Lezione 3
Definizioni

Programmazione Sicura (6 CFU), LM Informatica, A. A. 2016/2017
Dipartimento di Scienze Fisiche, Informatiche e Matematiche
Università di Modena e Reggio Emilia
http://weblab.ing.unimo.it/people/andreolini/didattica/programmazione-sicura

Quote of the day

(Meditate, gente, meditate...)

“If you think technology can solve your security problems, then you don’t understand the problems and you don’t understand the technology.”

Bruce Schneier (1963-)

Crittografo, esperto di sicurezza informatica, membro EFF

Autore di “Applied Cryptography”

Autore di “Cryptography Engineering”

Asset

(Una entità che interagisce con il mondo circostante)

Un **asset** è una entità generica che interagisce con il mondo circostante. La natura dell'entità è variegata e dipende dal contesto di cui si parla.

- Un edificio (palazzo, ufficio, gabbiotto di ingresso).
- Dispositivo hardware (PC, portatile, apparato).
- Software (OS, libreria, applicazione).
- Dato sensibile (personale, proprietà intellettuale).
- Algoritmo (cifratura, autenticazione).
- Procedura (riconoscimento di una persona).
- Persona (impiegato, segretario, programmatore, CEO).

Focus del corso

(Asset → Software scritto o da scrivere)

Il focus del corso è sulla programmazione sicura.

Pertanto: asset → software.

- Già scritto (da altri o da noi).
- Da scrivere.

Caratterizzazione di un asset

(Funzionalità, prestazioni, sicurezza)

Un asset è sempre è caratterizzabile in tre ambiti distinti.

Funzionalità. L'insieme delle operazioni che l'asset è chiamato a svolgere (come da requisiti e specifiche).

Prestazioni. Il livello prestazionale che un asset è supposto mantenere durante la sua esecuzione.

Sicurezza. Il grado di protezione di un asset contro operazioni "pericolose" (maliziose o accidentali).

Interazioni e rischi

(Cosa può andare storto nell'interazione con un asset?)

Un utente può interagire con un asset in tre modi:

- correttamente;
- incorrettamente, in modo involontario;
- incorrettamente, in modo malizioso.

Come già visto in precedenza, un **uso non corretto** di un asset **può comportare rischi gravi**, fa cui:

- furto di informazioni sensibili, beni preziosi, denaro;
- modifica/distruzione di informazioni sensibili;
- compromissione di servizi fisici e virtuali.

Minaccia

(La spada di Damocle pendente sull'asset)

Una **minaccia** (**threat**) è una qualunque potenziale causa di incidente, risultante in un danno all'asset. Ad es.:

- danno fisico (incendio, inondazione, avvelenamento).
- evento naturale (climatico, sismico, vulcanico).
- perdita di servizi (elettricità, telecomunicazioni).
- compromissione di informazioni (spionaggio, furto).
- avaria tecnica (apparati, software).
- compromissione di funzioni (errore, abuso, negazione).

Le **minacce** possono tramutarsi in realtà in due modi: **accidentale** o **doloso**.

Classificazione STRIDE

(Classificazione delle minacce informatiche operata da Microsoft)

Una classificazione possibile delle minacce in ambito informatico è quella introdotta da Microsoft: **STRIDE**.

Spoofing (spacciarsi per un'altra entità).
Tampering (modificare le informazioni).
Repudiation (impossibilità di attribuire le azioni).
Information Disclosure (divulgare informazioni).
Denial of Service (negare un servizio).
Elevation of Privilege (elevare i propri privilegi).

