Using Database to Manage Local Access Users on Network Assets

Fábio Oliveira dos Santos[1], **Alberto Savio**[1], **Marcos Araujo**[1] **e Natalia Castro Fernandes**[1]

[1]Grupo Midiacom - PPGEET/Escola de Engenharia
Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

{santosfabio,savio,marcosaraujo,nataliacf}@id.uff.br

Abstract. *Nowadays, we have different efficient access control systems to manage remote access to network assets. Such systems are usually based on a centralized authentication server available throw network. Nevertheless, there is a challenge when the network is not available and there is a need for local access, which demands a local login. The scale of this problem is even greater in scenarios of service providers, which manage thousands of network assets. On one hand, the network is under risk if all devices use the same login/password for local access. On the other hand, a different login/password to each device is unmanageable. We propose a tool that manages local user authentication on network equipment. Our system is based on a validation mechanism for the local accesses in an automatic and centralized way. With our proposal, we can work around the security risks of local users without causing major impacts and costs in the daily operation of the network regardless of the size of the network. Since our proposal is based on automatic validations of users and their information even though this information cannot be tested directly on the network. The main go of our proposal is to generate and manage a unique set of username and password for each network element so that they can only be used once for local access to the network elements whenever they are without access to the servers authentication. In our proposal the OTP passowrds functions (one time passwords) do not need to be implemented by the network elements. We implemented and tested our proposal, validating the proposed approach. Our implementation was realized an Intel machine, with Ubuntu operating system, where we were able to generate and manage users and passwords for more than 5 thousand routers from Cisco manufacturer.*

1. Introduction

Most current networks use the TACACS++ (Finseth 1993) or Radius protocols (Rigney 2000) to provide centralized authentication for accessing network equipment. These protocols are usually executed in centralized platforms and provide frameworks for auditing and verifying logs related to the accesses. The TACACS++ protocol, in addition to authentication, also supports verifying the authorization of each command executed by the user during the session based on the user profile. Problems arise when the platform that provides the centralized authentication service is somehow inoperable or inaccessible, due to failure on the platform or in the infrastructure that provides the connectivity between the platform and the network equipment. Because of the unavailability of the remote authentication service, the username and password verification of each access is under the local authority of each network equipment.

Managing local user accounts is not simple since this set of usernames and passwords needs to be well known by operators during failures on the remote authentication servers, but this same set of credentials is not used during normal operation. Indeed, the number of devices in a network can exceed thousands, which usually entails creating the same set of user and password locally on all network equipment. Having the same username and password in all devices however is not advisable because this information could be used by unauthorized agents to access any device in the network once disclosed. On the other hand, a decentralized choice of authentication data without a central authority control does not work when there are large teams of operators and a large number of devices. Hence, it becomes clear the need for an automated process.

Besides the need to keep this set of user and password securely configured for an eventual use, there is also the need to verify the authenticity of this configuration, since it can be modified in a malicious way and this modification can not be identified in the normal operation processes of the network. Indeed, most devices won't generate alerts in case of changes in the local accounts. As network managers do not have an effective way to maintain and audit these local sets of login and password, this turns into a huge vulnerability to the network.

Due to these facts, we propose a framework that can manage the generation of groups of credentials in a way that these are unique by network equipment and, still, provide easy and safe access. To do this task, our proposal relies on a web application with full (Read and Write) access to network equipment and internal database. Our proposed framework bring the following advantages:

1. Generation of sets of credentials for each device;
2. Deployment of this set on the proper network equipment;
3. Identification of the hash generated by the equipment to each password of the pool, storing such information on a central database;
4. Safe and easy access to these sets of credentials for network operators;
5. Generation and deployment of new sets whenever a set is used by a network operator;
6. Verification whether the sets configured on network devices are up-to-date with information stored in local base; In case of inconsistencies, generation and deployment a new set of users and passwords.

We implemented and tested the proposed framework. The results show the consistence of the developed tool, which counts on a web interface to interact with operators and provide a set of scripts to configure, update, and verify local credentials.

The remaining of this paper is organized as follows. Section 2 shows the state-of-art in local credential management of network assets. Section 3 presents the proposed framework and Section 4 describes the implementation details. Section 5 provides a detailed analysis of the developed software. Finally, Section 6 concludes the paper and present future works.

2. Related Work

Multi-user access to computer systems is a well-discussed theme, which already resulted in different system such as the aforementioned RADIUS and TACACS++, among other classic systems such as Kerberos (Steiner et al. 1988). Hildebrandt and Saake addresses the problem of authentication in multidatabase systems (Hildebrandt and Saake 1998). The authors made an excellent job showing form of inter-operable authentication between databases. In theses cases, authentication information is distributed through different databases, which brings challenges in how to correlate data or to define levels of access. These are well-known problems for distributed or federated authentication databases (Silva et al. 2018; Silva et al. 2014).

Liang and Wang proposed a control scheme to locally authorize inter-domain roaming users in wireless networks. They developed a detailed procedure to establish local security associations (SAs) considering the traffic and mobility patterns of a mobile user (MU), as well as the number of hops between the MU and its home AAA server (Liang and Wang 2004).

