

CESNET Technical Report 1/2008

Effects of Incorrect EAP Termination in eduroam

JAN TOMÁŠEK

Received 19. 3. 2008

Abstract

Extensible Authentication Protocol (EAP) is a universal authentication framework frequently used in wired and wireless networks. EAP is used to build secured tunnels between a user device and the authentication server. With some EAP methods user credentials are being verified after the secured tunnel is successfully build. If such EAP method is used and if the authentication server is misconfigured then this server can be tricked to forward user credentials to another server. If a malicious user discovers such a server, then he can use this server as an anonymizing proxy for hiding his true identity.

Keywords: Eduroam, EAP termination, security

1 Introduction

The *eduroam* is an IP roaming infrastructure build up to allow users of one organisation to connect to a WiFi network in another organisation. The authentication is always done in the home organisation of the user trying to access the network. The home organisations in the *eduroam* are also called the Identity Providers (IdP) and the visited organisations are called the Service Providers (SP). To allow user to authenticate, before he gets an IP connectivity, the IEEE 802.1X protocol EAP was chosen. The IEEE 802.1X standard uses EAP as a method for allowing users to build secured tunnel to their IdP. Secured tunnels consist of number of EAP messages, which are transparently transported through the whole *eduroam* RADIUS infrastructure. Devices participating on transporting EAP messages are not able to understand the messages which they transport. Only the user device and the IdP has keys for decrypting transported data. That assures the high level of protection to the user credentials.

User identity in the *eduroam* consists of a username and a realm. The realm is used to route EAP messages to the right IdP. For example: A user with the identity `semik@orgC` visits premises of the `orgA` and wishes to access their WiFi network. He opens his laptop and starts a supplicant program. The supplicant is a program which automatically communicates with a NAS (WiFi router) identifying the user as `semik@orgC`. The NAS transparently passes all EAP messages to the configured RADIUS server. Because the realm of user's ID doesn't match the realm of the RADIUS server in the `orgA`, it passes EAP messages to the *eduroam* infrastructure, which delivers them to the user's home RADIUS. The user's home RADIUS verifies the user credentials and responds by sending the Access-Reject message for rejecting the network access or by sending the Access-Accept message for allowing the network access.

The routing of EAP messages depends on the realm used in the user's ID. That is a very simple idea which allowed to build up the whole *eduroam* infrastructure. It was assumed that when the RADIUS server issues the Access-Accept for a user then it's granted this user really belongs to the same IdP as the issuing RADIUS server. I have discovered the method which allows me to trick a misconfigured RADIUS server to issue the Access-Accept for non-existent or faked user identity. The problem arises when the RADIUS server is misconfigured in such a way, which allows proxying the internal EAP authentication back into the *eduroam* infrastructure, instead of verifying them locally. For the detailed explanation I prepared an image:

