GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

An Architecture for Implementing Enterprise Multifactor Authentication with Open Source Tools

GIAC (GSEC) Gold Certification

Author: Student Tom Webb, Tcw3bb@gmail.com
Advisor: Kees Leune

Accepted: December 5th, 2013

Abstract

Credential stealing has become an epidemic. Organizations need an effective and cost efficient way to reduce the likelihood of being exploited thought this vulnerability. Leveraging Google Authenticator, RADIUS and a REST API will allow enterprises to enroll users into their existing service portal and integrate services easily with this well supported protocol.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

1. Multifactor background

We are all familiar with how password authentication works as we log into dozens of systems each day to check email or view bank account balance. This type of authentication is considered single factor authentication. Authentication can happen using something you know, something you have, something you are or somewhere you are (Bishop, 2004). Multifactor combines two or more of these methods to create a stronger authentication.

Traditionally, multifactor authentication has been expensive to deploy, based on the cost of buying equipment and the time it took to enroll individuals into the systems. The use of smart phones in multifactor authentication has lowered the cost of deployment significantly. This is due to savings from not requiring hardware tokens and the use of open source software tokens on the phones.

Time based one-time passwords (TOTP) is the method this paper focuses on. They are considered the *something you have* portion of authentication methods. There are several TOTP products, of which RSA SecureID is the most well-known. These products use various methods to generate a number specific to the individual user that expires after a predefined time.

Biometrics is based on something you are. Two types of biometrics we find being used on consumer devices are fingerprint scanners on laptops and facial recognition used on Android phones. Deployment of biometrics to a large organization is difficult due to every computing device must have a reader and enrollment of individuals into an enterprise solution is time consuming. While the new iPhone 5s is one of the largest consumer devices that have this integrated, there is a reported 20 percent authentication failure rate (Kosner, 2013). Collecting this data also has privacy concerns (Prabhakar, Pankanti, & Jain, 2003) which could slow individual's adoption of the technology.

1.1. What risk does it reduce?

Multifactor authentication prevents attackers from gaining access to systems if they are able to steal the user's passwords. Collecting passwords is very easy for

© 2013 The SANS Institute Author retains full rights.

advanced attackers and sixty percent of people are likely to fall for a well crafted phishing email (Sixty percent will fall to a phishing attack that might herald an APT, 2013). Adding another piece of information the attacker needs to access the systems makes the attack more complicated.

In the NIST Special Publication 800-63-1 pg. 45, many types of threats and mitigation strategies are covered in section 6.2 (National Institute of Standards and Techonology, 2011). Two of the more common threats are theft and revealing the token using social engineering. To reduce the effectiveness of theft, you can use a token that requires a pin before presenting the OTP. For a social engineering attack, making sure that token entropy is high enough to prevent guessing future values based on a single known one.

1.2. Google Authenticator OTP

There are two main type of OTP supported by Google Authenticator. One based on time (TOTP) discussed in RFC 6238 (M'Raihi, Machani, Pei, & Rydell, 2011) and one based on an event HMAC one time password (HOTP) discussed in RFC 4226 (M'Raihi, Bellare, Hoornaert, Naccache, & Ranen, 2005).

Time-based tokens are the most common, but you can also implement HOTP with this configuration. Changing what type of token you generate during the initial enrollment process will allow HOTP.

2. Architecture

The system design described in this paper uses Redhat EL 6 as a starting point, but can be easily ported to CentOS for individuals who do not use Redhat. Kerberos is used to authenticate to Microsoft Active Directory (AD) and FreeRADIUS will use the Pluggable Authentication Module (PAM) for Googles TOTP. Authorization in this design should occur at the application level due to this system authenticating a large number of different services. If you want to do authorization from RADIUS, get more information on LDAP authorization at their website[1].

[1] http://wiki.freeradius.org/modules/rlm_ldap

© 2013 The SANS Institute Author retains full rights.