Kemper proposed a system for local and remote authentication based on a local client database and a remote server database. The system updates the client database according to results of the local and remote authentication frequency. Hence, if a user usually uses the remote authentication, his/her account is copied to the local authentication database. Even though the proposal deals with important issues such as the selection of the proper user accounts for local access, it depends on the modification of authentication protocols in devices, which is not possible when considering current networks (Kemper 2001). Other approaches are based device-based authentication. In these cases, a small device such as a tablet or a mobile phone stores the credential and perform the local authentication or reinforce a password-based authentication using a wired or wireless technology (Me et al. 2005). Although this approach has many applications, current network assets such as routers and switches are not ready for this kind of local authentication.

Our system uses a different approach for local authentication then the previous works. Instead of creating local accounts for different users, our proposal do frequent updates of sets of passwords in all managed devices. The sets are available for operators, making it easy and secure to access distributed network assets. The sets of usernames and password receive a status of OK, only if the hash on network equipment is the same stored in the database and if this set is not used yet.

Network Security is a vital part of any corporate and enterprise network. Network attacks greatly compromise not only the sensitive data of the consumers but also cause outages to these networks. Thus inadequately protected networks need to be “hardened”. (Me et al. 2018). This work shows the importance of security in enterprise networks and the protocol’s and config changes that needed.

3. Proposed Framework

The difficulty in using locally configured users during access or failures in the TACACs or Radius server comes from the need to manage users on the various devices spread across the network infrastructure. Because of this, it is very common to have only one secure user configured on the network to provide access when centralized authentication is out of operation and/or access to it is unavailable. Since current networks have a large amount and variety of network equipment, deploying and managing local authentication databases become a difficult and impractical task. In addition to providing security risks since we would only have one user/password configured on all network elements, even if that user's use is only possible during the interruption of centralized access.

We propose a framework to generate and manage a set of unique credentials per equipment. In our application we keep the sets of users and password already used in the internal database, but with the status set to INVALID, and only the valid sets are also configured and keeped in the network devices. Our proposal also allows easy and secure access to such information to the authorized users.

Our framework, which runs on a centralized server, is divided into three blocks, as shown Figure 1. The first block is an internal database (INTERNAL_DB) that stores the current local databases of the network elements. This information is used to frequently check whether there was a non-authorized manual change in the local database of any network asset. The second block is a web server (HTTP_SERVER) that provides network operators with secure access to configured sets of users/passwords on network equipment. The third and main block (PROCEDURE MANAGER) has the set of procedures and routines necessary to provide all the intelligence of the system. Hence, it monitors local database of all network assets, generates and registers new sets of local credentials, disables old sets of local credentials and registers data for accountability.

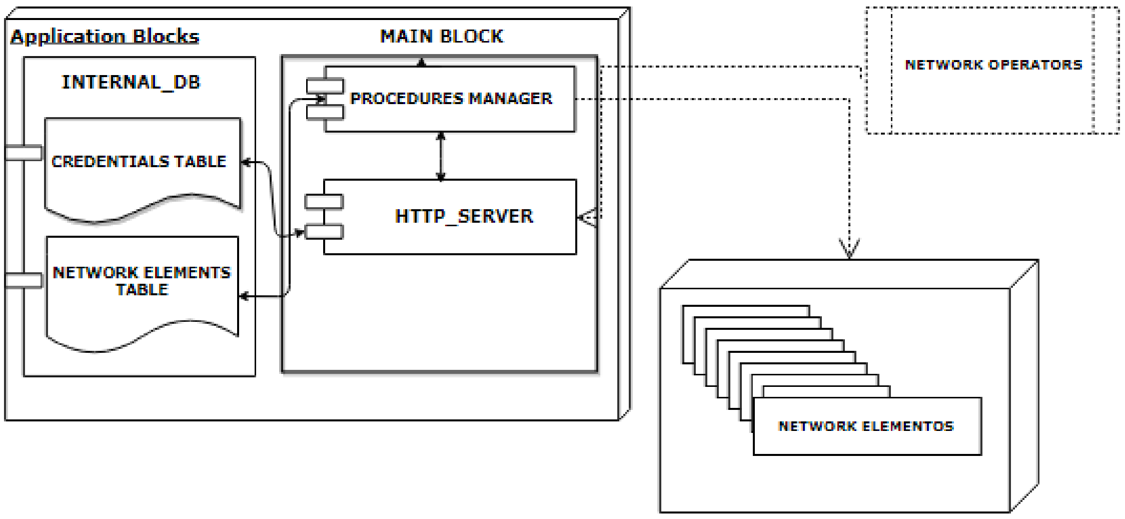


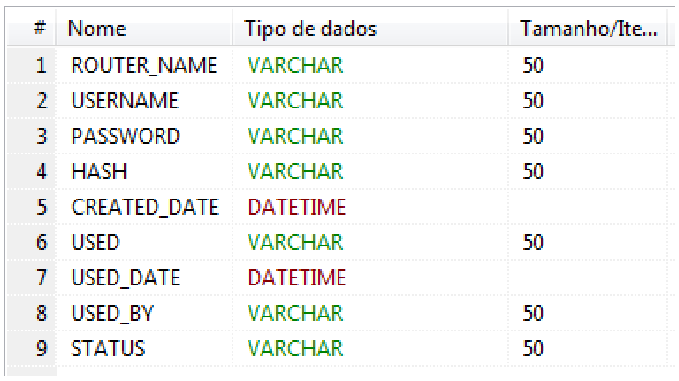
Figure 1. Components of the proposed Solution.