Utente attaccante

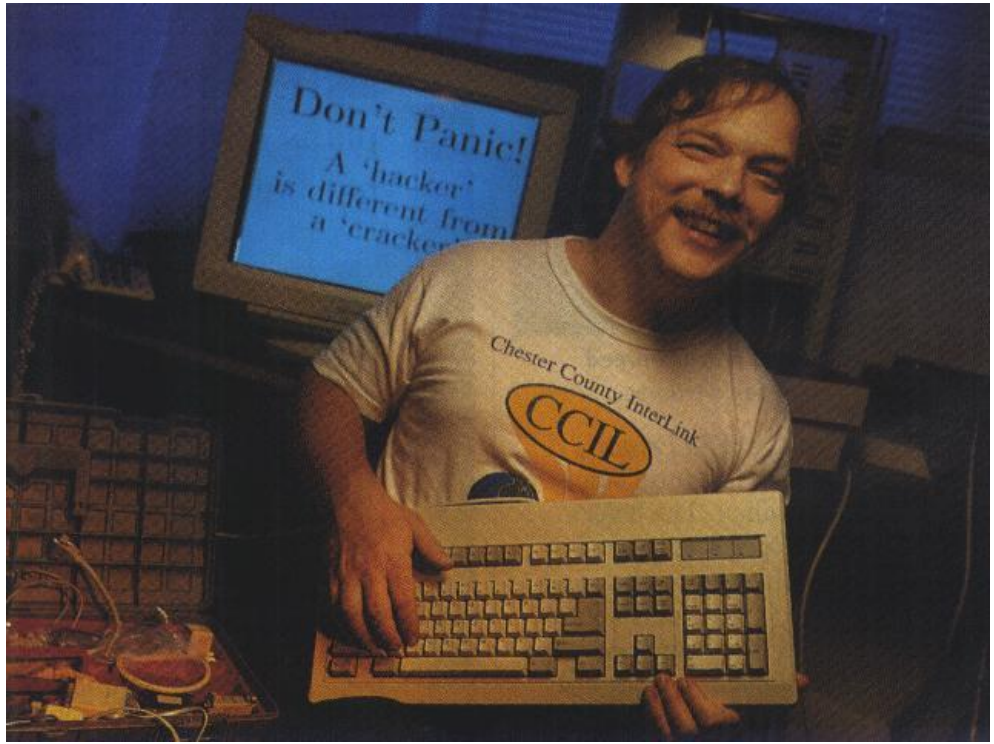
(Porta avanti le minacce ad un asset in maniera deliberata)

Un **utente attaccante** (o, più semplicemente, **attaccante**, o anche **cracker**) interagisce con l'asset:

- in modo deliberato, malizioso, doloso (mai accidentale);
- alla ricerca di un malfunzionamento sfruttabile;
- con il fine ultimo di tramutare una minaccia in una realtà;
- motivato dal conseguimento di un vantaggio (economico, di potere, di conoscenza).

A hacker is not a cracker

(So says Eric Raymond, father of the Open Source Movement; so should you!)



http://www.catb.org/esr/faqs/hacker-howto.html

Classificazione degli attaccanti 1/3

(Vale oggi e non è detto che valga domani; vogliate bene lo stesso al docente)

White hat (ethical hacker). Viola asset per fini non maliziosi (stimarne il livello di sicurezza, svolgere test di penetrazione).

Black hat. Viola asset per fini maliziosi o per tornaconto personale.

Gray hat. È una via di mezzo tra white hat e black hat. Scenario tipico: violano asset e, in cambio di soldi, si offrono di irrobustirli.

Elite hacker. Uno status sociale che descrive gli individui più abili. Sono in grado di scoprire nuovi difetti e sfruttarli.

Classificazione degli attaccanti 2/3

(Vale oggi e non è detto che valga domani; vogliate bene lo stesso al docente)

Script kiddie. È un individuo senza particolari abilità, se non quella di usare strumenti automatici scritti da altri (con poca convinzione di causa).

Noob. Un principiante, privo di competenze ma con tanta voglia di imparare.

Blue hat. Un individuo pagato espressamente per collaudare asset in vista del loro lancio sul mercato.

Classificazione degli attaccanti 3/3

(Vale oggi e non è detto che valga domani; vogliate bene lo stesso al docente)

Hacktivist. Un individuo che viola asset per pubblicizzare messaggi sociali, ideologici, religiosi, politici. Gli hacktivist svolgono attività di:

- cyber terrorismo (defacciamento Web, Denial of Service);
- freedom of information (rendono accessibili al pubblico documenti altrimenti non pubblici).

Nation state. Team di attaccanti sponsorizzati da una nazione. Sono in grado di orchestrare attacchi sofisticati.