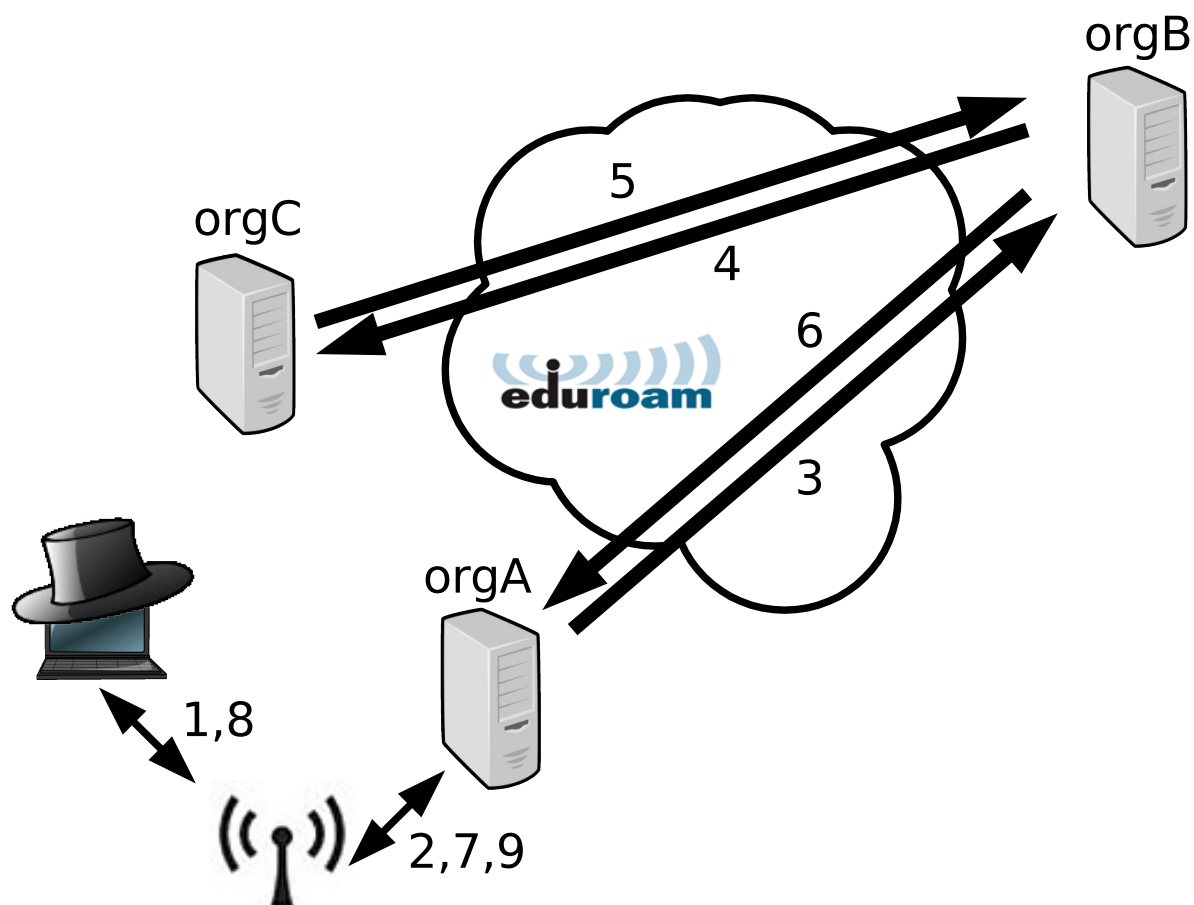


Figure 1. Incorrect EAP termination in the *eduroam*

A procedure is as follows:

1. The malicious user `semik@orgC` visits the `orgA`. He gets associated to the WiFi router of the `orgA` and identifies himself as `vcelka-maja@orgB`.
2. EAP packets with the outer identity `vcelka-maja@orgB` are being passed from the user's supplicant via the WiFi router to the RADIUS server of the `orgA` locality.
3. As the realm `orgB` doesn't match the realm of the local RADIUS of the `orgA`, it proxies EAP packets into the *eduroam* infrastructure, which delivers them to the RADIUS server responsible for the realm of the `orgB`. The RADIUS server

of the orgB terminates EAP session because the realm of the outer identity is defined as local for it. After that the orgB RADIUS server sees the inner authentication payload which identifies the user semik@orgC.

4. Because of the mistake in the setup of RADIUS server at the orgB, the server passes authentication payload back to the *eduroam*, which delivers it to the RADIUS server in the orgC.
5. The RADIUS server of the orgC knows semik@orgC and responds by sending the Access-Accept message, if the user's password is correct, off course.
6. The RADIUS server of the orgB receives the Access-Accepts from the orgC and responds to the orgA with the Access-Accept as if the user was local to the orgB. That means that the outer identity is still vcelka-maja@orgB and most likely the User-Name attribute of the Access-Accept also identifies the user as vcelka-maja@orgB.
7. The Access-Accept packet is passed to the NAS, which permits the network access to the user's device.
8. The user's supplicant gets the WiFi encryption keys from the Access-Accept packet and the user semik@orgC can start using the IP connectivity of the orgA cloaked as vcelka-maja@orgB.
9. The WiFi router sends the Accounting-Start packet for the user vcelka-maja@orgB.

2 Test methods

During my work on the position of administrator of the Czech NREN RADIUS server I have seen several cases of this type of the *eduroam* abuse. For proving the concept described above I developed several test methods. Bellow I'm describing a method which can be used by a malicious user for cloaking his identity. Later in the second part of the text I'm describing methods for detecting the misconfigured servers.