Our database uses two tables: one for network devices and one for credential information. The PROCEDURE_MANAGER interacts with both tables, since it must know the information to access network devices, as well as it must verify the consistency between credentials in database and in network elements. Besides this block also changes credential sets in network devices, which demands writting new data to credential table. In the same figure we see on block of procedures that interact with network equipment's, internal database and provide on web service that will be used by network operators.

The PROCEDURE_MANAGER can change, monitor, and audit users configured to locally access network devices. Hence, this block matches the following requirements:

1. Generate a random set of username and password for each network device;
2. Provide username and password for operators for a specific device whenever required;
3. For each usage of the username and password provided for an operator in a specific device, a new set must be generated and configured in the network device;
4. Ensure periodic verification of the integrity of local authentication database to avoid manual insertions/deletions of users and passwords;
5. Remove a set of username and password from a device when this set has fail on the periodic verification;
6. Provide a web interface to securely provide access to credentials for an authorized network operator.

In order to meet the above requirements our framework stores not only the username/password of each network device, but also usage information, validations, and status of each set. And a set of procedures that can be used to validate local users for their security as well as generate new sets and deploy them to their respective network equipment. Figure 2 shows an overview of the credential table fields.

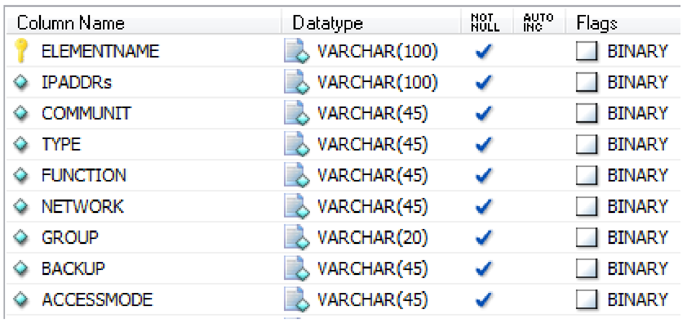


#	Nome	Tipo de dados	Tamanho/Ite...
1	ROUTER_NAME	VARCHAR	50
2	USERNAME	VARCHAR	50
3	PASSWORD	VARCHAR	50
4	HASH	VARCHAR	50
5	CREATED_DATE	DATETIME	
6	USED	VARCHAR	50
7	USED_DATE	DATETIME	
8	USED_BY	VARCHAR	50
9	STATUS	VARCHAR	50

Figure 2. Credential table structure, which stores user and password information.

We also need a table with important information of network devices such as hostname, IP address, etc. This information, which is stored in the Network Device Table, will be used to access the devices, but also in defining which equipment should have local users, assuming that it may be convenient to provide the user set and password for all network equipment. In this way, we adopted the use of two tables to compose our database. A table, called ”ELEMENTOS TABLE”, was intended as network equipment information. And another called ”CREDENTIAS TABLE”, was destined as information related to the username of the local set /password. Figure 3 shows the structure of this table.

Note that the Network Device Table contains the information of the network devices and also other data not previously related. Such data could be used on future applications that requires user policy and restrictions on the scope of the creation of the local user password.



Column Name	Datatype	NOT NULL	AUTO INC	Flags
ELEMENTNAME	VARCHAR(100)	✓		☐ BINARY
IPADDRs	VARCHAR(100)	✓		☐ BINARY
COMMUNIT	VARCHAR(45)	✓		☐ BINARY
TYPE	VARCHAR(45)	✓		☐ BINARY
FUNCTION	VARCHAR(45)	✓		☐ BINARY
NETWORK	VARCHAR(45)	✓		☐ BINARY
GROUP	VARCHAR(20)	✓		☐ BINARY
BACKUP	VARCHAR(45)	✓		☐ BINARY
ACCESSMODE	VARCHAR(45)	✓		☐ BINARY

Figure 3. Network Device Table structure, which holds all network equipment information.

3.1. Detailed Operation

Figure 4 shows the flow chart of the proposed framework. First, it starts three threads: the HTTP_SERVER, a local database integrity check, and a automatic local database update. These last two threads are part of the PROCEDURE_MANAGER.

