# Thread Modeling

Il Thread Modeling comprende una **serie di tecniche, che vanno a supportare il processo di Risk Assessment**. In particolare, vanno a supportare l'identificazione di scenari di attacco contro le risorse critiche (ovvero gli assets) del sistema → con il termine Thread Modeling, quindi, si intendono tutte quelle tecniche che svolgono due punti:

- hanno l'obiettivo di identificare le varie minacce, che possono compromettere il funzionamento di un nuovo sistema o di un nuovo servizio sviluppato dall'organizzazione;
- aiutano ad identificare nuove misure di protezione o i controlli, necessari per gestire il rischio posto da tali attacchi.

> 💡 Tipicamente, queste tecniche di Thread Modeling vengono utilizzate quando il processo di Risk Assessment è finalizzato ad identificare problematiche di sicurezza nelle fasi iniziali di sviluppo di un nuovo sistema software o di un nuovo servizio sviluppato dall'organizzazione.

Capiamo, allora, che le tecniche di Thread Modeling vengono utilizzate per implementare il concetto di **Security-by-Design**, ovvero: Invece di considerare le problematiche di sicurezza **dopo** che il sistema è stato sviluppato (che può causare gravi problemi economici di implementazione del nuovo sistema o servizio), attraverso le tecniche di Thread Modeling possiamo limitare i problemi economici, andando ad i**ndividuare eventuali problematiche di sicurezza durante la fase di Design e di sviluppo del software**.

Le domanda a cui vogliamo rispondere attraverso un'attività di Thread Modeling sono le seguenti:

- Cosa stiamo modellando? Quindi, dobbiamo creare un modello del sistema software o del servizio che vogliamo implementare, focalizzandoci in particolar modo su:
  - quali sono le componenti del sistema;
  - come i dati vengono dati in input al sistema e quali output vengono prodotti;
  - chi interagisce con il sistema;

    - quali sono i **Trust boundary**, ovvero quali sono le parti del nostro sistema di cui ci possiamo fidare, dato che l'individuo che implementa e gestisce il sistema, può applicare delle misure di sicurezza su di esse.
- Come può essere attaccato il nuovo sistema? Ovvero come può un attaccante interno oppure esterno andare a compromettere le funzionalità offerte dal nuovo sistema?
- Cosa possiamo fare per ridurre il livello di rischio?
- Abbiamo fatto un lavoro sufficientemente buono nel costruire il modello del sistema e a identificare le misure di protezione?

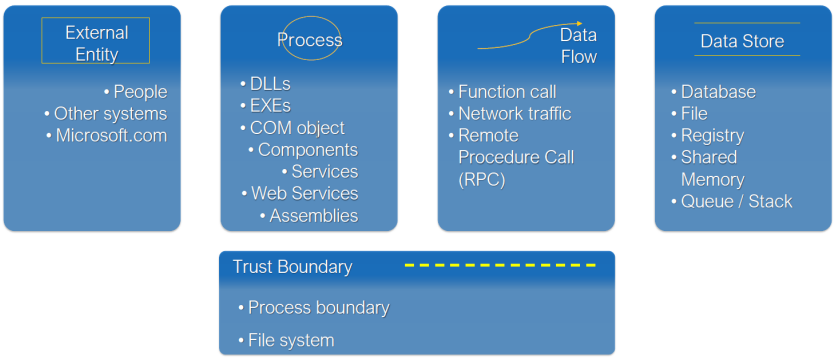
Una delle metodologie maggiormente utilizzate per fare Thread Modeling è **Microsoft STRIDE** → STRIDE è una componente fondamentale del processo di sviluppo del software implementato da Microsoft. Quindi, Microsoft non produce un prodotto senza prima aver eseguito un'attività di Thread Modeling attraverso STRIDE, dato che (come abbiamo detto prima) l'attività di Thread Modeling consente di ridurre i costi di sviluppo del software, dato che le misure di protezione vengono incluse nella fase di Design e di sviluppo del software. **STRIDE, inoltre, ci fornisce un processo strutturato e sistematico, per identificare le problematiche di sicurezza nelle fasi iniziali dello sviluppo di un software**, tenendo conto che i soggetti incaricati di sviluppare un prodotto software (come per esempio gli analisti e/o i software engineering) non hanno propriamente conoscenze di sicurezza.

