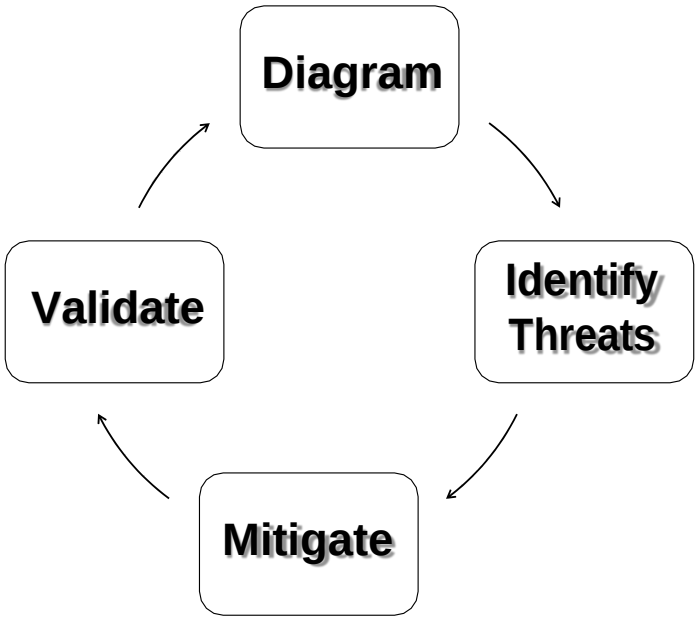
Threat Modeling

Introduction

The goal of this lab is to practice and think about the threat modelling process.

Almost all software systems today face a variety of threats, and the number of threats grows as technology changes. Malware that exploits software vulnerabilities grew 151 percent in the second quarter of 2018, and cyber-crime damage costs are estimated to reach $6 trillion annually by 2021. Threats can come from outside or within organizations, and they can have devastating consequences. Attacks can disable systems entirely or lead to the leaking of sensitive information, which would diminish consumer trust in the system provider. To prevent threats from taking advantage of system flaws, administrators can use threat-modeling methods to inform defensive measures.

In this lab, we will follow a process of threat modelling and have a basic view about how to design a secure system. The following diagram shows a nutshell of the process:



1 Diagram

The Data Flow Diagram (DFD) is a structured analysis and design method. It is traditional visual representation of the information flows within a system. Data Flow Diagram (DFD) is widely used for software analysis and design. A neat and clear DFD can depict a good amount of the system requirements graphically.

1.1 DFD Diagram Notations:

- **External Entity**
  An external entity can represent a human, system, or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

- **Process**
  A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.

- **Data Store**
  A data store (also known as data repository) holds information for processing. It represents a situation when the system must retain data because one or more processes need to use the stored data in a later time.

- **Data Flow**
  A data flow represents the flow of information, with its direction represented by an arrowhead that shows at the end(s) of flow connector.

- **Trust Boundaries**
  Trust boundaries represent the border between trusted and untrusted elements. They can be points/-surfaces where an attacker can interject. (e.g., Machine boundaries, privilege boundaries, and integrity boundaries are examples of trust boundaries) Threads in a native process are often inside a trust boundary, because they share the same privileges, rights, identifiers, and access. Processes talking across a network always have a trust boundary.