2.1 wpa_supplicant config

To demonstrate this attack we need a supplicant that supports the setting of the different inner and outer user identity. There is a large number of supplicants supporting that. I will show here an example configuration for the wpa_supplicant¹ which is commonly used on Linux systems.

The config file I used for testing:

```
network={
ssid="eduroam"
scan_ssid=1
key_mgmt=WPA-EAP IEEE8021X
eap=TTLS
#or eap=PEAP
pairwise=CCMP TKIP
group=CCMP TKIP WEP104
```

¹ http://hostap.epitest.fi/wpa_supplicant/

```

anonymous_identity="vcelka-maja@orgB"
identity="semik@orgC"
password="tajemstvi"
phase2="auth=MSCHAPv2"
#or phase2="auth=CHAP"
#or phase2="auth=PAP"
}

```

2.2 rad_eap_test

Editing the `wpa_supplicant`'s config file on laptop isn't very convenient for automated testing. The `rad_eap_test`² script is much better suited for these tasks. The script uses the `eapol_test` from the `wpa_supplicant` and offers a simple command line interface.

The example of usage of the test with the TTLS-MSCHAPv2 method:

```

rad_eap_test -H <server-name> -P 1812 -S <shared-secret> \
-m WPA-EAP -e TTLS -u <semik@orgC> -p <tajemstvi> \
-A <vcelka-maja@orgB> -i "vcelka-maja test" -t 15

```

An example of usage of the test with the PEAP-MSCHAPv2 method:

```

rad_eap_test -H <server-name> -P 1812 -S <shared-secret> \
-m WPA-EAP -e PEAP -u <semik@orgC> -p <tajemstvi> \
-A <vcelka-maja@orgB> -i "vcelka-maja test" -t 15

```

When the server `<server-name>` is misconfigured then the script will print `access-accept; <delay in sec>`. You may use `-v` or `-c` switch to print the last RADIUS packet or all RADIUS packets.

2.3 Automated tool for NREN RADIUS admins

Administrators of large RADIUS proxy networks might be interested in automated tool I'm using for checking all the organisational servers connected to the Czech *eduroam*. All that my script needs is a working test account and the list of the realms to be tested. I named my script `vcelka-maja.sh`³ and it is published on the Czech *eduroam* portal⁴. The script requires the config file `/etc/vcelka-maja.conf` with the following parameters:

A working test account. The home RADIUS server of this account must be capable of doing MSCHAPv2 authentication.

```

TUSER='user@orgC'
PASSWORD='tajemstvi'

```

² http://www.eduroam.cz/rad_eap_test

³ <http://www.eduroam.cz/vcelka-maja/>

⁴ <http://www.eduroam.cz/>

The name of RADIUS server connected to the RADIUS proxy network (*eduroam*). This server is used as a gate to the *eduroam*. If script is used on the NREN RADIUS server it should be set to `localhost`.

```
SRV='radius.orgA'
SECRET='xxx'
```

Path to the `rad_eap_test`⁵ script.

```
RAD_EAP_TEST=/usr/local/rad_eap_test-0.21/rad_eap_test
```

It's possible to tune `$PATH` where the executable `eapol_test`⁶ is stored.

```
PATH="$PATH:/usr/local/bin"
```

When the script is executed it first tests if the provided testing account is valid. After that it starts reading realms from the standard input and executes TTLS-MSCHAPv2 and PEAP-MSCHAPv2 tests with 2 sec delay. The result of the tests is being written at standard output in a simple form:

```
Testing testing account: access-accept; 1
Testing PEAP orgB: access-accept; 0
Testing TTLS orgD: access-accept; 1
Testing PEAP orgD: access-accept; 2
Testing TTLS orgE: access-reject; 4
Testing PEAP orgE: access-reject; 3
Testing TTLS orgF: timeout; 15
Testing PEAP orgF: timeout; 15
```

The lines with `access-accept` identify the RADIUS servers with the incorrect setup of EAP tunnel termination. The lines with `access-reject` identify the correct RADIUS server setup and the lines with `timeout` represent the servers that were not responding at the moment of testing. The number at the end of the line show how much time testing took (in seconds).