Organized criminal gang. Team di criminali che violano asset per ottenere profitti illegali.

Difetto

(Una qualunque deviazione da requisiti e specifiche di progetto)

Un **difetto** è una qualcunque deviazione da requisiti e specifiche di progetto esibita da un asset.

- L'asset è, di per sé, implementato correttamente.
- L'implementazione fa fede a requisiti male interpretati o non previsti inizialmente.

Bug

(Un errore di implementazione a livello di codice sorgente)

Un **bug** è un errore di implementazione dell'asset (a livello di codice sorgente, nel caso di software).

I programmatori hanno fatto il possibile per seguire requisiti e specifiche imposti.

L'implementazione, tuttavia, non rispecchia la specifica imposta.

OSS.: un difetto non implica necessariamente un bug. I due concetti sono distinti.

Debolezza

(Un difetto o bug che potrebbe rendere reale una minaccia di sicurezza)

Una **debolezza** (**weakness**) è un difetto o bug che potrebbe, sotto opportune ipotesi, rendere reale una minaccia di sicurezza.

Un asset debole non è detto che sia compromesso.

Deve poter essere raggiungibile da un attaccante.

Una volta raggiunto, deve poter essere violabile.

Vulnerabilità

(Un particolare tipo di weakness)

Una **vulnerabilità software** o **vulnerabilità** (**software vulnerability**, **vulnerability**) è una debolezza che un attaccante è in grado di usare per tramutare una minaccia in realtà.

Una vulnerabilità è la somma di tre fattori:

una debolezza esistente;

l'accessibilità dell'attaccante alla debolezza;

la capacità dell'attaccante di "sfruttare" la debolezza per conseguirne un vantaggio.

Un invito alla riflessione

(Provate a rispondere alla domanda)

Staccando il cavo di rete del PC, parecchie vulnerabilità si trasformano in debolezze.

Perché?

Un invito alla riflessione

(Provate a rispondere alla domanda)

Staccando il cavo di rete del PC, parecchie vulnerabilità si trasformano in debolezze.

Perché?

Perché in tal modo parecchie debolezze non sono più accessibili da un attaccante remoto.

Vulnerabilità a diversi livelli

(Progetto, implementazione, configurazione, ...)

Una vulnerabilità può annidarsi in diversi ambiti di un software.

- Progetto (introduce un modus operandi non sicuro).
- Implementazione (bug).
- Configurazione (credenziali in chiaro/default).
- Utente (con comportamenti sciocchi/insicuri).
- (Ab)Usi non previsti (SPAM).

Focus del corso

(Vulnerabilità a livello di design ed implementazione)

Il focus del corso è sulla programmazione sicura.

Pertanto:

design → progetto software.

bug → errore di implementazione nel software.

vulnerabilità → a livello di progetto/implementazione.

Vettore di attacco, superficie di attacco

(Misurano l'esposizione di un asset alle mazzate^W^Wagli attacchi)

Un **vettore di attacco** è un qualunque strumento attraverso il quale si può veicolare una vulnerabilità.