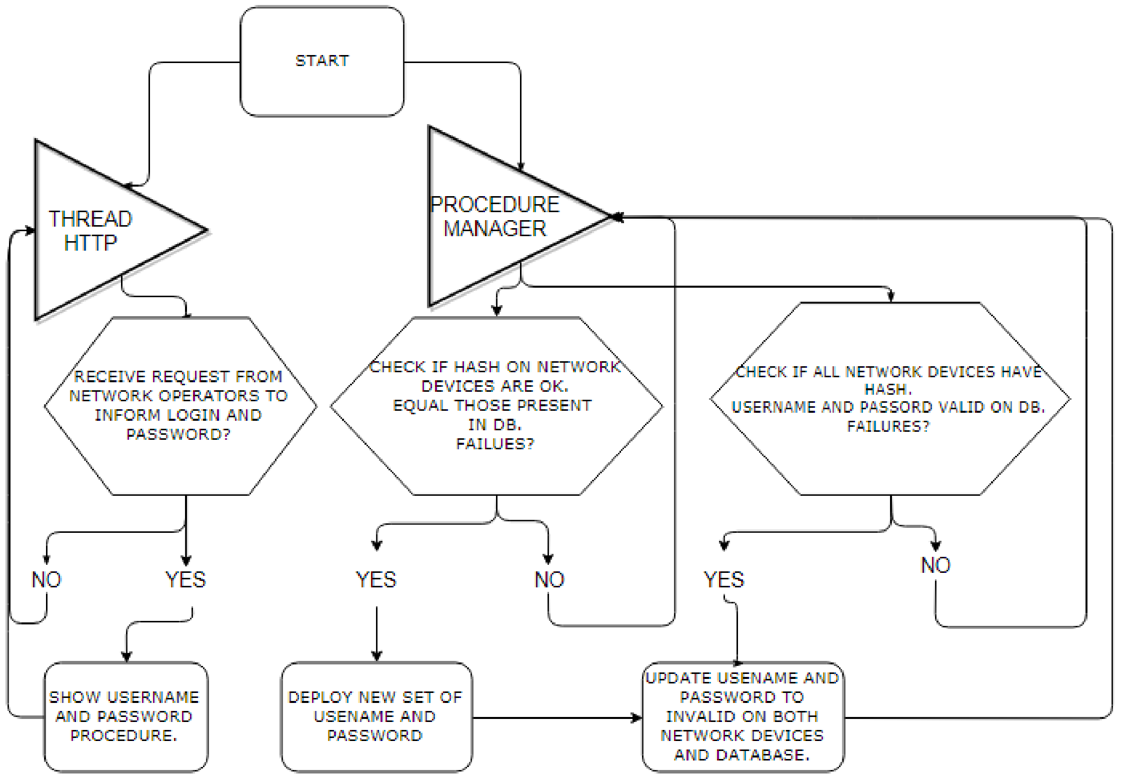

START
THREAD HTTP
PROCEDURE MANAGER
RECEIVE REQUEST FROM NETWORK OPERATORS TO INFORM LOGIN AND PASSWORD?
CHECK IF HASH ON NETWORK DEVICES ARE OK. EQUAL THOSE PRESENT IN DB. FAILUES?
CHECK IF ALL NETWORK DEVICES HAVE HASH. USERNAME AND PASSORD VALID ON DB. FAILURES?
NO
YES
YES
NO
YES
NO
SHOW USERNAME AND PASSWORD PROCEDURE.
DEPLOY NEW SET OF USENAME AND PASSWORD
UPDATE USENAME AND PASSWORD TO INVALID ON BOTH NETWORK DEVICES AND DATABASE.

Figure 4. High level diagram of the framework.

In the HTTP_SERVER flow, first, the network operator authenticates himself in order to prove he can access local credential data. Next, the network operator selects a device and receives a corresponding credential. Next, the HTTP_SERVER accesses the database and sets that credential entry as invalid, in order to guarantee that it will be removed from the local database of the network device as soon as the network equipment restores the connection to central authentication servers, such as TACACS or RADIUS.

Hence, we guarantee that the local credential will be valid only when the central authentication servers are unavailable. This is of special importance, because we want to assure that the network operators will use the central authentication servers whenever they are available to network assets.

The second flow checks local database integrity on each network device. This is a timed operation, and the interval between checks is configurable. Hence, the PROCEDURE_MANAGER checks all network device IPs in the Network Device Table and access each element to obtain the current authentication database. Next, it compares the collected data to the Credential Table information. It is important noticing that this operation does not check the password itself, but the stored password hash, because different devices may use different hash functions. Hence, it is easier to check the stored hash than the password itself. In case the local and remote database differs, then the PROCEDURE_MANAGER sets all entries on the Credential Table for that device as invalid and create a new set of credentials for that device on the Credential Table.

The third flow is also part of the PROCEDURE_MANAGER and performs an automatic update of the local databases. Therefore the PROCEDURE_MANAGER looks in the Credential Database for invalid and non-deployed credential entries. Whenever a invalid credential is found, the PROCEDURE_MANAGER accesses the corresponding device and deletes it from the local database. When a non-deployed credential is found, it is added to the corresponding local database.