Il processo è molto semplice e consiste di quattro fasi:

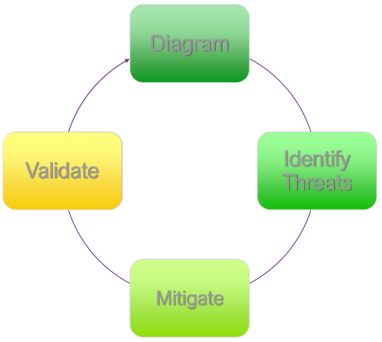
1. **Modellare il sistema** → STRIDE utilizza una particolare rappresentazione del sistema da sviluppare ed in particolare utilizza la rappresentazione dei **Data Flow Diagrams**. Per riuscire a modellare il sistema, dobbiamo avere una visione su:
    a. quali sono le componenti che costituiscono il sistema;
    b. come i dati vengono manipolati dalle componenti o meglio dire, come i dati passano da una componente all'altra;
    c. quali sono le possibili interazioni tra le varie componenti del sistema;
    d. dove i dati vengono memorizzati (come ad esempio: in un file oppure in un Database).

    Abbiamo già detto, che per rappresentare i dati, Microsoft STRIDE utilizza i **Data Flow Diagrams,** i quali sono formati da cinque componenti principali:

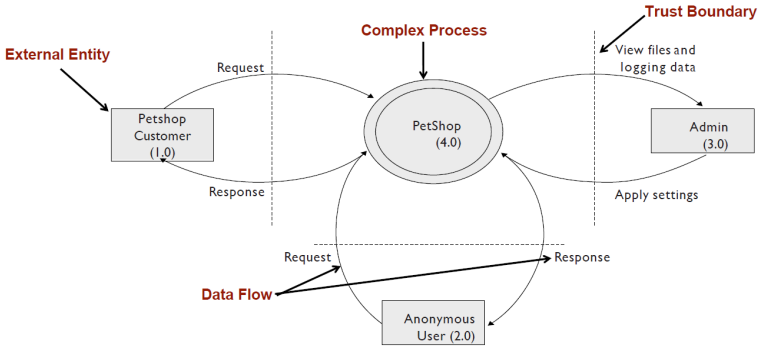
- **entità esterne** → le quali rappresentano i processi oppure gli utenti che interagiscono con il sistema e tipicamente, le entità esterne vengono rappresentate attraverso un rettangolo;
- **processi** → rappresentano le funzionalità implementate dal sistema o più in generale, rappresentano le componenti che implementano le funzionalità del sistema. Tali componenti, possono essere:
  - semplici → ovvero che implementano una sola funzionalità e in questo caso, esse (ovvero le componenti semplici) vengono rappresentate da un cerchio;
  - complesse → ovvero che implementano diverse funzionalità e in questo caso, esse vengono rappresentate da un doppio cerchio.
- **data flow** → possono rappresentare comunicazioni di rete (quindi, un utente che comunica con l'applicazione), oppure possono rappresentare chiamate di funzione → principalmente, quindi, i data flow rappresentano iterazioni che avvengono tra gli utenti e i processi del sistema;
- **data store** → nei quali salviamo i dati, gestiti dall'applicazione, in maniera permanente. Quindi, i data store, possono rappresentare i registri di memoria della macchina, la memoria di un processo oppure Database o file. Da notare, infine, che i data store sono rappresentati attraverso due linee parallele;
- **trust boundary** → essenzialmente, essi sono dei **limitatori** rappresentati attraverso una linea tratteggiata e in particolare, indicano quando un elemento del nostro sistema può essere considerato fidato oppure no. Solitamente i trust boundary, vengono messi tra un'entità esterna (la quale va a fornire dei dati al sistema e normalmente l'entità esterna opera ad un basso livello di privilegi) e i processi (i quali rappresentano le funzionalità implementate dal sistema e normalmente i processi operano ad un livello di privilegi più alto rispetto all'entità esterna).