To setup a system or service to use multifactor, in this design, it must support RADIUS protocol for authentication. Each service that you wish to add should have a separate RADIUS secret key to prevent others from reading authentication traffic. The RADIUS shared secret is susceptible to offline dictionary attacks (Aboba, 2005). Each key length should be a minimum length of 25 characters to make brute forcing the secret a less attractive option for attackers. If authentication is occurring over untrusted networks, use IPSEC to provide more robust encryption to the protocol.

Figure 1 displays the process of a SSH service setup to use RADIUS as authentication. When the user enters his credentials, the SSH service passes the information to the RADIUS server.

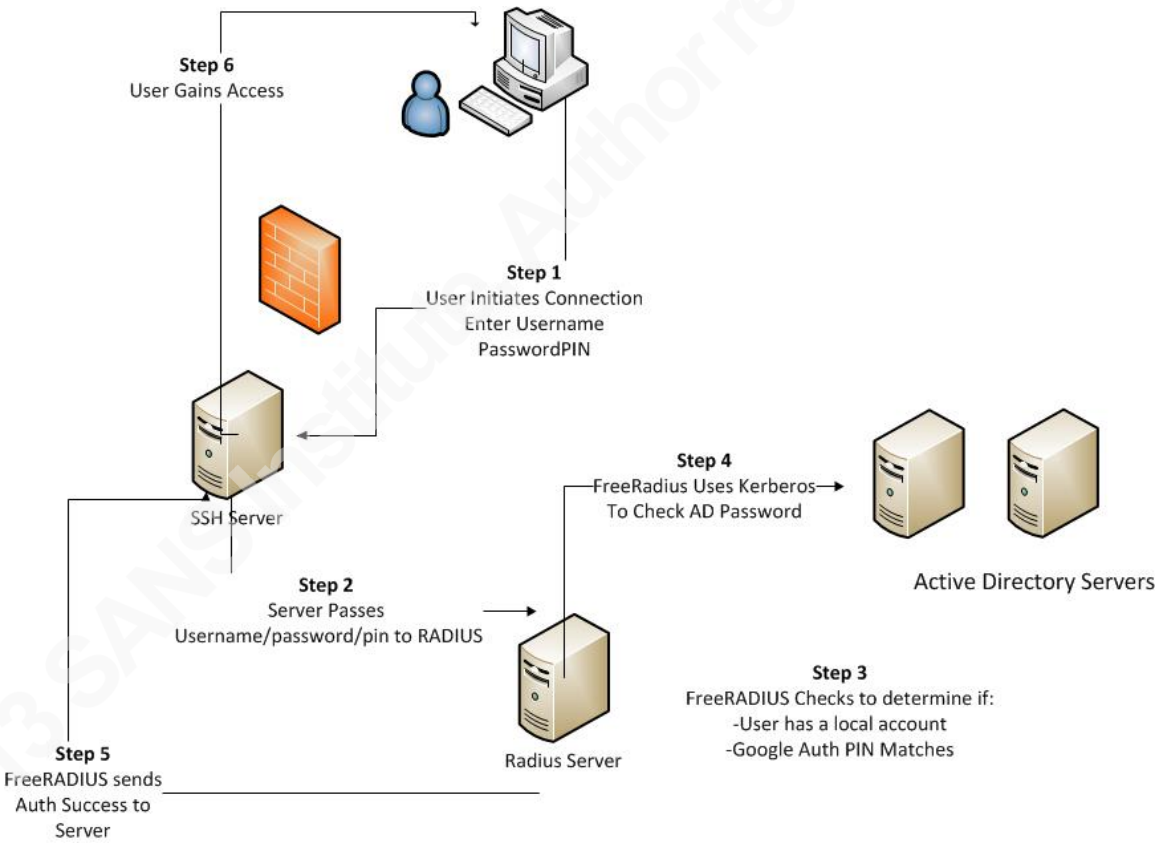


Figure 1. Radius Authentication using Kerberos and Google Authenticator.

