CTSC Recommended Security Practices for Thrift Clients: Case Study - Evernote

May 21, 2014

For Public Distribution

Randy Heiland, Suresh Marru, Marlon Pierce, Von Welch

Introduction

The Science Gateway Platform (SciGaP, scigap.org) will provide services to help communities create Science Gateways. SciGaP (via Apache Airavata) will use the Apache Thrift framework (thrift.apache.org), a language-independent, richly typed interface definition language (IDL) to generate both client and server software development kits (SDKs). Thrift takes a departure from many public services in that it is not a RESTful (http://en.wikipedia.org/wiki/Representational_state_transfer) API.

To gain a better understanding of Thrift (for the CTSC-SciGaP engagement), we examine an existing application/service that uses it: Evernote (evernote.com). Hopefully, the design and use cases of Evernote will help inform the design and use cases of SciGaP, at least from a security perspective. This document provides an overview of Evernote with an emphasis on its Cloud API, some examples of its SDKs, and a list of recommended practices for using Evernote.

Evernote is a cloud-based note taking, archiving, syncing, and sharing service. A “note” might consist of text, an image, or an audio memo. Notes are organized into “notebooks”. Notebooks are of two flavors: *cloud* (a.k.a. *synchronized*) notebooks and *local* notebooks, which should be self-explanatory.

Evernote is a “Freemium” service - Free for basic functionality; Premium for additional functionality (currently $5/mo; $45/yr). In addition to just storing and sharing notes, Evernote provides search functionality on notes and metadata. This includes some optical character recognition (OCR) functionality for hand-written notes and images.

Evernote provides applications for many clients, e.g. web browser, desktop, tablet, and phone (Figure 1). It also openly exposes its “Cloud API” (dev.evernote.com/doc/reference). More importantly, it provides free SDKs (software development kits, dev.evernote.com/doc/) for multiple programming languages to access its Cloud API. The SDKs make it quite easy for 3rd party developers to provide a variety of apps (appcenter.evernote.com). Evernote has a large user base, apparently having reached 100 million in 2014.

