# Activity: Threat Modeling

## Introduction

The purpose of this activity is to help you enumerate and model the security of your system architecture by looking at it from the point of view of *threats*. According to the Microsoft Threat Modeling methodology, we treat the word "threat" as a class of exploits. They fall into the following categories (STRIDE):

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Today, let's model **Dropbox** (http://www.dropbox.com/). Dropbox is a cloud-based storage system for your desktop and mobile devices. Store a file in a folder on your desktop, and it gets immediately uploaded to the server, and synchronized across all of your registered devices. Users can share folders in their Dropbox, allowing other users or the public view the files. Also, each change to a file is kept in a version history, so that users can revert to old copies of their files. All of this can be managed via a desktop client, a mobile client, and a webapp.

One feature you may not know about Dropbox is their *de-duplication engine*. The clients maintain a local list of hashes of the files in your folder. When a file is changed, Dropbox re-computes the hash, and sends all of the hashes to the server. The server then uses this logic on each hash:

- **New change.** If the server doesn't have the hash, upload the file.
- **No change.** If the server has the hash associated with the user's account already, no files are uploaded.
- **De-duplicate.** If the server has the hash already, but another user owns it, then make a record that a user is storing a copy of the file associated with that hash (i.e. no files are uploaded, just some bookkeeping)

De-duplication can provide a huge speedup in upload times and a gives big savings in storage for Dropbox. It also may have some security risks that may become apparent once we do some threat modeling.

Additionally, Dropbox servers are spread across multiple physical sites, and data is replicated across those sites for reliability and performance. This replication is handled by a single database management system. Mobile and desktop clients (even the webapp) all interact with a common API.

**Note:** you may make reasonable assumptions about the features of the system for the sake of this exercise. **This** (https://www.dropbox.com/en/help/1968) link may be helpful for understanding how

Dropbox works.

## Setup

This activity is for groups of 2-3.

1. Discuss as a group briefly what the architecture of the system will look like.
2. Fire up the Microsoft Threat Modeling tool. Before going further, save your file, calling it `Dropbox.tm7`. The tool can be found **here (https://aka.ms/threatmodelingtool)** (Windows only - sorry).
3. Start your model by creating **processes**, **interactors** and **stores**.
4. Next, add **data flow relationships**. Be sure to name every relationship with a noun describing a *type of data*. For example, "Authentication Data" or "Passwords" are types of data that would flow from one primitive to another. *Note:* you can have multiple dataflows from one primitive to another. Don't go too crazy with it - try to name your data in the most useful way possible.
5. Next, add **trust boundaries**.
6. For each primitive in the diagram, update the **Properties** to be as realistic as possible.
7. Now go to **Analyze** model.
8. **Expand** a few of the items. Notice the vast number of potential threats that can arise.
9. Let's **eliminate** some threats to be generated with some assumptions. Discuss as a group which threat categories you believe are not possible in Dropbox. Change those to "Not Applicable". Note that improbable is not the same as impossible! Be ready to discuss your choice with the class.
10. Now let's **revise a threat**, getting our inspiration from the diagram. Start with `DenialOfService` on a data store. Fill in the potential mitigations you might take.
11. Now let's also "test" your model hypothetically. Try to think of a security concern in your system. This could be an anti-requirement, a design concern, or a specific vulnerability. Does the model capture this concern somewhere? If it doesn't, revise the model a bit by adding flows, interactors, separating existing entities, or adding new trust boundaries.
    - **Note:** Any threats generated from new entities or relationships in the data flow diagram will appear at the end of the Threat List
12. At this point, **check the report** by going to Reports->Generate Full Report in the menu. Note anything you believe is missing.
13. **Continue** adding threats to the system by revising your model. Be ready to discuss your results with the class.
14. Save your Threat Model. Share the `Dropbox.tm7` save file and the HTML report with your group. We will use the threat model again during the semester.