Summarizing, our framework follows the methodology:

- In normal network situation, it periodically checks the integrity of local authentication database of each device. If the database is correct, no action is taken.
  - If the information has been violated in the network device, it is marked as invalid in the database, a new set of username/password for the network device is created, and after that, all the data is updated in the local authentication database of the device.
- Upon receiving the request of a local username/password for a certain device, the framework will inform the credential to the operator, mark the credential entry as invalid in the database, and create a new credential to that device in the database.

All communication between the framework and the network devices must be performed in a secure manner. Hence, the use of protocols such as Telnet is not allowed.

4. Implementation and validation scenario

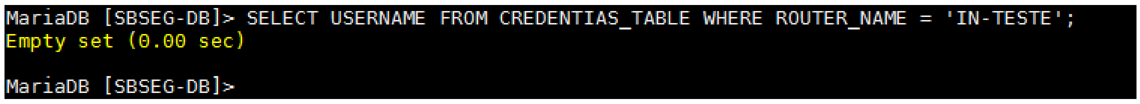
To implement this application we use a Intel server with a Linux CentOs with the Python3 language installed, which is the language used for the development of the framework. Several languages could be used, but Python presents great acceptance and availability of modules for accessing network equipment. Python also provides database support, which is a requirement for the framework.In this server we have one active MariaDB database, which is used as database for the entire project.

For accessing network devices we used the Netmiko library. Technically we can use any DevOps tools that support automation in network devices, for instance, ansible. However, as the tests and verifications are performed by equipment as well as the set of username/password is individual by elements. We opted for the use of a standard and widely used library. And for access to the database the default mysql connector. Our database access was developed using the Flask framework, which provides secure access. In the test, we use a Cisco Systems router. As all the interactions will be carried out only with sshv2 accesses and how all the network elements implement hashes to store the passwords. Our proposal is to periodically validate all passwords by performing a simple match operation between the hash and the hash present in the network element.

5. Results

Our experiment objective is to validate the software operation. Hence, we go step-by-step in the described flows in a real environment.

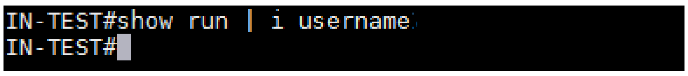
Our first test is to check if our application is capable to create a new set of username and password to one router, which we call ‘IN-TESTE’. To check it, we will use a command line in python in order to follow the process of creating the username and password on both router and MariaDb. Figure 5 shows that initially there is no authentication data on both database and router.



```
MariaDB [SBSEG-DB]> SELECT USERNAME FROM CREDENTIAS_TABLE WHERE ROUTER_NAME = 'IN-TESTE';
Empty set (0.00 sec)

MariaDB [SBSEG-DB]>
```

(a) Empty table for network devices.

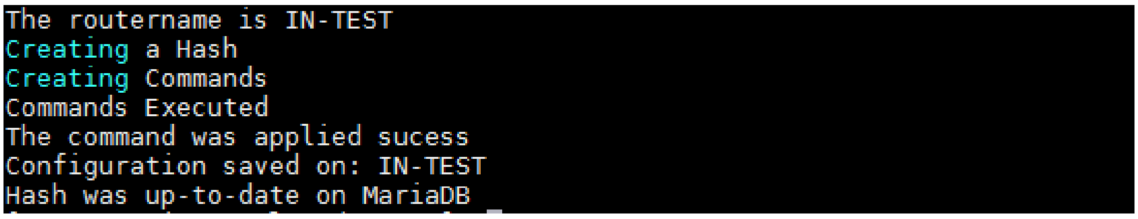


```
IN-TEST#show run | i username
IN-TEST#
```

(b) No users or passwords in network devices.

Figure 5. Initial state of database and device.

Next, we started our application and waited for the initial deploy of a credential set on all stored network devices, in our case, a single router. Figure 6 shows our application running while creating the initial credential set. The result is presented in Figure 7. As a result, we have created a username and a password not only on the router but also on the Credential Database.

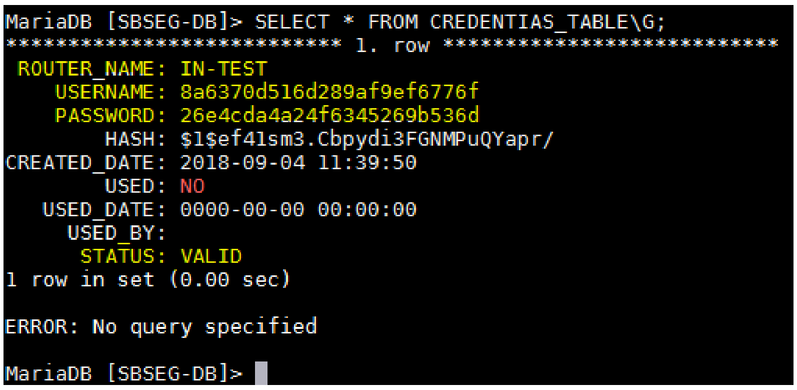


```
The routername is IN-TEST
Creating a Hash
Creating Commands
Commands Executed
The command was applied sucess
Configuration saved on: IN-TEST
Hash was up-to-date on MariaDB
```

Figure 6. Developed application creating and installing a credential for ‘IN-TEST’ router.