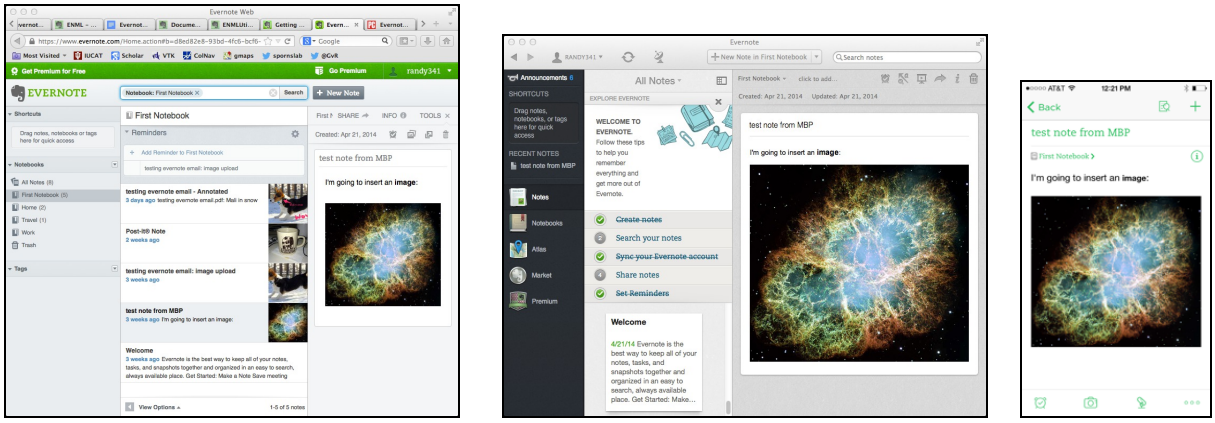

Figure 1. Evernote clients for browser, desktop, and phone

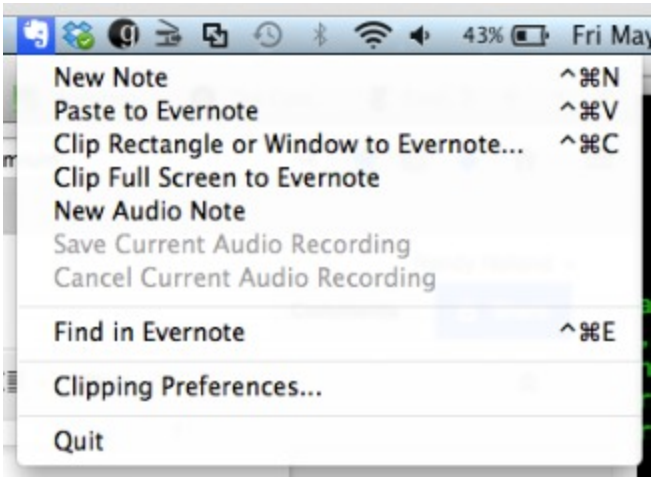

Figure 2. Evernote desktop application menu

Glossary

- IDL - Interface Definition Language. A high-level specification language for defining the interface between software components that need to communicate. From the IDL, client and server implementation stubs can be generated for multiple programming languages.
- ENML - Evernote Markup Language - a superset of XHTML used to represent the content of notes. (dev.evernote.com/doc/articles/enml.php)
- EDAM - Evernote Data Access and Management - a protocol for exchanging Evernote data with an Evernote service. The data structures and remote procedures supported by EDAM are expressed using the Thrift IDL.
- http://dev.evernote.com/support/glossary.php

Evernote Cloud API

We are primarily interested in how “thick” clients, i.e. clients other than a web browser (“thin”) client, using the Evernote SDKs, access and use the Evernote Cloud API in a secure manner. After gaining a better understanding of the API and showing some examples of simple clients, we will provide a list of recommended security practices. We believe these practices will apply to SciGaP’s thick clients as well.

The Evernote Cloud API provides two logical services: the **UserStore** and the **NoteStore**. The UserStore manages user accounts. The NoteStore manages the content of a user’s account. We illustrate, using a simple Python client script, how one might use the NoteStore service to print out the number of notebooks and the notebooks’ names:

```
from evernote.api.client import EvernoteClient

auth_token = "your secret developer token string"
client = EvernoteClient(token=auth_token, sandbox=True)
note_store = client.get_note_store()
notebooks = note_store.listNotebooks()
print '# of notebooks in store =',len(notebooks)
for nb in notebooks:
  print nb.name
```

Running this client via ‘`python notebook_names.py`’ will generate the following output:

```
# of notebooks in store = 5
First Notebook
CTSC Notebook
Work Receipts
Home Receipts
Fav Images
```

In addition to providing SDKs for multiple languages, Evernote even makes available the Thrift IDL files for its services. Most developers will never use these (given the SDKs), but they can be helpful if it becomes necessary to generate one’s own client code for some reason. The IDL files can be found at:
https://github.com/evernote/evernote-thrift/tree/master/src:
`Types.thrift, Errors.thrift, Limits.thrift, NoteStore.thrift, UserStore.thrift`

Shared Notebooks

It is possible for users to share a notebook - even on the free plan. However, on the free plan, only the notebook owner can edit it. For both/multiple users to be able to edit the notebook, the one sharing the notebook has to be a Premium user.

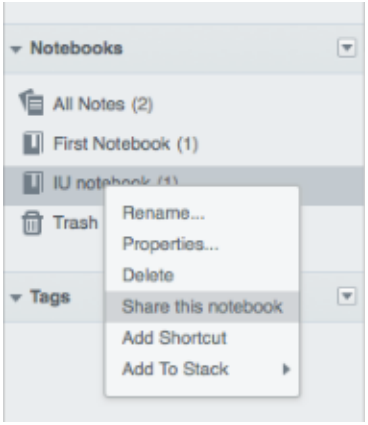

Figure 3. Sharing a notebook from the GUI

From a SDK client, one can discover any notebooks shared with them, e.g.:

```
client = EvernoteClient(token=auth_token, sandbox=True)
note_store = client.get_note_store()
shared_notebooks = note_store.listSharedNotebooks(auth_token)
print 'shared_notebooks=',shared_notebooks
```

example output:

```
shared_notebooks= [SharedNotebook(username='randy', requireLogin=True,
userId=myID, allowPreview=False, notebookModifiable=False, notebookGuid='long
string', shareKey='55a8-s1', email='randy.heiland@gmail.com',
serviceUpdated=1400511058000, recipientSettings=None, privilege=0,
serviceCreated=1400510953000, id=anotherID)]
```