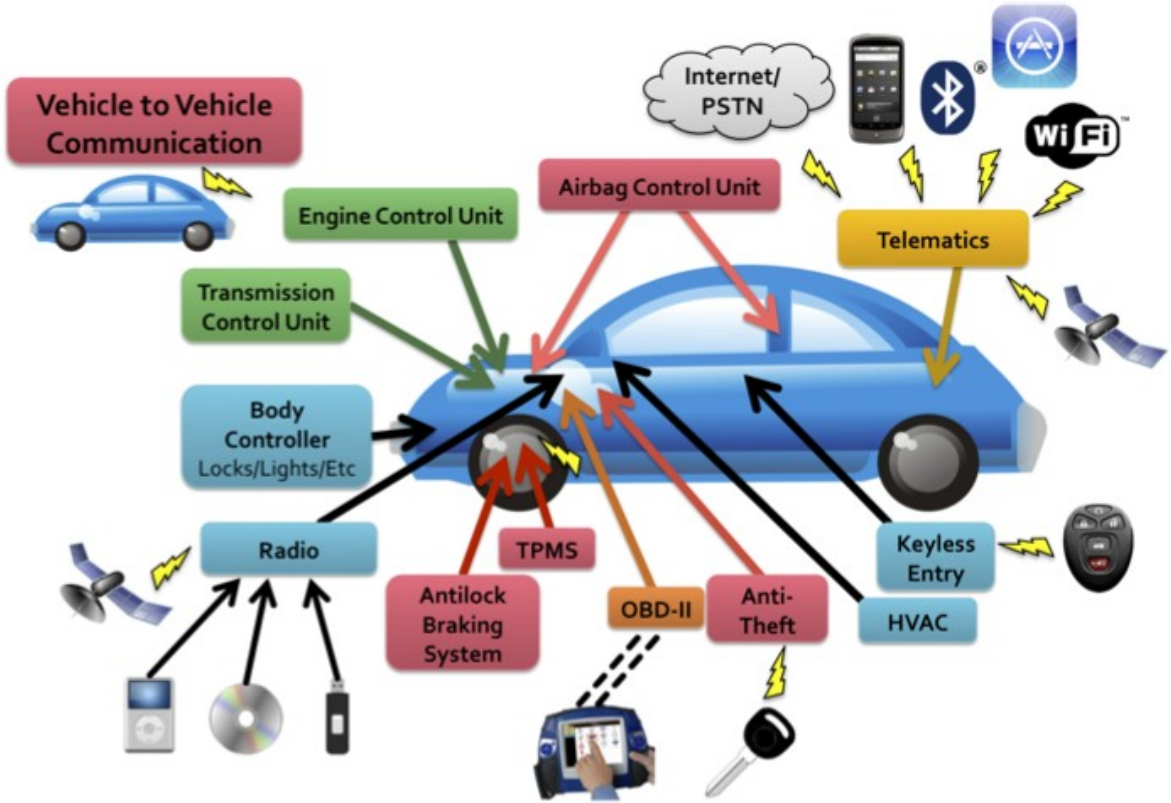
- Una connessione TCP verso un server, una shell locale, una linea telefonica a disposizione, una serratura incustodita, una guardia…

La **superficie di attacco** di un asset è l'insieme di tutti i suoi vettori di attacco.

- Essa misura l'esposizione di un asset agli attacchi.

Un esempio di superficie di attacco

(L'asset "automobile"… Good luck with choosing your next car!)



Exploit

(Una procedura che sfrutta una vulnerabilità per concretizzare una minaccia)

Un **exploit** è una procedura che:

- sfrutta una vulnerabilità;
- causa un comportamento inatteso in un asset;
- permette di trasformare una minaccia in realtà.

Spot the weaknesses!

(Trovate tre debolezze nello pseudocodice sorgente che segue)

```
int balance;

void decrease(int amount)
{
    if (balance <= amount)
        { balance = balance - amount; }
    else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Spot the weaknesses!

(Trovate tre debolezze nello pseudocodice sorgente che segue)

```
int balance;

void decrease(int amount)
{
    if (balance <= amount)
        { balance = balance - amount; }
    else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

<= dovrebbe essere >=.
Banalissimo errore logico.

Spot the weaknesses!

(Trovate tre debolezze nello pseudocodice sorgente che segue)

```
int balance;

void decrease(int amount)
{
    if (balance <= amount)
        { balance = balance - amount; }
    else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Che succede se **amount** è negativo?
Mancata validazione di input non fidato.
Errore di progetto?
O di implementazione?

Spot the weaknesses!

(Trovate tre debolezze nello pseudocodice sorgente che segue)

```
int balance;

void decrease(int amount)
{
    if (balance <= amount)
        { balance = balance - amount; }
    else { printf("Insufficient funds\n"); }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Che succede se la somma è troppo grande per un **`int`**? Dipende dalla piattaforma sottostante.
Problema "low-level".

Perché succede tutto questo?

(Bella domanda! Proviamo a dare una risposta sensata...)

Mancanza di consapevolezza sulle minacce esistenti, ma anche su cosa proteggere.

Mancanza di conoscenza dei problemi posti e delle soluzioni fornite in ambito di sicurezza.