With this test, we were able to demonstrate that our application is able to generate a random set of username and password for a certain network device, as well as to install the same in both the internal database and the network device.

The next test is related to the check of local authentication database integrity. Hence, we will show the verification if a credential set was violated in a network device

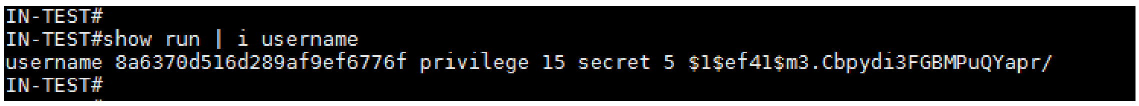


```
MariaDB [SBSEG-DB]> SELECT * FROM CREDENTIAS_TABLE\G;
*************************** 1. row ***************************
 ROUTER_NAME: IN-TEST
    USERNAME: 8a6370d516d289af9ef6776f
    PASSWORD: 26e4cda4a24f6345269b536d
        HASH: $1$ef41sm3.Cbpydi3FGNMPuQYapr/
CREATED_DATE: 2018-09-04 11:39:50
        USED: NO
   USED_DATE: 0000-00-00 00:00:00
     USED_BY:
      STATUS: VALID
1 row in set (0.00 sec)

ERROR: No query specified

MariaDB [SBSEG-DB]>
```

(a) Username and password created on MariaDB.



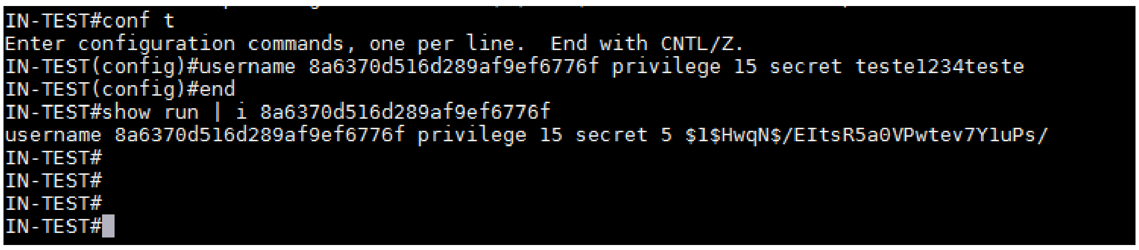
```
IN-TEST#
IN-TEST#show run | i username
username 8a6370d516d289af9ef6776f privilege 15 secret 5 $1$ef41$m3.Cbpydi3FGBMPuQYapr/
IN-TEST#
```

(b) Username and password created on the router.

Figure 7. Initial state of database and device.

using the developed application. Our developed application checks on the Credential Database the username and password plus the hash generated by the network device and compare this data to the information collected from the network device. Our application accesses the device using a credential valid in the central authentication server to perform this action.

Our test consists of modifying the password of the local user and verifying whether the developed application will notice this modification and will initiate the necessary procedures for access restoration, which consists on the removal of the violated user and creation of a new set of credentials for the concerned network device. Figure 8 shows the modification on the local database of ‘IN-TEST’ router.

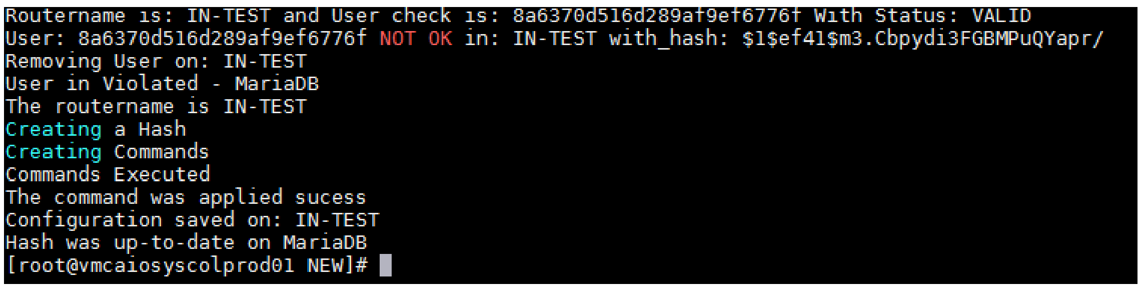


```
IN-TEST#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
IN-TEST(config)#username 8a6370d516d289af9ef6776f privilege 15 secret teste1234teste
IN-TEST(config)#end
IN-TEST#show run | i 8a6370d516d289af9ef6776f
username 8a6370d516d289af9ef6776f privilege 15 secret 5 $1$HwqN$/EItsR5a0VPwtev7Y1uPs/
IN-TEST#
IN-TEST#
IN-TEST#
IN-TEST#
```

Figure 8. Accessing the router to change local user accounts.