Authentication

Evernote clients provides two options for authentication: tokens and OAuth. In summary:

- Tokens (long strings) are used by **developers** to bypass OAuth and thereby make it easier to develop an app
- OAuth is used by **users**/apps to authenticate to a user's account

- the Evernote-provided SDKs help make OAuth easy

Developer tokens allow one to use the Evernote API to access a personal Evernote account. Upon receiving a token, one will receive an email with the following information:

> Protect this token as carefully as you protect your Evernote password! Anybody with access to this token has full access to your Evernote account. You can revoke this token at any time by returning to this page.
>
> To get started, copy the token below and paste it into your code. It can be used as the authenticationToken parameter in any authenticated Evernote API call. Authenticated NoteStore calls should be made to the NoteStore URL displayed below.
>
> Developer Token (a string consisting of 100 characters)
> S=s1:U=8e640:E=14cdc6-blah:C=14584afc1b7:P=1cd:A=en-devtoken:V=2:H=24df-blah-blah
>
> NoteStore URL:
> https://sandbox.evernote.com/shard/s1/notestore
>
> Expires:
> *(one year from date requested)*

In addition to a developer token, one can also receive a Key/Secret with which to authenticate:

> Thanks for your interest in the Evernote API! This message contains your new API key and some information to help you get started.
>
> Consumer Key: randy-blah
> Consumer Secret: *blah-blah-blah*
> Application Name: simple
>
> Please be sure to request a separate API key for each application that you develop.
>
> Your new key is currently active on our testing sandbox, https://sandbox.evernote.com/, but not on our production service. Because the sandbox is completely isolated from our production environment, you will need to create a new account to do your testing. You can create as many sandbox accounts as you need at https://sandbox.evernote.com/Registration.action, but don't use the sandbox for any critical data, since it's not protected and backed up like our production service.
>
> To use the API key, you'll need to visit our developer site, http://dev.evernote.com/doc/, and download the appropriate API SDK for your platform. The SDKs include our API client code as well as sample code, and the API page also contains links to our API Overview and detailed documentation.
>
> Once you have completed your development, visit http://dev.evernote.com/support/ to request activation of your API key on our production service.

Two-step verification is currently available to Evernote Premium and Evernote Business users. This feature will be available to free users in the future.

Two-step verification is an optional service that adds extra security to your Evernote account; however, it may not be right for everyone. If, for example, you keep sensitive personal documents and information or other data that you wish to keep as secure as possible stored in your account, then we encourage you to take advantage of this feature. If you are a more casual user of Evernote, however, the additional steps involved in accessing your account using two-step verification may not be desirable.