When authenticating, users will need to combine both their AD credential and TOTP password into one word. If your password is bob and your token is 987654 then, when prompted, you will "smash" them together (e.g. bob987654). Appendix A covers the must have security considerations when implementing this architecture.

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.

3. Enrollment

Most large organizations already have a portal for their users to perform self-service functions. A web API will be used to allow self-enrollment into the system.

When the user initiates the enrollment, a local user is created on the RADIUS system. The user must have a local account, as this is a way to track enrollment and create the secret key for creating the OTP. Once the local user is created, a secret seed key is created in `/var/Google-auth/userid`. The secret key will be sent to the web service that requested the creation via a QRCode. This code should be strictly protected. If attacker gets access to this code, they will be able to create OTP for that user. The attacker will still need to have the users AD credential, but they will have a critical piece of data to perform an attack.

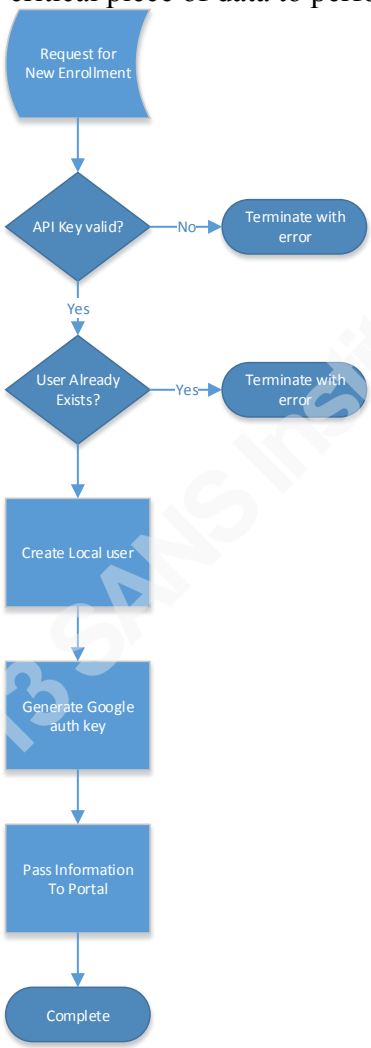


Figure 2. Logic of Enrollment API.

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.

4. Disenrollment

To remove a user from the multifactor system, you should delete their local user along with the OTP secret seed. If you disable the users in Active Directory, they will no longer be able to authenticate using the RADIUS server. To make sure that users do not linger, this system should also be part of your HR off-boarding process.

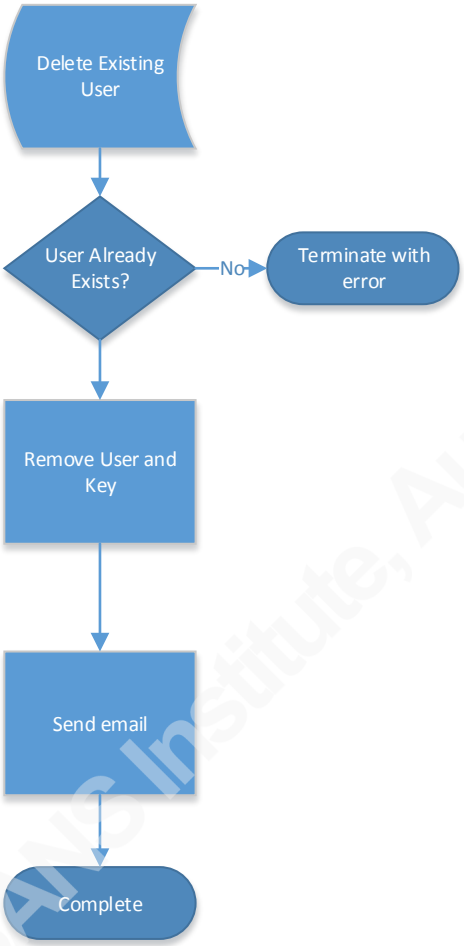


Figure 3. Logic of disenrollment process.