After this step, on the next scheduled verification, our application was able to observe the modification, as shown in Figure 9. After that, the application reacts to the violation updating the database. After that, the scheduled update changes the information on the router, as we can see in Figure 10

Our last test is checking if our application can provide access via web browse to username and password that resides on internal database. The first step of this test is access the URL for authentication, as shown on Figure 11. Then, after requesting the credential, the network operator receives the message on Figure 12.



```
Routername is: IN-TEST and User check is: 8a6370d516d289af9ef6776f With Status: VALID
User: 8a6370d516d289af9ef6776f NOT OK in: IN-TEST with_hash: $1$ef41$m3.Cbpydi3FGBMPuQYapr/
Removing User on: IN-TEST
User in Violated - MariaDB
The routername is IN-TEST
Creating a Hash
Creating Commands
Commands Executed
The command was applied sucess
Configuration saved on: IN-TEST
Hash was up-to-date on MariaDB
[root@vmcaiosyscolprod01 NEW]#
```

Figure 9. The reaction of application face the change of password on Network equipment

6. Conclusions

Local database management on network devices is a challenge for service providers with a huge number of devices spread over different areas. We proposed a framework for securing access to this devices when there are no connection to central authentication servers due to failures on the infrastructure or network links.

We implemented and validated our proposal in a small testing environment using real network equipment. As the tests demonstrated, the tool proved to be effective in generating the usernames and passwords, as well as in the provision of such information in the network device and the internal information database for each element involved in the test. In addition to providing and provisioning such information we also found that the tool succeeded in understanding changes that could generate inconsistencies and security breaches in the use of these credentials. These results motivate us to extend the scope of testing to other equipment and wider network areas.

6.1. Next Steps

During the tests we realized non-emergency needs of new functions that would help us during the operation of the network. These functions would be:

1. Allocation of the web access logs to the tool.
2. Generation of a dashboard to accompany the generation, provisioning and use of username and password sets.

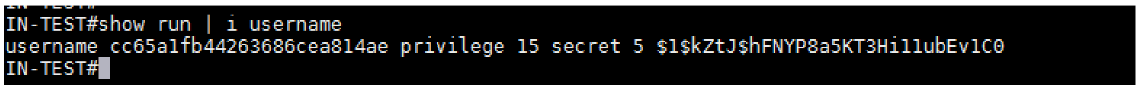
Other future works include the insertion of policies related to users and areas of actuation when releasing a username and password for local access. Besides, further work can be applied in correlating local logs of the devices with the authentication logs generated using our framework, which would help to identify the actions of each user in case of failures or attacks.

References

[Finseth 1993] Finseth, C. (1993). An Access Control Protocol, Sometimes Called TACACS. RFC 1492, IETF.

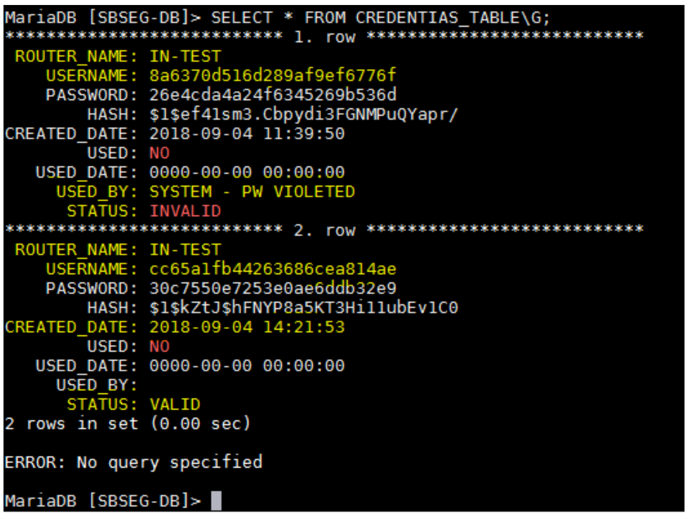
[Hildebrandt and Saake 1998] Hildebrandt, E. and Saake, G. (1998). User authentication in multidatabase systems. In *Proceedings Ninth International Workshop on Database and Expert Systems Applications (Cat. No.98EX130)*, pages 281–286.

[Kemper 2001] Kemper, S. (2001). Computer security with local and remote authentication. Technical Report US Grant US7222361B2, Hewlett-Packard Development Co LP.



```
IN-TEST#show run | i username
username cc65a1fb44263686cea814ae privilege 15 secret 5 $1$kZtJ$hFNYP8a5KT3Hi1lubEv1C0
IN-TEST#
```