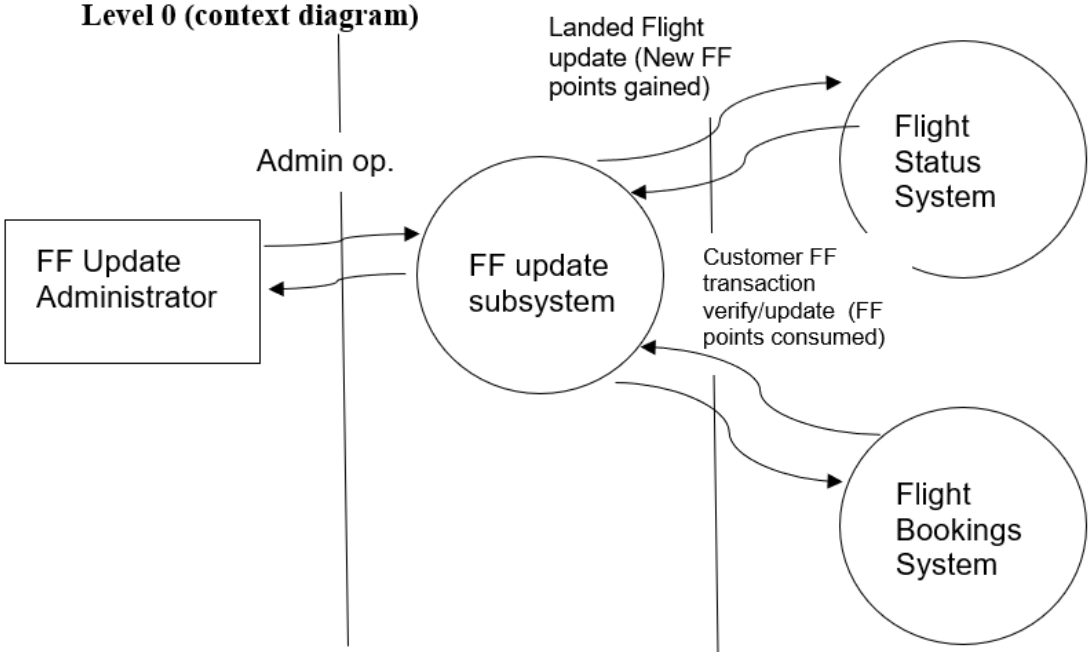
Question 1:
Assume we are going to design a frequent flyer update operation of an airline:

Frequent flyer points of valid customers are updated by a separate program. As soon as the plane lands at the destination, it triggers a program (say copy program) which copy (transmits) the frequent flyer information of the passengers in that flight into another area. An update program takes this data and updates the frequent points of the passengers. There was a breach detected where an individual who had access to the system which updates the frequent flyer info was able to rack up millions of miles without even flying!

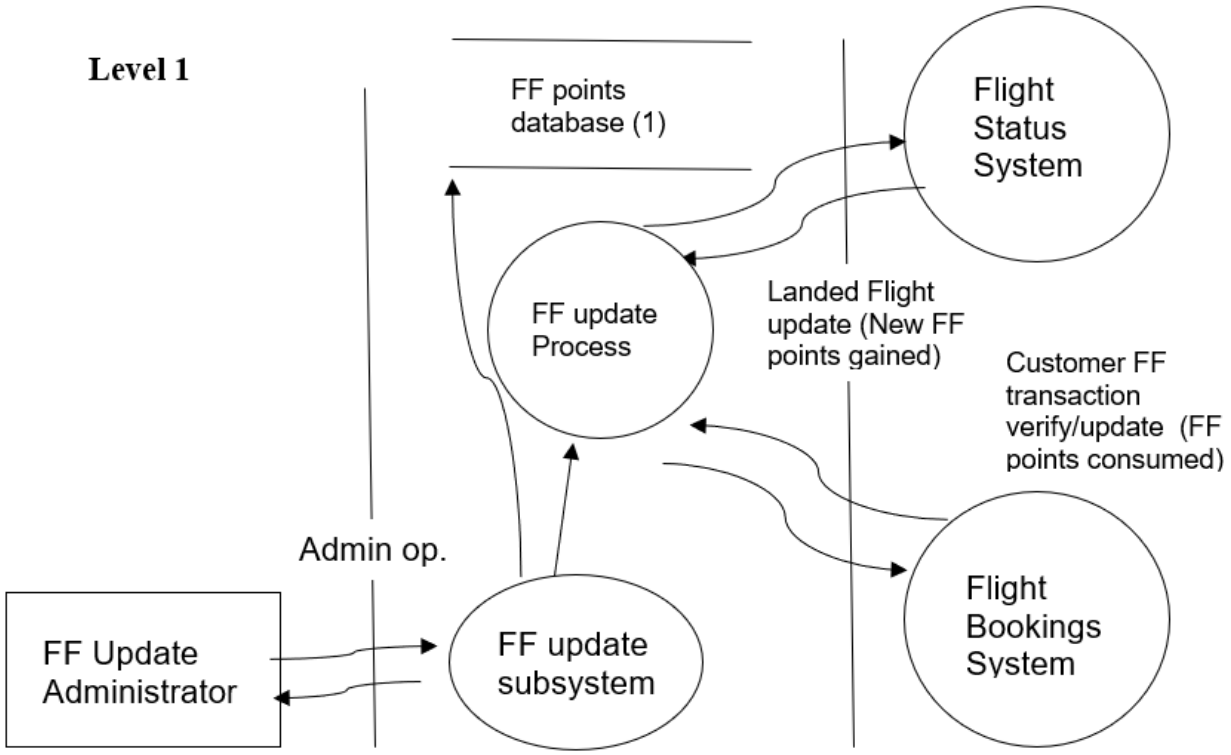
Draw a DFD for this sub-system and how it interacts with other parts of the system.