Complessità eccessiva degli asset (ad es., il software è scritto in linguaggi complessi, usa API corpose, esegue su una infrastruttura enorme).

Precedenza alla funzionalità sulla sicurezza ("la sicurezza non fattura").

Come affrontare questi problemi?

(Bisogna intendere la sicurezza come "gestione del rischio")

Le funzionalità esposte da un un asset implicano un rischio di abuso.

La **sicurezza informatica** si occupa della mitigazione del rischio connesso a tali funzioni.

Si mitiga il rischio fino a quando?

- Fino a quando non sparisce del tutto?
- Oppure fino ad un livello minimo di decenza?
- Nel caso, chi stabilisce il livello minimo di decenza?

Politica di sicurezza

(Definisce il giusto livello di sicurezza in base ad una analisi dei rischi)

La **politica di sicurezza** (**security policy**) è un documento che definisce in modo non ambiguo il livello di sicurezza di un asset.

Che cosa significa "l'asset è sicuro"?

Da quale interazione ci si intende difendere?

Da quali utenti si intende difendere?

La politica di sicurezza nasce spesso da una **analisi dei rischi** (**risk analysis**) che cerca di identificare:

cosa potrà andare storto con l'asset in questione;

quanto sarà probabile un incidente;

quanto costerà gestire un incidente.

Meccanismi di sicurezza

(A disposizione per l'attuazione di una politica di sicurezza)

Un **meccanismo di sicurezza** è un qualunque strumento a disposizione per l'attuazione delle contromisure previste da una politica di sicurezza. I meccanismi si dividono in tre grandi categorie.

Meccanismi di prevenzione. Impediscono alla radice le cause legate al rischio di sicurezza.

Meccanismi di rilevazione. Misurano e notificano le possibili cause legate al rischio di sicurezza.

Meccanismi di reazione. Ripristinano il sistema, riparano le falle, perseguono i malintenzionati.

Meccanismi di prevenzione/rilevazione

(Come operano?)

Meccanismi di prevenzione. Tendono ad impedire le interazioni tra un asset ed un utente.

Esempi: firewall, controllo di ammissione richieste.

Meccanismi di rilevazione. Controllano le interazioni tra un asset ed un utente.

Esempi: sistema di rilevazione delle intrusioni, controllo degli input di una funzione.

Perché è bella la prevenzione?

(Perché tiene lontani anche gli utenti maliziosi)

Un asset soggetto a prevenzione non è in grado di interagire con nessuno (neppure con eventuali malintenzionati).

Esempi:

- un PC completamente sconnesso dalla rete è inattaccabile.
- un software che non riceve input non può essere da essi manomesso.
- un consulente a cui non è permesso l'accesso fisico in una ditta non può fare danni in essa.

Perché è insufficiente la prevenzione?

(Perché impedisce le interazioni con gli utenti normali)

Un asset soggetto a prevenzione è funzionalmente inutile per gli utenti normali.

È necessario un aumento dell'esposizione dell'asset affinché gli utenti possano fruirne.

- Apertura di porte TCP in un servizio di rete.
- Lettura di input in una applicazione locale.
- Accesso di un consulente esterno ad aree interne della ditta.

Perché è necessaria la rilevazione?

(Perché individua le anomalie nelle interazioni permesse)

Con l'aumento dell'esposizione di un asset, aumentano anche i rischi.

I rischi vanno controllati con gli appositi meccanismi di rilevazione.

- Controllo del traffico sulle porte TCP aperte.
- Controllo degli input passati ad una applicazione.
- Controllo delle attività di un consulente esterno all'interno di una ditta.

Operazioni tipiche dei meccanismi

(Autenticazione, controllo degli accessi, auditing, reazione)

Autenticazione. Stabilire che un utente che si presenta all'asset sia effettivamente chi dice di essere.

Controllo degli accessi. Stabilire se un utente ha i diritti ad accedere a funzioni/informazioni presso l'asset.

Auditing. Monitorare e registrare le interazioni di un utente con un asset.

Azione. Svolgere azioni correttive per far rispettare la politica di sicurezza.