(a) New username and password in the router created by our application due to the unauthorized change of password on network equipment.



```
MariaDB [SBSEG-DB]> SELECT * FROM CREDENTIAS_TABLE\G;
*************************** 1. row ***************************
 ROUTER_NAME: IN-TEST
    USERNAME: 8a6370d516d289af9ef6776f
    PASSWORD: 26e4cda4a24f6345269b536d
        HASH: $1$ef41sm3.Cbpydi3FGNMPuQYapr/
CREATED_DATE: 2018-09-04 11:39:50
        USED: NO
   USED_DATE: 0000-00-00 00:00:00
     USED_BY: SYSTEM - PW VIOLETED
      STATUS: INVALID
*************************** 2. row ***************************
 ROUTER_NAME: IN-TEST
    USERNAME: cc65a1fb44263686cea814ae
    PASSWORD: 30c7550e7253e0ae6ddb32e9
        HASH: $1$kZtJ$hFNYP8a5KT3Hi1lubEv1C0
CREATED_DATE: 2018-09-04 14:21:53
        USED: NO
   USED_DATE: 0000-00-00 00:00:00
     USED_BY:
      STATUS: VALID
2 rows in set (0.00 sec)

ERROR: No query specified

MariaDB [SBSEG-DB]>
```

(b) Information updated on database with new username and password created by our application and the new status of old user and password sets.

Figure 10. Reaction to a violation on a local authentication database of a monitored network device.

[Liang and Wang 2004] Liang, W. and Wang, W. (2004). A local authentication control scheme based on aaa architecture in wireless networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, volume 7, pages 5276–5280 Vol. 7.

[Me et al. 2005] Me, G., Strangio, M. A., and Dellutri, F. (2005). Local authentication with bluetooth enabled mobile devices. In *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services - (icas-isns'05)(ICAS-ICNS)*, page 72.

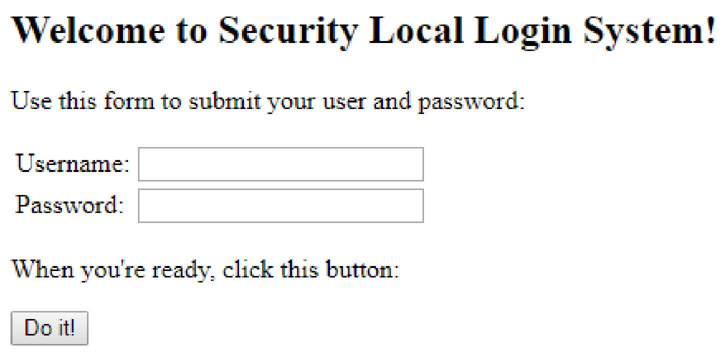
[Me et al. 2018] Me, G., Strangio, M. A., and Dellutri, F. (2018). Hardening cisco devices based on cryptography and security protocols - part one: Background theory. In *Annals of Emerging Technologies in Computing (AETiC)*, page 18.

[Rigney 2000] Rigney, C. (2000). Remote Authentication Dial In User Service (RADIUS). RFC 2865, IETF.

[Silva et al. 2014] Silva, E. F., Fernandes, N. C., Rodriguez, N., and Muchaluat-Saade, D. C. (2014). Credential translations in future internet testbeds federation. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–6.

[Silva et al. 2018] Silva, E. F., Muchaluat-Saade, D. C., and Fernandes, N. C. (2018). Across: A generic framework for attribute-based access control with distributed policies for virtual organizations. *Future Generation Computer Systems*, 78:1 – 17.

[Steiner et al. 1988] Steiner, J. G., Neuman, C., and Schiller, J. I. (1988). Kerberos: An authentication service for open network systems. In *IN USENIX CONFERENCE PROCEEDINGS*, pages 191–202.



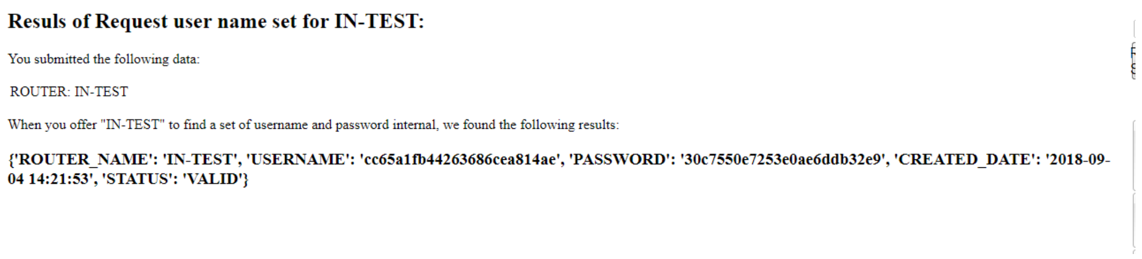
Welcome to Security Local Login System!

Use this form to submit your user and password:

Username:	
Password:	

When you're ready, click this button:

Figure 11. The login page for obtaining a valid credential for a local access on ‘IN-TEST’ router.



Resuls of Request user name set for IN-TEST:

You submitted the following data:

ROUTER: IN-TEST

When you offer "IN-TEST" to find a set of username and password internal, we found the following results:

```
{'ROUTER_NAME': 'IN-TEST', 'USERNAME': 'cc65a1fb44263686cea814ae', 'PASSWORD': '30c7550e7253e0ae6ddb32e9', 'CREATED_DATE': '2018-09-
04 14:21:53', 'STATUS': 'VALID'}
```

Figure 12. The web page with response (username/password) requested by network operator.