5. Configuration

Software Installation

```
>sudo yum install ntp  pam_krb5  krb5-workstation
freeradius-utils.x86_64 make gcc bzip2 gcc++ pam-devel
libpng libpng-devel freeradius
```

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.

Install the latest software for QRcodes from the libqrencode web site2 by downloading and compiling it. This is used to create the QRcode for the secret seed key during enrollment process to prevent users from mistyping the key.

```
>tar -zxvf qrencode
>cd qrencode*
>./configure
>make;make install
```

5.1. NTP

When using TOTP, having time synchronization on the server and your token device is critical. The Network Time Protocol (NTP) is used to keep time synchronized. Have it run at startup using the following commands:

```
> chkconfig ntpd on
```

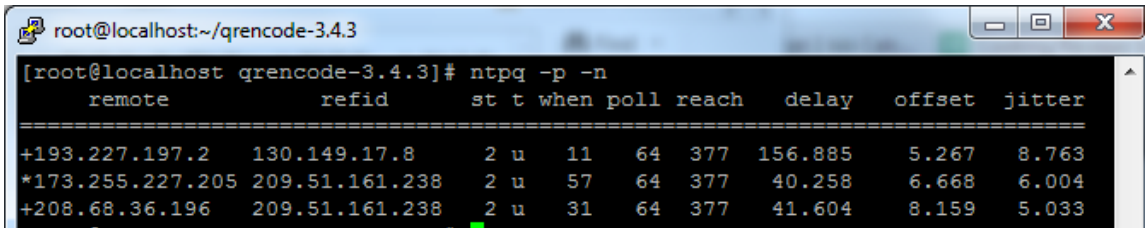
On the server, you will need to edit the `/etc/ntp.conf` and add your primary time server. Comment any default servers, unless these are your primary time servers. The file format is "server <ip address>".

```
server  1.2.3.4
```

Next, you should start the NTP daemon and check to make sure the time sync is working appropriately.

```
>sudo service ntpd start
>ntpq -p -n
```

The `ntpq` results should resemble the output in figure 4. The most important columns to check for are the last time it polled and offset. These numbers are in milliseconds, but the values should not be very high.



[2] http://fukuchi.org/works/qrencode

© 2013 The SANS Institute Author retains full rights.

Figure 4. Typical ntpq results.

5.2. Kerberos

Using Kerberos to authenticate to Active Directory is easy. By following the basic steps on the Indiana University site (In Red Hat Enterprise Linux, how do I authenticate to ADS.IU.EDU using Kerberos?, 2013) we will configure our systems.

```
> /usr/sbin/authconfig-tui
* Use Kerberos
```

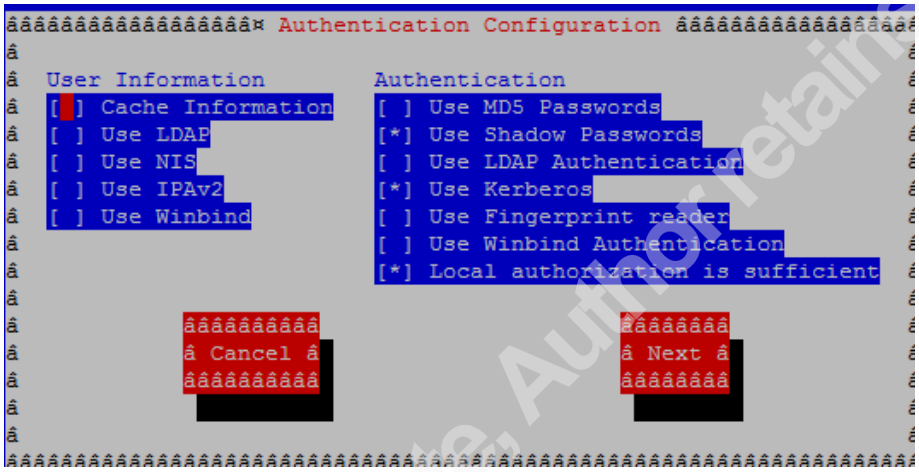


Figure 5. Authconfig-tui interface.