2. Per ciascun elemento rappresentato nel diagramma, dobbiamo **identificare i potenziali attacchi**;
3. Una volta identificati tutti i potenziali attacchi, per ogni attacco che rappresenta un rischio elevato per l'organizzazione, dobbiamo **individuare delle misure di protezione**, le quali dovranno essere incorporate nel Design del sistema software;
4. Infine, dobbiamo **validare il modello del sistema e i Data Flow Diagrams** e ricontrollare se:
   a. eventualmente ci siamo dimenticati qualche possibile attacco relativo ad uno o più elementi dei Data Flow Diagrams;
   b. sono emerse nuove tipologie di attacco;
   c. siamo riusciti a mitigare tutti i rischi elevati per l'organizzazione.



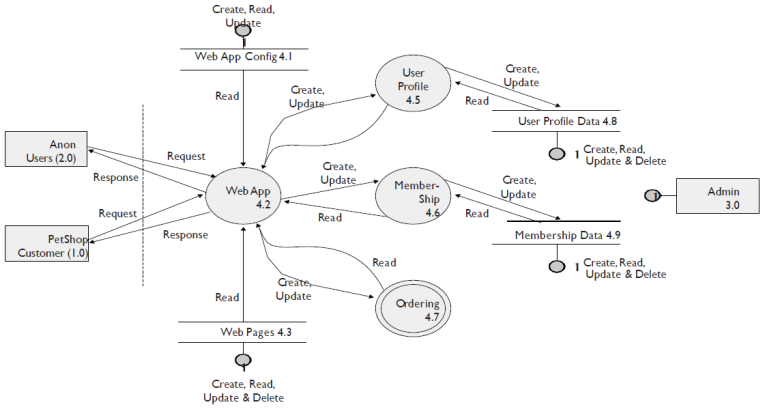

Andiamo ad analizzare in maniera più approfondita la prima fase (ovvero quella dei diagrammi) del processo STRIDE. In particolare, abbiamo che il primo Data Flow Diagram che viene realizzato è il cosiddetto **Diagramma di Contesto** (**Context Diagram**) → esso fornisce una rappresentazione ad alto livello, del sistema a cui vogliamo analizzare le problematiche di sicurezza. Un esempio di Context Diagram è il seguente:




Una volta eseguito il diagramma di contesto, ovverosia il diagramma di più alto livello, dobbiamo iterare su ciascun elemento del sistema ed in particolare, andare a vedere se i processi e i data stores possono essere decomposti in sotto-elementi. Per esempio, un'azienda permetterà di registrarsi e di conseguenza autenticarsi al sito Web, oppure permetterà agli utenti finali di effettuare degli ordini e di effettuare delle ricerche. Conseguentemente, tutte queste informazioni dovranno essere salavate da qualche parte (presumibilmente un Database) → quindi, il passo successivo alla realizzazione del diagramma di contesto, è di andare a dettagliare ulteriormente quest'ultimo (ovvero il Context Diagram), andando a specificare:

- le **funzionalità** implementate dai processi;
- i **dati** processati da ciascuna funzionalità.

Un esempio di questo, può essere visto nel seguente diagramma:




A questo punto, **si andrà ad utilizzare il diagramma dettagliato per individuare ed analizzare tutte le problematiche di sicurezza presenti nel sistema** (che dobbiamo analizzare) **e assegnarli un valore di rischio**. Quindi, per ciascun elemento del Data Flow Diagram dobbiamo fare una **tabella/lista**, in modo tale da raggrupparli per tipologia (iniziamo, allora, dalle entità esterne, per poi proseguire con i data flow, i processi ed infine i data store).

Nel fare questa operazione, possiamo accorpare insieme alcuni degli elementi. In particolare, possiamo accorpare tra loro alcuni elementi se:

- per implementarli viene utilizzata la **stessa tecnologia**;
- appartengono allo **stesso trust boundary**;
- i **dati**, manipolati dai processi e successivamente memorizzati in un data store, sono **esattamente gli stessi**.