Possible DFDs:

Level 0 (context diagram)



Level 1



2 Identify Threat

In this step, we will use the STRIDE threat-modeling method to step through the diagram elements and identify threats. Invented in 1999 and adopted by Microsoft in 2002, STRIDE is currently the most mature threat-modeling method. It can help you identify potential threats in your product during a security analysis.

Recall: S.T.R.I.D.E. stands for:

- **Spoofing:** Spoofing refers to the act of posing as someone else (i.e., spoofing a user) or claiming a false identity (i.e., spoofing a process).
- **Tampering:** Tampering refers to malicious modification of data or processes. Tampering may occur on data in transit, on data at rest, or on processes.
- **Repudiation:** Repudiation refers to the ability of denying that an action or an event has occurred.
- **Information disclosure:** Information Disclosure refers to data leaks or data breaches. This could occur on data in transit, data at rest, or even to a process.
- **Denial of Service:** Denial of Service refers to causing a service or a network resource to be unavailable to its intended users.
- **Elevation of privilege:** Elevation of Privileges refers to gaining access that one should not have.

Question 2:

Example Question

a) What property each threat type is targeting?

Threat Type	**Security Property**
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

b) Could you provide an example for each threat type?

Spoofing: One user spoofs the identify of another user by brute-forcing password credentials.

Tampering: A user modifies data at rest/on disk.

Repudiation: A user denies performing a destructive action (e.g., deleting all records from a database).

Information Disclosure: A user is able to eavesdrop, sniff, or read traffic in clear text.

Denial of Service: The storage (i.e., disk, drive) becomes too full.

Elevation of Privilege: A user takes advantage to gain root-level privileges on a system.

Question 3:

The table in Appendix lists some of the common threats you will come across when developing software applications.

a) In the lab, for each threat, your task is to identify the type of threat according to the STRIDE classification.

b) Apply STRIDE to each element in your DFD and identify the threat(s).

3 Mitigate

After identifying the threats of a system, we need to find an appropriate mitigation approach for each one of them. Mitigation is an important part of Threat Modeling. since it constitutes the way to address or alleviate the threats and design a secure software solution.

There are four ways to address threats:

- Redesign to eliminate.
- Apply standard mitigations.
- Invent new mitigations.
- Accept vulnerability in design.

While redesigning and accepting the vulnerability do address the threats, we would focus more on the mitigation.

Inventing mitigations is hard. Commonly, mitigations are an area of expertise, such as networking, databases, or cryptography. Amateurs would make mistakes and the mitigation failures will appear to work until an expert looks at them. In this lab, we will focus more on applying standard mitigations.

Recall:
Some standard mitigation examples for each threat type:

Threat Type	Mitigation Examples
Spoofing	authentication
Tampering	validation of users' inputs and proper encoding of outputs
Repudiation	audit logging
Information Disclosure	encryption
Denial of Service	log rotation and monitoring/alerting when disk is nearing capacity
Elevation of Privilege	authorization mechanism

Question 4:

a) Identify the potential mitigation technique(s) for the threats in the Appendix table based on the threat type you found in the previous step.

b) Describe mitigation strategies for your DFD.

4 Validate

Now you can validate the whole threat model. If some threats that are not mitigated correctly are found, then this issue can be corrected in a next iteration of the threat model design process.

Asking yourself:

- Does diagram match the final code?
- Are all the threats enumerated?
- Is each threat mitigated?
- Are mitigations done in a right manner?

Question 5:

a) Identify the issues related to the threat and/or its mitigation for the threats in the table based on the threat type you find in the previous step.

b) Further improve your design by improving your DFD.