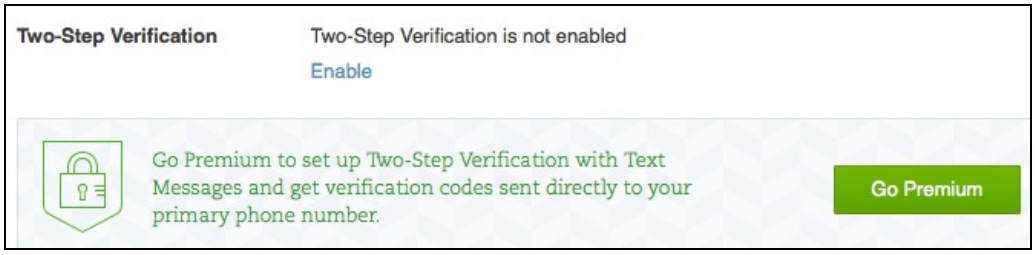

Figure 4. Two-step verification

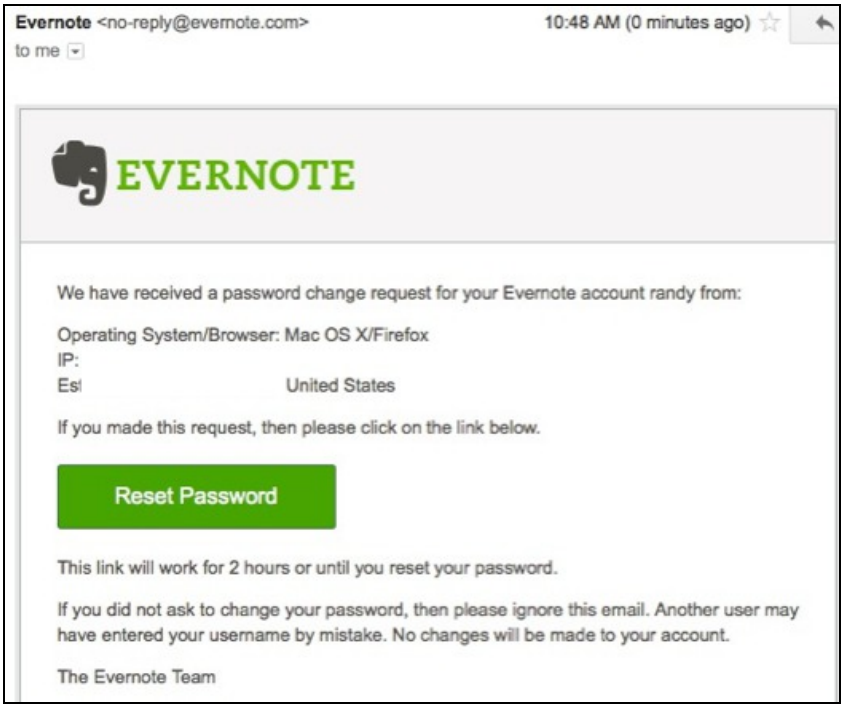

Figure 5. Allowing a ‘click to reset’ option in email is not good practice

Evernote SDKs

Evernote provides SDKs in several languages: github.com/evernote/.

Thrift interface definition files (IDL) files for the Evernote Cloud API can be obtained from: github.com/evernote/evernote-thrift.

Security

Security issues associated with Evernote are typical for any cloud service. Some of these issues include:

- your password will be stored (in some form) on a remote server
- Personally Identifiable Information (PII) may be stored on a remote server
- insecure interfaces and APIs
- information that may be critical or sensitive will be sent over the network
- phishing attacks are more likely for a well-known cloud service
- governments may obtain data stored in the cloud

Recommended Security Practices

Since Evernote controls their servers, we do not offer any recommended practices for their setup and operation. This lack of known practices might, however, result in more strict recommended practices for a client. For example, not knowing the security measures for a server should cause a user to think more seriously about uploading sensitive documents.

Regarding identity management (IdM), there is no choice: Evernote requires the use of OAuth (OAuth 2), so clients will need to use it.

A list of recommended practices related to Evernote, some specifically for developers [dev] and some for users [user], includes:

- [dev] For 3rd party services (apps), the services should be clearly identified and documented. This will help users easily determine whether the service is something they want without downloading and trying it out. The documentation should also include information about security.
- [dev] Client software should be designed and implemented using the best practices of secure coding. For example, a client should be careful to avoid a buffer overflow, e.g.

when accepting user input strings. Another example: sanitize any strings that become commands to a database or operating system.

- [dev,user] Protect your credentials (token and/or key) as you would any password. In fact, Evernote warns (http://dev.evernote.com/appcenter/best_practices.php):

  > It is essential that the user's authentication token is encrypted when stored. While we cannot easily verify implementation of this requirement, integrations found to not be satisfying this requirement will meet with a full review and potential API revocation.

- [user] Use the optional *two-step verification* for additional access security
- [user] Encrypt sensitive information that will be stored in the cloud; optionally, store sensitive information in a *Local* notebook

Further Reading

http://www.pcmag.com/article2/0,2817,2383054,00.asp - review of Evernote
http://evernote.com/evernote/guide/web/#9 - 2-step verification for Evernote users
https://github.com/evernote/evernote-thrift - Thrift IDL files for Evernote's Cloud API
http://dev.evernote.com/appcenter/best_practices.php
https://dev.evernote.com/doc/articles/authentication.php
http://blog.evernote.com/blog/2008/04/15/evernote-privacy-and-security/
http://blog.evernote.com/blog/2011/03/24/evernotes-three-laws-of-data-protection/
http://blog.evernote.com/blog/2013/05/30/evernotes-three-new-security-features/
http://nakedsecurity.sophos.com/2013/03/03/evernote-reset-password/
http://en.wikipedia.org/wiki/Cloud_computing_security