(Nel nostro esempio, i data flow associati alle entità esterne PetShop Customer e Anonymous User possono essere analizzati considerando un unico data flow per la risposta e un unico data flow per la richiesta).

Il passo successivo è di individuare, per ogni elemento presente nella tabella/lista, gli attacchi. Sorge a questo punto spontanea la domanda: "**Come fa un utente, non esperto di sicurezza, ad individuare gli attacchi?**" Microsoft STRIDE definisce una serie di categorie di attacchi, che possono essere identificate per le componenti presenti nel Data Flow Diagram. In particolare, gli attacchi (che corrispondono all'acronimo STRIDE) possono essere:

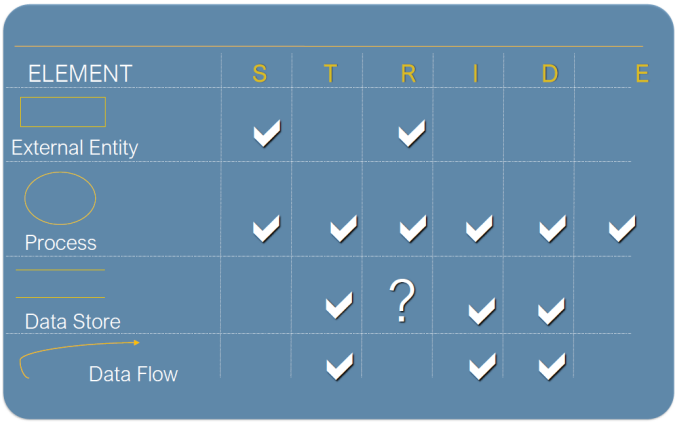
| Threat | Property we want |
|---|---|
| Spoofing | Authentication |
| Tampering | Integrity |
| Repudiation | Nonrepudiation |
| Information Disclosure | Confidentiality |
| Denial of Service | Availability |
| Elevation of Privilege | Authorization |

Come possiamo vedere, ognuno di questi attacchi va a compromettere una particolare proprietà di sicurezza del sistema. Più precisamente, abbiamo che:

- gli attacchi di **Spoofing** sono degli attacchi contro la proprietà di autenticazione, la quale verifica l'identità di un utente. Ovvero, la proprietà di autenticazione mira a garantire, che l'utente e/o il processo che viene autenticato, è effettivamente chi dice di essere. Negli attacchi di Spoofing, quindi, comprendiamo tutti quegli attacchi, in cui un attaccante impersonifica un utente legittimo del sistema oppure un componente software va ad impersonificare un utente legittimo;
- gli attacchi di **Tampering** vogliono compromettere la proprietà di integrità e di conseguenza, si concentrano sulla modifica non autorizzata dei dati oppure del codice (e quindi dell'implementazione delle funzionalità dei processi);

- gli attacchi di **Repudiation** vanno a compromettere la proprietà di Non-Repudiation, la quale garantisce che un utente non può negare di aver compiuto una determinata azione all'interno del sistema (per esempio, l'utente non può negare di aver mandato una determinata email, oppure non può negare di aver modificato un file, oppure di aver visitato un certo sito Web). In questa categoria, quindi, comprendiamo tutti gli attacchi che in un qualche modo vanno a modificare i log del sistema oppure che bypassano i meccanismi di identificazione;
- gli attacchi di **Information Disclosure** comprendono tutti quegli attacchi, che danno accesso alle informazioni ad utenti non autorizzati. Tipicamente, sono tutti attacchi in cui un utente ha accesso ad uno o più file di cui non dovrebbe aver accesso (per esempio, l'attaccante riesce ad accedere al file delle password oppure riesce ad accedere a file contenenti informazioni sensibili);
- gli attacchi di **Denial of Service** sono attacchi contro la disponibilità del sistema e/o di un servizio e di conseguenza, comprendono tutti quegli attacchi che rendono una risorsa e/o un servizio non disponibili sulla macchina delle vittime (per esempio, far crashare Windows oppure far crashare il sito Web);
- gli attacchi di **Elevation of Privilege** riescono a bypassare il meccanismo di Access Control, quindi comprende tutti quegli attacchi in cui l'attaccante riesce a compiere determinate azioni di "alto livello" senza un'autorizzazione adeguata (per esempio, permettere ad un utente remoto di Internet di eseguire comandi, oppure quando l'attaccante riesce a passare da privilegi di "utente limitato" a privilegi di utente amministratore).