Additional Link about Threat Modeling:

- 'The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software'(Chapter 9)
  https://download.microsoft.com/download/8/1/6/816C597A-5592-4867-A0A6-A0181703CD59/Microsoft_Press_eBook_TheSecurityDevelopmentLifecycle_PDF.pdf
- 'Improving Web Application Security: Threats and Countermeasures'(Chapter 3 Threat Modeling)
  https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v%3dpandp.10)
- 'Security Briefs: Reinvigorate your Threat Modeling Process'
  http://msdn.microsoft.com/en-us/MAGAZINE/cc700352.aspx

Some open-source tools that you can use for this tutorial:
Microsoft Threat Modeling Tool: *https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling*
Threat Dragon: *https://owasp.org/www-project-threat-dragon/*

Appendix Table

THREAT	THREAT TYPE	MITIGATION TECHNIQUE(S)	DESCRIPTION OF ISSUES
Access to or modification of confidential HTTP data	T & I	Use SSL/TLS, WTLS (Wireless TLS), or possibly IPSec HTTPS, Encryption	Need to set up the HTTP server to use a private key and a certificate. Configuring IPSec can also be cumbersome process. Large performance hit when establishing the connection. Small performance hit for rest of the traffic.
Access to or modification of confidential RPC or DCOM data	T & I	Use integrity and privacy options	Might require code change. Small performance hit
A device that contains confidential data might be lost	I	Use PIN on device. Lock out after too many attempts.	Don't forget the PIN!
Flood service with too many connections	D	Provide throttling based on, perhaps, IP address. Require authentication. Firewalls, IDS/IPS	IP address checking will not work correctly through proxies. Need to give users accounts and passwords. May Need to install new hardware.
Attacker attempts to guess passwords	S, I, D& E	Use increased delays for each invalid password. Lock out after too many attempts. Support strong passwords. CAPTCHA, 2FA	Attacker might create a DoS attack by guessing and then force the account to lock out so that a valid user cannot access the account. In which case, lock the account out for small amount of time, say 15 mts. Need to add code to enforce password strength.
Read confidential cookie data	I	Encrypt the cookie at the server. Include timestamps/nonces. TLS	Need to add encryption code to the web site. Setup hardware/software
Tamper with cookie data	T	MAC or sign cookie at the server.	Need to add MAC or digital signature code to the web.

Access private/secret data	I, E	Do not store secret data in the first place! Or perhaps try using external device to store the data. If that won't work, consider hiding the data on a best effort basis, leveraging the operating system. ACLs, Encryption	Can be difficult problem to solve.
Attacker spoofs a server	S	Use an authentication scheme that supports server authentication, such as SSL/TLS, IPSec or Kerberos.	Configuration can be time consuming.
Attacker posts HTML or script to your site	T, E	Limit what can be posted using regular expression. Input sanitisation	Need to define appropriate regular expression and determine what is valid input.
Attacker opens thousands of connections but does nothing with them	D	Expire oldest connections using a scoring algorithm. For ex, admin connections do not time out.	You will waste time perfecting the scoring algorithm.
Unauthenticated connection can consume memory	D	Require authentication. Treat unauthenticated connections with disdain; never trust them. Be aggressive, and never allocate lots of resources to an unknown connection.	Need to support authentication and impersonation in your application
Your data packets can replayed	T, R, I & D	One approach is to use SSL/TLS, IPSec, or RPC/DCOM privacy to hide data. However, you can also enforce a packet count or timeout on the packets. Do this by appending a timestamp to the packet in the clear text and hashing the timestamp with the MAC on the packet. When the recipient software receives the packet, it can determine whether the packet is time worthy.	Can be tricky to get right. But it's worth the effort.
Attacker attaches a debugger to your process	T, I & D	Restricts which account have the SetDebug privilege.	Error detection, File extensions, No checking
Attacker gains physical access to hardware	S, T R, I, D & E	Physical security. Encrypt sensitive data, and do not store key on the hardware.	Never a fail-safe solution.
Attacker shuts down your process	D	Authenticate all administrative tasks. Require local administrator group membership to shut process down.	Need to perform Windows NT style checks in code.