Appendix: Evernote Blog on OAuth

Security enhancements for third party authentication
http://blog.evernote.com/tech/2012/04/24/security-enhancements-for-third-party-authentication/
April 24, 2012 | Posted by Seth Hitchings in API

The security and privacy of our users' data is our top priority. This is reflected in our three laws of data protection, and it's reflected in the way that we design our service and the products that access it. As a result, our users trust us, which is one of the reasons that we've been so successful.

Since we launched the Evernote API in October of 2008, we've allowed third party applications to authenticate to Evernote the same way that our applications do – by collecting a user's Evernote username and password and sending them to our web service. Username and password authentication is easy for developers to implement, but it's not great from a

security perspective. Today, we're making some big changes to improve the security of apps built on our API, starting with a transition from username and password authentication to OAuth.

We are now requiring all new applications to authenticate to the Evernote service using OAuth, a standard authorization protocol used by Google, Twitter, Dropbox and most other major web service providers. We will no longer activate applications on the production Evernote service if they use username and password for authentication. The Evernote service has long supported OAuth, and now we're making it mandatory.

Developers have until November 1, 2012 to modify existing applications that authenticate using username and password. At that time, we will cut off third party access to the UserStore.authenticate function. We will email developers who hold "client" API keys (those that authenticate via username and password) this week to let them know about this change, and again in September if they have not converted their application to OAuth.

Most developers are familiar with OAuth from working with other APIs, but we recognize that properly implementing an OAuth client is more work than simply prompting for the user's Evernote username and password. To make the transition easier, we've taken two steps. First, developers who are simply experimenting with the API or scripting access to their own personal account can obtain a developer token. These tokens allow a developer to access their account through the API without any additional authentication. Developer tokens make it easy to get started learning the Evernote API or automating actions for your own account. To learn more about developer tokens, visit dev.evernote.com.

Second, we've added OAuth functionality to our iOS and Android SDKs, which we've published on GitHub. The new SDK functionality implements the entire OAuth flow and can be plugged into an application by simply copying and pasting a few blocks of code. The SDKs also include sample applications that demonstrate how to use the OAuth functionality. Our SDKs for PHP, Python and Ruby contain sample code showing how to use popular OAuth libraries to authenticate to Evernote. We'll be releasing SDKs and sample code for other platforms and languages over the next few weeks.

Full documentation of Evernote's OAuth provider is available on dev.evernote.com. As usual, our developer relations team is available to answer any questions. If you have trouble implementing OAuth, please let us know. We're here to help.

Appendix: Evernote Blog on Security Breach

Security Notice: Service-wide Password Reset
Posted by Dave Engberg on 02 Mar 2013

http://blog.evernote.com/blog/2013/03/02/security-notice-service-wide-password-reset/

Evernote's Operations & Security team has discovered and blocked suspicious activity on the Evernote network that appears to have been a coordinated attempt to access secure areas of the Evernote Service.

As a precaution to protect your data, we have decided to implement a password reset. Please read below for details and instructions.

In our security investigation, we have found no evidence that any of the content you store in Evernote was accessed, changed or lost. We also have no evidence that any payment information for Evernote Premium or Evernote Business customers was accessed.

The investigation has shown, however, that the individual(s) responsible were able to gain access to Evernote user information, which includes usernames, email addresses associated with Evernote accounts and encrypted passwords. Even though this information was accessed, the passwords stored by Evernote are protected by one-way encryption. (In technical terms, they are hashed and salted.)

While our password encryption measures are robust, we are taking additional steps to ensure that your personal data remains secure. This means that, in an abundance of caution, we are requiring all users to reset their Evernote account passwords. Please create a new password by signing into your account on evernote.com.

After signing in, you will be prompted to enter your new password. Once you have reset your password on evernote.com, you will need to enter this new password in other Evernote apps that you use. We are also releasing updates to several of our apps to make the password change process easier, so please check for updates over the next several hours.