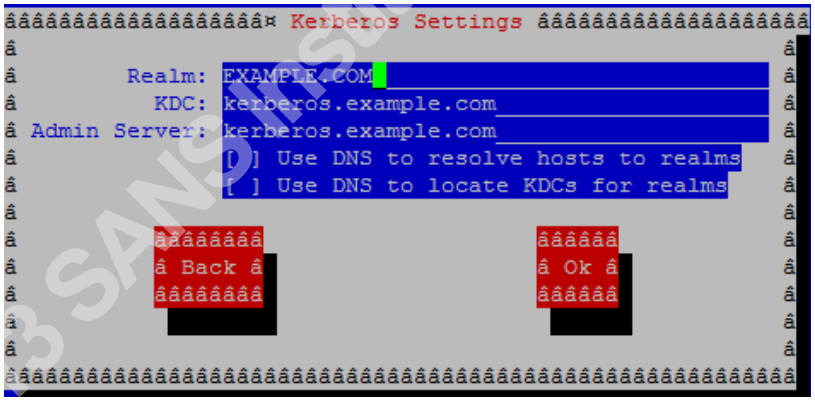


Figure 6. Kerberos Configuration page from Authconfig-tui.

In this configuration, we are not going to use LDAP for user information. Users will be added to this box to confirm they have enrolled in the multifactor system. Fill in the appropriate setting for Active Directory environment as seen in Figure 3.

© 2013 The SANS Institute Author retains full rights.

5.3. PAM

With the PAM RADIUS module, we are setting up how PAM should authenticate anyone who uses this service. Setup each user with his own token in a custom location for Google Authenticator to store the keys. In this case, place the keys into `/var/Google-auth` folder and we name the file with the user name. With the configuration below, both the Google Authenticator and Kerberos password to be correct before access is granted by RADIUS.

```
/etc/pam.d/radiusd
# Use the right 6 digits for google-authenticator
(forward_pass)
auth requisite pam_google_authenticator.so user=root
secret=/var/Google-auth/${USER}_google_auth forward_pass
auth        required    pam_krb5.so use_first_pass
account     sufficient    pam_localuser.so
```

5.4. Google Authenticator

Download the latest Google authenticator PAM module at its home page.[3] Follow the instructions from the README file, but should be your traditional gcc compile instructions. Once you are done compiling, remove gcc. This will make it more difficult for attackers, who gain access to your system, to install additional tools.

```
>bzip2 -d libpam-google-authenticator-<VERSION>-
source.tar.bz2;tar -zvf libpam-google-authenticator-
<version>-source.tar.bz2
>cd libpam*
>make install
>sudo yum erase gcc pam-devel
```

5.5. FreeRADIUS

FreeRADIUS has been around for a while and it has many features. These instructions cover version 2.2 for installation. FreeRADIUS has released version 3.0 but

[3] https://code.Google.com/p/Google-authenticator/downloads/list.

© 2013 The SANS Institute Author retains full rights.

is not considered stable at this time. One of the main security features it now supports is RadSec or TLS encryption.