"**Date queste categorie di attacco, come facciamo ad associarle agli elementi del data Flow Diagram?**" STRIDE ci fornisce una tabella, che ci indica per ogni tipologia di elementi nel Data Flow Diagram, a quali sono le categoria di attacco a cui sono soggette:

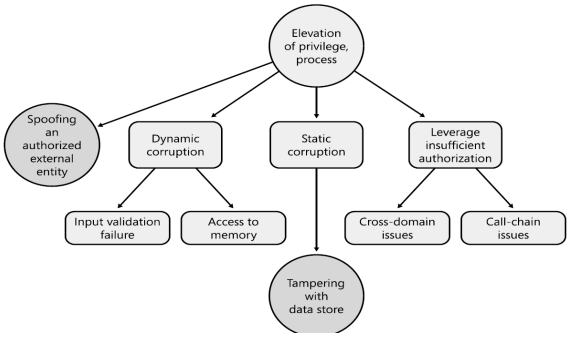


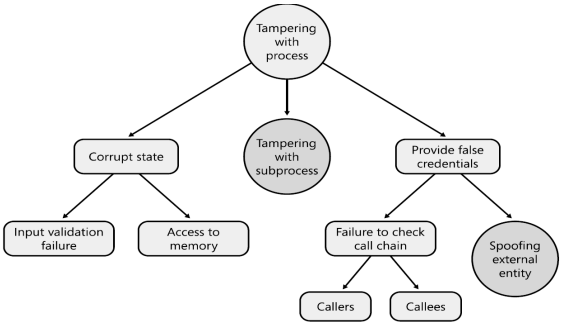
Per esempio, le entità esterne sono soggette ad attacchi di Spoofing e Repudiation. Stessa cosa vale per le altre tipologie di elementi presenti nel Data Flow Diagram. Da notare, che per i Data Store vi è un punto di domanda. Questo perchè, nel caso in cui il Database contenga dei dati, quali ad esempio i log delle attività fatte dagli utenti del sistema, allora i Data Store possono essere soggetti anche ad attacchi di tipo Repudiation, perchè l'attaccante potrebbe andare a modificare oppure cancellare i log per cancellare le sue "tracce" (nel momento in cui sta conducendo l'attacco), in modo da non essere identificato.

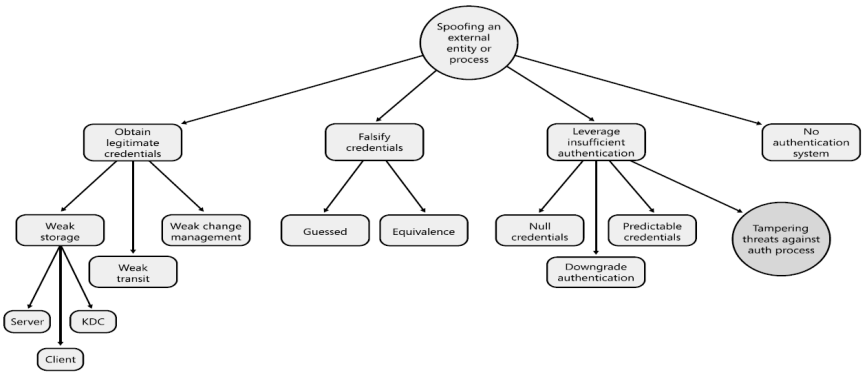
Una volta associate le categoria di attacco ai vari elementi del Data Flow Graph, il passo successivo è di andare a individuare degli scenari concreti, per ogni categoria di attacco, che si possono presentare all'interno del sistema. Per riuscire a fare ciò, Microsoft STRIDE fornisce i cosiddetti **thread tree patterns** → ovvero, per ogni categoria di elemento (del Data Flow Graph) e per ogni categoria di attacco che si applica all'elemento, STRIDE fornisce un albero, il quale cattura tutti i possibili scenari con ogni categoria di attacco può essere condotta. In particolare, STRIDE fornisce 12 thread tree patterns:

- 1 thread tree per lo Spoofing;
- 3 thread trees per il Tampering;
- 1 thread tree per il Repudiation;
- 3 thread trees per l'Information Disclosure;
- 3 thread trees per il Denial of Service;
- 1 thread tree per l'Elevation of Privileges.