Obiettivi primari della sicurezza

(Preservare tre proprietà: confidenzialità, integrità, disponibilità)

L'obiettivo delle politiche e dei meccanismi di sicurezza è quello di preservare tre proprietà.

Confidenzialità. Impedire l'interazione in lettura con un asset ad utenti non esplicitamente autorizzati.

Integrità. Impedire l'interazione in scrittura con un asset ad utenti non esplicitamente autorizzati.

Disponibilità. Rendere disponibili le funzioni dell'asset ad utenti esplicitamente autorizzati.

Le tre proprietà sono note con il termine di **triade CIA** (**C**onfidentiality, **I**ntegrity, **A**vailability).

Chi è più importante?

(Tra confidenzialità, integrità e disponibilità?)

L'integrità è quasi sempre più importante della confidenzialità.

Esempi:

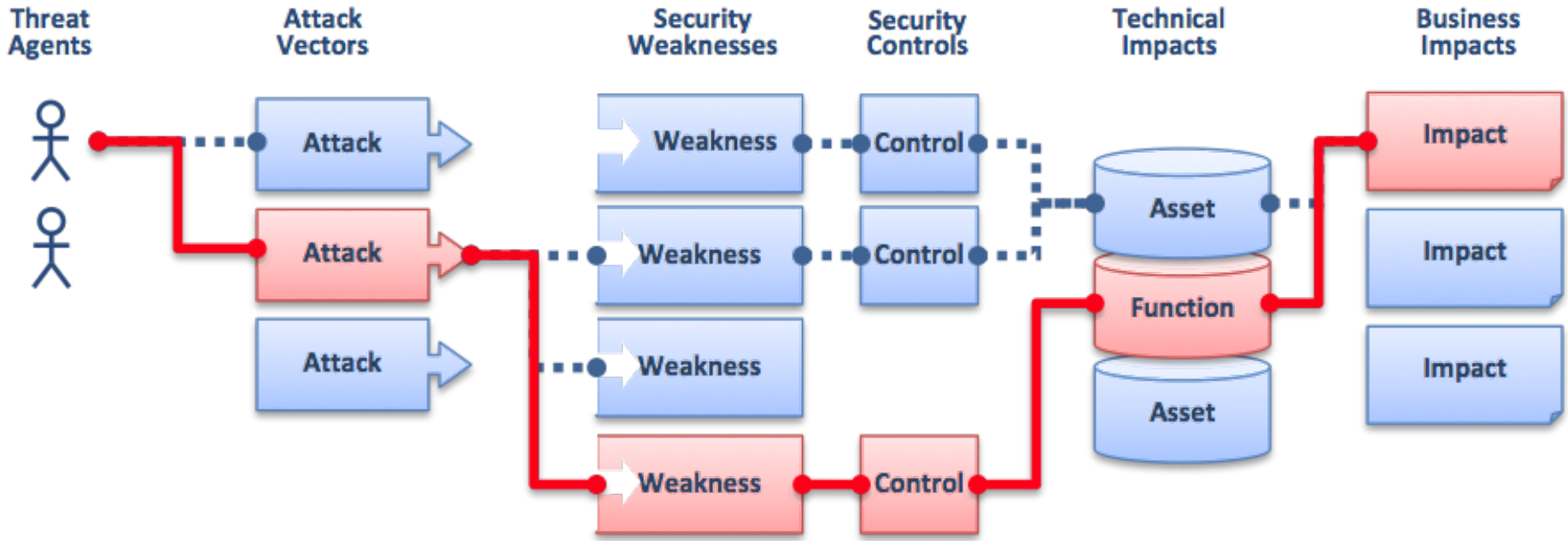
- conto bancario;
- informazioni sanitarie;
- la totalità dei software usati (SO, librerie, applicazioni).

La disponibilità potrebbe addirittura essere d'intralcio in alcuni scenari.

- La privacy di un utente può implicare la mancata disponibilità di informazioni e servizi a terzi.

L'operato di un attaccante

(Dalla minaccia all'esecuzione)



Un diagramma E-R riassuntivo

(Illustra le relazioni tra attacco e difesa)