Unfortunately, due to FreeRADIUS needing PAM, it has to run as the root user. To help reduce the attack profile, SELinux should be enabled on your system, which is default. Additionally, limit access to UDP port 1812 to only system that will be authenticating to the system. Make the following changes to the RADIUS configuration file below.

```
>vi /etc/raddb/radius.conf

user=root
group=radiusd

destination = syslog
       stripped_names = yes
 auth = yes
```

Setup the default auth method for RADIUS to PAM. Add the settings below to the `/etc/raddb/users` file. Additionally, in the file `/etc/raddb/sites-enabled/default`, enable PAM under the authenticate section.

```
>vi /etc/raddb/users
DEFAULT Auth-Type := PAM

>vi /etc/raddb/sites-enabled/default
pam #Remove the # sign in front
```

Finally, make sure that the appropriate permissions are set on the folders to prevent others from accessing them. The most critical folder is the Google auth secret seed codes, makes sure no one can access this folder to prevent someone from duplicating a key and impersonating an authorized user

```
> chown radius:radiusd /var/google-auth/
> chmod 400 /var/Google-auth/
> chcon -v --type=radiusd_t /var/google-auth/
```


© 2013 The SANS Institute Author retains full rights.

5.6. Enrollment API

When designing the enrollment code, it's critical to keep security in mind. This process will create the secret key and send the QRcode to your portal. If attackers gain access to this system, they could steal a key or even create a key for themselves.

As this is a simple service for creating secret keys, a REST API will be great for this. According to Rodriguez, REST APIs use HTTP methods explicitly, are stateless and expose directory structure-like URIs (2008). In cases where information is read from the API, a get request with the desired query should be used. A post request will create the users.

All traffic must be encrypted between the portal and enrollment API. This is a web service, and SSL is the best way to do this. Any system that will request a user enrollment should have its own unique API. This key acts as additional authentication for the request and helps keep track of different servers that are using the API. Access to the API should be limited via local or network firewall to only allow your portal IP's to access it.

To prevent injection into the API, we need to make sure we create a whitelist of approved input values and encode responses output to stop other possible attack vectors. The API should keep detailed log information about all requests made. The web service making these requests should have limited permissions to prevent the possibilities of being used to escalate privileges. An example API will be made available at my github site.[4].

6. Service Integration

Integration with systems can be complicated and cumbersome. The key to rolling out multifactor is to determine what system, based on your data classification, should be required to have strong authentication. By rolling out per service, you can target communications and have a methodical approach to help make the deployment a success.

[4] https://github.com/tcw3bb/google-auth-api

© 2013 The SANS Institute Author retains full rights.

6.1. VPN

Many mature products support RADIUS as an authentication method. One of the most common items to implement multifactor authentication is VPN. Juniper SSL, Cisco ASA and OpenVPN all support this method. Authorization is critical to limit only approved individuals to use the VPN. Remote access and Email are the most targeted services for stolen credentials. Multifactor deployment for these services is the most common for this reason.

6.2. Web

Integrating Apache directly with RADIUS is easy using `mod_auth_radius`. To protect specific web directories with multifactor add them to the `apache.conf` file. This is an easy way to retrofit a website without modifying the code at all. The FreeRADIUS website[5].has a great article on proper setup. You can setup the user login form to directly integrate with your RADIUS. However, a better architecture is setting up an enterprise web single sign on (SSO) using Shibboleth or another CAS system.

6.3. SSH

SSH configuration is done through the PAM `sshd` plugin. With this setup you will allow multifactor auth for remote connections, but local authentication will still use the local user's password. You may use your AD credentials for `sudo` or require and additional TOTP, but this is may be covered in a later paper.

Test connectivity

Before you setup SSH for RADIUS, you want to make sure that the server can communicate to RADIUS to prevent being locked out of the server. Additionally, keep an additional SSH session open to the server while testing to limit the chance of loss of access to the system. You should see traffic on the RADIUS servers `tcpdump` output. If you do not, then you have a local or network firewall issue that needs to be resolved before moving on.

[5] http://freeradius.org/mod_auth_radius/

© 2013 The SANS Institute Author retains full rights.