It is advisable to use this or similar script for periodic tests of all servers connected to the *eduroam*. For example on the Czech National RADIUS server I've set up this cron job:

```
cat /etc/radiator/radius-auto.conf | grep '^#SRVLST' | \
  cut -d " " -f 4 | sed "s/^@//" | sort -u | \
  /root/scripts/vcelka-maja.sh | grep access-accept
```

This job is able to extract all realms defined on the Czech NREN RADIUS server automatically and to do tests of all connected RADIUS servers. It's being executed weekly and the report is being sent to me by email.

⁵ http://www.eduroam.cz/rad_eap_test

⁶ http://www.eduroam.cz/rad_eap_test/eapol_test

3 Configuration fixes

I'm going to describe fixes for two types of the RADIUS servers I have known. Administrators of other server types should search for the solution how to stop their servers from proxying internally inserted requests.

3.1 Radiator

Due to the way Radiator implements PEAP methods it's impossible to misconfigure it. In case of TTLS methods a misconfiguration might happen but because of a good example configuration in the *eduroam cookbook* and also in the goodies directory of its distribution package I think that the number of misconfigured Radiator RADIUS servers will be rather low.

PEAP/TTLS methods should be configured this way:

```
<Handler Realm=/^orgB$/i>
    AuthBy CheckLDAP
...
</Handler>
<Handler TunnelledByTTLS=1>
    AuthBy CheckLDAP
...
</Handler>
<Handler TunnelledByPEAP=1>
    AuthBy CheckLDAP
...
</Handler>
```

3.2 FreeRADIUS 1.x.x

FreeRADIUS internally proxies the internal EAP payload to localhost in the similar way as Radiator, but with FreeRADIUS it is possible to misconfigure both PEAP and TTLS methods. To prevent that, it is necessary to put the following lines at the beginning of users file:

```
DEFAULT FreeRADIUS-Proxied-To == 127.0.0.1, Proxy-To-Realm := LOCAL
Fall-Through = Yes
```

The condition `FreeRADIUS-Proxied-To == 127.0.0.1` matches to all internally proxied packets, the command `Proxy-To-Realm := LOCAL` marks all matched packets to be processed locally. The `Fall-Through = Yes` is necessary so that the existing users file functionality wasn't broken.

The author of this configuration directive is Stefan Winter from RESTENA. This fix was consulted in the SA5-GN2 mailing list, and successfully tested on several FreeRADIUS installations.

It's necessary to check if the FreeRADIUS installation is using users file to be sure that the fix will work.

4 Conclusion

I have discovered that many FreeRADIUS 1.x.x installations are vulnerable to similar attacks. It can be due to the fact that the example configuration in the *eduroam cookbook* is not resistant against such attacks. I tested several deployments of RADIUS 3.xx and 4.xx, Cisco RADIUS, Microsoft IAS and FreeRADIUS 2.0. They do not seem to be vulnerable to these attacks in commonly used configurations.

I have done intensive testing in the Czech *eduroam* and also in the world wide *eduroam*. In the Czech *eduroam* I tested all 66 connected RADIUS servers, 28 of them were misconfigured. In the world wide scope I used accounting database to get the list of the realms which visited the Czech sites in the past. There were 99 lines in the list. 35 sites are/were misconfigured.

Finally I would like to say: It is always possible to discover the real identity of the malicious user. The quality of proofs will vary with the quality of logs kept at misconfigured site. The records from log files from another RADIUS servers participating on communication might serve as another proof. Tracking malicious users in logs of a NREN RADIUS server is not easy and because it requires other RADIUS servers admins cooperation, it is very time consuming.

I would suggest to all NREN level RADIUS servers administrators to do periodical testing of all institutional RADIUS servers connected to their NREN RADIUS server. And to ask admins of misconfigured servers to fix their setup as soon as possible.

Some FreeRADIUS installations fill the correct inner identity into the Username attribute of final the Access-Accept sent to visited organisation. This can help to identify the user is doing something wrong. The user has no way to discover if such attribute is being sent. That means that the user, unless he is an *eduroam* administrator, can never be sure if his cheating succeeded.