Alcuni esempi sono:






Una volta fatto ciò, dobbiamo assegnare a ciascun scenario che abbiamo identificato per il nostro sistema, un livello di rischio, dato che il nostro obiettivo finale è quello di dare una priorità ai rischi e per ciascun rischio individuare delle misure di protezione, le quali verranno incluse nel Design del sistema. Ci sono vari modi per definire il rischio, ma Microsoft STRIDE considera quattro livello di rischio:

1. **molto alto** → deve essere risolto durante la fase di Design;
2. **alto** → deve essere risolto durante la fase di Design;
3. **medio** → deve essere risolto prima che il prodotto diventi una release candidate, ovvero prima che il prodotto venga rilasciato sul mercato;
4. **basso** → deve essere risolto solo se il tempo lo permette.

> 💡 L’obiettivo di questa fase, quindi, è di individuare tutti i rischi di livello 1,2 e 3, tenendo conto che i rischi di livello 1 e 2 devono essere assolutamente risolti nella fase di Design.

“**Come facciamo ad assegnare questi livelli di rischio ai diversi scenari?**”
STRIDE va a considerare diversi fattori, tra cui:

- se l’attacco ha come **obiettivo** la parte Client oppure la parte Server dell’applicazione, oppure entrambe le parti;
- il **livello di accesso** di cui l’attaccante ha bisogno per condurre l‘attacco (quindi, se l’attaccante ha bisogno di privilegi di utente normale, oppure di utente amministratore, oppure di utente root);
- se l’**effetto** di condurre l’attacco (specialmente negli attacchi di Denial of Service) è di compromettere le risorse del sistema per un tempo limitato, oppure in modo permanente;

- se l’attacco **comporta** una diffusione di informazioni sensibili o personali.

Vediamo alcuni esempi di scenari di attacco con il relativo livello di rischio:

| STRIDE Threat Type | Client/Server | Scope | Risk Level |
|---|---|---|---|
| Spoofing | Client | Ability for attacker to present a UI that is different from but visually identical to the UI that users are accustomed to trust in a specific scenario. | 3 |
| | Server | Client user or computer is able to masquerade as a different, random user or computer using a protocol that is designed and marketed to provide strong authentication. | 3 |

| STRIDE Threat Type | Client/Server | Scope | Risk Level |
|---|---|---|---|
| Tampering/Repudiation | Client/Server | Permanent modification of any user data or data used to make trust decisions in a common or default scenario that persists after restarting the OS/application. | 2 |
| | Server | Temporary modification of data in a common or default scenario that does not persist after restarting the OS/application. | 3 |
| | Client | Temporary modification of any data that does not persist after restarting the OS/application. | 4 |

| STRIDE Threat Type | Client/Server | Scope | Risk Level |
|---|---|---|---|
| Information Disclosure | Client/Server | Disclosure of PII (email addresses, phone numbers, credit card information) | 2 |
| | Client/Server | Attacker can locate and read information from anywhere on the system | 2 |
| | Client/Server | Attacker can locate and read information from known locations | 3 |
| | Client Server | Any untargeted information disclosure including runtime data | 4 |

| STRIDE Threat Type | Client/Server | Scope | Risk Level |
|---|---|---|---|
| Denial of Service | Client | Requires reinstallation of system and/or components | 2 |
| | Client | Requires cold reboot or causes Blue Screen/Bug Check | 3 |
| | Client | Temporary DoS: restart of application | 4 |
| | Server | Anonymous user sends a small amount of data | 2 |
| | Server | Authenticated permanent DoS | 3 |

| STRIDE Threat Type | Client/Server | Scope | Risk Level |
|---|---|---|---|
| Elevation of Privilege | Client/Server | Remote user with the ability to execute arbitrary code | 1 |
| | Client | Local, low-privilege user can elevate himself to another user, administrator or local system | 2 |
| | Server | Local authenticated user has the ability to execute arbitrary code or obtain more privilege than intended | 2 |