As recent events with other large services have demonstrated, this type of activity is becoming more common. We take our responsibility to keep your data safe very seriously, and we're constantly enhancing the security of our service infrastructure to protect Evernote and your content.
There are also several important steps that you can take to ensure that your data on any site, including Evernote, is secure:

- Avoid using simple passwords based on dictionary words
- Never use the same password on multiple sites or services
- Never click on 'reset password' requests in emails — instead go directly to the service

Appendix: Evernote and OpenSSL

Evernote Service Not Affected By OpenSSL Bug

Posted by Rich Tener on 10 Apr 2014

On Monday of this week, a group of security researchers discovered and publicly disclosed a vulnerability in OpenSSL, a software package that is widely used to secure online communications. They called the bug *Heartbleed*.

Evernote does not use, and has not used, OpenSSL, so we were not vulnerable to this bug. As an Evernote user, you don't need to take any action.

Some of the services that we use, for example, our support ticketing system, do use OpenSSL. These services have all fixed the bug. We do not believe that any sensitive data was accessed. We are actively monitoring the situation and will notify you if we discover anything.

Appendix: API Key

Thanks for your interest in the Evernote API! This message contains your new API key and some information to help you get started.

Consumer Key: randy<blah>
Consumer Secret: ea9<blah>
Application Name: simple

Please be sure to **request a separate API key for each application** that you develop.

Your new key is currently active on our testing sandbox, https://sandbox.evernote.com/, but not on our production service. Because the sandbox is completely isolated from our production environment, you will need to create a new account to do your testing. You can create as many sandbox accounts as you need at https://sandbox.evernote.com/Registration.action, but don't use the sandbox for any critical data, since it's not protected and backed up like our production service.

To use the API key, you'll need to visit our developer site, http://dev.evernote.com/doc/, and download the appropriate API SDK for your platform. The SDKs include our API client code as well as sample code, and the API page also contains links to our API Overview and detailed documentation.

Once you have completed your development, visit http://dev.evernote.com/support/ to request activation of your API key on our production service.

Appendix: Evernote Blog on Single Notebook Authorization

We're pleased to announce the beta release for our most-requested API feature: Single Notebook Authorization — We're calling this feature "App Notebooks". Beginning this summer, third-party developers can connect their application exclusively to a single notebook within a given user's Evernote account.
This benefits both our users and developers:

- Users can choose the scope for which notebook an app can read
- Developers choose whether their application fits in this model
- User data is kept secure for business and personal notebooks
- Server-side optimizations allow for faster API access

How does App Notebook work?

When a user is prompted to authorize an application to access his/her Evernote account, the user will have a few options:

- Create a new notebook for the application to use; this is the default behavior; the notebook will have the same name as the application
- Create a new notebook with a custom name
- Select an existing notebook from their account to be used by the application

The good news is, App Notebooks will require very little in terms of code modifications — most of the magic happens on the server during the OAuth process.

We need your help!

This feature is about to enter private beta testing and we're looking for a small group of partners to help us work out the kinks. If you're interested, please let us know by filling out this form. All partners chosen for the beta will receive a complimentary year of Evernote Premium.

FAQs

"What if my app needs access to a user's entire Evernote account to function properly?"
During the API creation process, you'll have the opportunity to request full account access for your API key.
"What about existing API keys?"
All existing API keys will retain their current permission level.
If you have any questions, please post them in the comments below or in our developer forum and we'll answer them as quickly as possible. Also, documentation for this feature will be provided to testers during the beta period and published to the Evernote Developer site when App Notebooks go live this summer.

Appendix: Evernote Python SDK

We show an example of the Evernote SDK from a Python client.

Prerequisites

In order to use the code in this SDK, you need to obtain an API key from dev.evernote.com/documentation/cloud. You'll also find full API documentation on that page.

In order to run the sample code, you need a user account on the sandbox service where you will do your development. Sign up for an account at sandbox.evernote.com/Registration.action

In order to run the client client sample code, you need a developer token. Get one at sandbox.evernote.com/api/DeveloperToken.action

Python Client

To get started, download the Evernote Python SDK and follow the instructions at https://dev.evernote.com/doc/start/python.php.