From the RADIUS Server

```
>sudo tcpdump -nnvi eth0 host <New Server IP> and not port
22
```

From the new SSH Server

```
>nc -u <radius IP> 1812
```

On the RADIUS SERVER

Once connectivity has been confirmed, setup the client to authenticate to the server. Make sure to setup a unique secret for each client for the maximum security. This will prevent someone who has brute forced one key from sniffing all passwords being used for authentications.

```
>vi /etc/raddb/clients.conf

client NAME {
 ipaddr = <SSH Server IP>
 secret = SECRET
}
```

On the SSH Server

Download the latest version of PAM RADIUS module from the FreeRadius site[6]. You will also need to install a couple additional packages to compile the software.

```
>yum install pam-devel
>tar -zxvf pam*; cd pam*;./configure;make
>sudo make install
```

To setup the new authentication against the RADIUS server, edit the `pam_radius_auth.conf` file and setup the shared secret and IP of the system. The file format is IP, shared secret and timeout.

```
>sudo vi /etc/pam_radius_auth.conf

#Place a # in front of the line with 127.0.0.1
1.1.1.1 shared_secret 3
```

[6] http://freeradius.org/pam_radius_auth

© 2013 The SANS Institute Author retains full rights.

The last piece is to setup PAM to use the RADIUS module for authentication. We are going to require RADIUS to log into the system. Place the change at the top of the file and restart the SSH deamon.

```
>sudo vi /etc/pam.d/sshd
auth        required     /lib/security/pam_radius_auth.so #place at top
of file

>sudo service ssh restart
```

7. Conclusion

While the technology seems to be the most cost prohibited portion of multifactor authentication, user awareness and education on enrollment and preparing your helpdesk on how to prepare for the onslaught of calls during the initial rollout is critical to the success of the deployment.

Targeting user's credentials is the easiest way to gain access to systems and attackers are going to continue to exploit this vulnerability until another attack vector becomes more effective. While using some custom code for enrollment and leveraging smart phones will allow for a low cost alternative for multifactor authentication.

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.

8. References

Aboba, B. (2005). RADIUS Security Issues.

Bishop, M. (2004). Introduction to Computer Security. In *Authenication.* Boston, MA: Addison-Wesley Professional.

In Red Hat Enterprise Linux, how do I authenticate to ADS.IU.EDU using Kerberos? (2013, June 19). Retrieved from Indiana University Information Technology Services: http://kb.iu.edu/data/akoo.html

Kosner, A. W. (2013, 10 15). *iPhone 5S Touch ID Fingerprint Scanner Is A Fail For 20% Of Users, Here's What To Do.* Retrieved from Forbes: http://www.forbes.com/sites/anthonykosner/2013/10/15/iphone-5s-touch-id-fingerprint-scanner-is-a-fail-for-20-of-users-heres-what-to-do/

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005, December). RFC 4226. Retrieved from https://tools.ietf.org/html/rfc4226

M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). RFC 6238. Retrieved from https://tools.ietf.org/html/rfc6238

National Institute of Standards and Techonology. (2011). Electronic Authentication Guidelines.

Prabhakar, S., Pankanti, S., & Jain, A. (2003). Biometric Recognition:Security and Privacy Concerns. *IEEE SECURITY & PRIVACY*, 33-42.

Rodriguez, A. (2008, November 6). *RESTful Web services: The basics.* Retrieved from IBM: http://www.ibm.com/developerworks/webservices/library/ws-restful/

Sixty percent will fall to a phishing attack that might herald an APT. (2013, January 15). Retrieved from Infosecurity-magazine: http://www.infosecurity-magazine.com/view/30220/sixty-percent-will-fall-to-a-phishing-attack-that-might-herald-an-apt/

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.

Appendix A (Checklist)

REST API

- SSL
- Logging
- Limited service permissions
- Input Validation
- Encode responses
- Pass secret or QRcode to portal.
- API Key per request system

Radius Security

- SELinux
- Kerberos AD authentication
- IPsec across untrusted networks
- NTP
- Syslog to another system
- File permissions on the secret key folder
- IPTables for only allowed hosts to make API requests

Author Name, email@address

© 2013 The SANS Institute Author retains full rights.