Una volta che abbiamo assegnato a tutti gli scenari un livello di rischio, il passo successivo è di realizzare un piano per mitigare/risolvere le minacce. In particolare, vi sono quattro modi per mitigare le minacce:

1. **accettare il rischio** e di conseguenza non fare nulla;
2. possiamo decidere che il rischio è talmente elevato, che decidiamo di **non implementare la feature**, ovvero la funzionalità del sistema, dato che magari non vi è una misura di protezione che possiamo applicare durante la fase di Design;
3. **accettiamo il fatto che vi sia una vulnerabilità nel sistema**;
4. **identificare una misura di protezione** per riuscire a mitigare la minaccia.

Anche in questo caso STRIDE aiuta gli sviluppatori (che non sono esperti in sicurezza), perchè per ciascuna categoria di attacchi suggerisce una serie di misure di protezione. Per esempio:

- per proteggersi da attacchi di Spoofing, possiamo implementare un meccanismo di autenticazione sicuro (quindi, implementare l'autenticazione a due fattori), oppure potremmo utilizzare dei protocolli di autenticazione (come per esempio Kerberos) basati sulla crittografia, oppure possiamo utilizzare delle tecnologie basate sulla crittografia per proteggere i dati e/o il codice (quali per esempio: la firma digitale);
- per proteggersi dagli attacchi di Tampering, possiamo limitare l'accesso, alle risorse del sistema, solamente a determinate categorie di utenti del sistema, oppure possiamo implementare tecnologie per controllare se i dati e/o il codice sono stati modificati;
- per proteggersi dagli attacchi di Repudiation, possiamo assicurarci che solamente determinati processi (come per esempio, solamente il processo di logging) possono generare dei logs; possiamo andare ad autenticare i soggetti che compiono azioni di modifica dei logs, attraverso un meccanismo di autenticazione, oppure andando a firmare i logs con la chiave privata dell'utente e del processo che li genera e li salva;
- per proteggersi dagli attacchi di Information Disclosure, possiamo implementare meccanismi di controllo degli accessi (i quali restringono l'accesso ai dati), oppure possiamo cifrare i dati;
- per proteggersi dagli attacchi di Denial of Service, possiamo limitare gli accessi ai processi o ai dati, oppure bisogna implementare soluzioni che consentono al processo e/o al servizio di gestire un'elevata quantità di traffico;
- per proteggersi dagli attacchi di Elevation of Privilege, possiamo gestire i permessi assegnati agli utenti del sistema (ovvero, non dare permessi non necessari agli utenti che non ne hanno bisogno).

Infine, l'ultima fase consiste nell'andare a valutare i risultati prodotti nelle fasi precedenti. In questa ultima fase, quindi, dobbiamo assicurarci che nel Data Flow Diagram non ci siamo dimenticati di analizzare qualche elemento importante e che per ogni elemento è stato individuato almeno uno scenario di attacco e che per i rischi di livello 1 e 2 sono state individuate delle misure di mitigazione adeguate.

Il thread modelling, quindi, comprende tutte quelle tecniche che ci permettono di identificare nelle fasi iniziali (di Design e di sviluppo) di un sistema software e/o di un servizio online, tutte le problematiche di sicurezza e le relative misure di protezione che possono adottate per gestire il rischio. Una delle metodologie maggiormente utilizzate per fare ciò è Microsoft STRIDE, il quale attraverso un approccio sistematico e strutturato, consente anche a non esperti di sicurezza, di individuare e ridurre le problematiche di progettazione della sicurezza. In particolare, STRIDE fornisce una tassonomia delle varie categorie di attacchi (quindi: Spoofing, Tampering, Repudiation…) e per ciascuna categoria di attacco fornisce degli esempi di istanze concrete (su come l'attacco può essere realizzato) e suggerendo per ciascuna categoria di attacco delle misure di protezione.