSSAPI, etc

# Janet Moonshot

The British organization JaNET is currently working on a Moonshot project as well. Since we do a similar project, it is to our interest to analyze their code to see what they have done in order to make it work.

**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 the different source codes using the source version of JaNET’s SSH source code and the official OpenSSH 5.9p1 source code from OpenBSD. Therefore we were able to compare the two sources and see what the differences are.

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 did some changes. They added two extra classes of code for GSS key exchange, named: “userauth\_gsskeyex”, “gssapi\_set\_username(Authctxt \*authctxt)”, and 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 an 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 an 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 an username. Radius will then check if the realm exists in his configuration and if so, it will set-up an EAP-TTLS connection with the client. When the connection is set-up the username will be send from the client to the radius server and thereon it will authenticate the user.

The radius log shows what information is send in the outer and inner tunnel. In the outer tunnel the realm will only be transported to the, from then, un-authenticated server (no EAP-TTLS connection exists from then). In the inner tunnel, which the radius server can see when the EAP-TTLS tunnel is set-up, the username will be transported.

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

**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 for successfully authenticating against another radius server in a radius chain. However, it is not possible at the moment to securely authenticate against an 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 if a default radius server can be used in a radius chain using GSS technology inside OpenSSH.

When using Radtest, you’ll run all software in debug mode it’s possible to see the user information 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 needs to be changed 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 room-account.

**Interesting files**

In this paragraph is a list with interesting files, which we could find, and of which we think they are interesting:

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.

## Overview changed files OpenSSH

|  |  |  |
| --- | --- | --- |
| Removed files | Changed files | Added files |
| /moonshot/openssh | **/moonshot/openssh** | **/moonshot/openssh** |
| gss-serv-krb5.c | auth2.c | changeLog.gssapi |
| sandbox-darwin.c | auth2-gss.c | kexgssc.c |
| sandbox-null.c | authfile.c | kexgsss.c |
| sandbox-rlimit.c | changelog | ssh\_prng\_cmds.in |
| sandbox-systrace.c | clientloop.c | ssh-rand-helper.0 |
| ssh-sandbox.h | configure | ssh-rand-helper.8 |
|  | configure.ac | ssh-rand-helper.c |
|  | entropy.c | WARNING.RNG |
|  | gss-genr.c |  |
|  | gss-serv-krb5.c |  |
|  | kex.c |  |
|  | kex.h |  |
|  | makefile.in |  |
|  | monitor.h |  |
|  | mux.c |  |
|  | PROTOCOL.mux |  |
|  | servconf.c |  |
|  | servconf.h |  |
|  | sftp.1 |  |
|  | ssh.c |  |
|  | ssh\_config.0 |  |
|  | ssh\_prng\_cmds.in |  |
|  | ssh-agent.c |  |
|  | sshconnect.c |  |
|  | sshconnect2.c |  |
|  | sshd.c |  |
|  | ssh-gss.h |  |
|  | ssh-keygen.0 |  |
|  | ssh-keygen.1 |  |
|  | ssh-keygen.c |  |
|  | **/moonshot/openssh/contrib/cygwin** |  |
|  | ssh-host-config |  |
|  | ssh-user-config |  |
|  | **/moonshot/openssh/contrib./redhat** |  |
|  | sshd.init |  |
|  | **/moonshot/openssh/contrib./suse** |  |
|  | openssh.spec |  |
|  | **/moonshot/openssh/openbsd-compat** |  |
|  | bsd-cygwin\_util.c |  |
|  | **/moonshot/openssh/regress** |  |
|  | cfgmatch.sh |  |

## STrace & LTrace

## 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

Yadayadayada

## Port Results

Yada yada yada

## Configuration

Yada yada yada

## Known Issues

Yada yada yada

# FreeRadius Module

Yadayadayada

## Used Library’s

yadyadayada

## Module

Yada yada yada

## Encryption

Yada yada yada

## Certificates

Yada yada yada

# Conclusion

Yada yada yada

# Appendix

## Output