5 PROGETTAZIONE DEL SOFTWARE SECURE/PRIVACY BY DESIGN

5.1 Processi di sviluppo del software sicuro

Questo capitolo illustra alcuni framework di processo di riferimento nello sviluppo sicuro delle applicazioni software, quali: il framework BSA for Secure Software (BSA), il Software Assurance Maturity Model (SAMMM), il Building Security in Maturity Model (BSIMM), il Comprehensive and Lightweight Application Security Process (CLASP) e il ciclo di vita di sviluppo sicuro (SDL) di Microsoft (MS-SDL).
Lo scopo nel presentare diversi framework di sviluppo è quello di ottenere una migliore comprensione delle tecniche di Threat Modeling e di come questo si integra in un ciclo di sviluppo sicuro del software. Indipendentemente dalla metodologia di sviluppo adottata, la definizione dei controlli di sicurezza nelle applicazioni inizia con, o precede, la fase di progettazione e continua per tutto il ciclo di vita dell'applicazione in risposta alle mutevoli esigenze organizzative, in un ambiente costantemente a rischio e in continua evoluzione.

5.1.1 BSA Framework for Secure Software (BSA)

BSA[3] è un framework di sviluppo sicuro del software sviluppato dalla statunitense Software Alliance BSA[4], che riunisce e consolida in un unico quadro, le migliori pratiche di sicurezza da attuare durante tutto il ciclo di sviluppo del software, in modo tale da poter essere efficacemente misurato, indipendentemente dall'ambiente di sviluppo e dallo scopo del software stesso. Il Framework si concentra sul prodotto software (includendo il Software-as-a-Service) considerando sia i processi adottati nello sviluppo e nella gestione del prodotto, sia la resilienza del prodotto stesso. L'obiettivo del framework è quello di integrare, piuttosto che sostituire, le linee guida per i processi di gestione del rischio organizzativo. Per quanto possibile, cerca l'allineamento con gli standard internazionali riconosciuti e nel contempo di rimanere flessibile, adattabile, focalizzato sui risultati e basato sul rischio.

5.1.2 Open Software Assurance Maturity Model (SAMM)

OpenSAMM[5] è un framework supportato da OWASP[6] (Open Web Application Security Project) che si basa su un insieme di procedure di sicurezza legate a quattro importanti funzioni di business critiche, coinvolte nello sviluppo del software, vale a dire Governance, Costruzione, Verifica e Distribuzione. Ciascuna funzione di business adotta tre pratiche di sicurezza e ciascuna di esse è suddivisa in tre livelli di maturità. La valutazione delle minacce è la prima pratica di sicurezza adottata durante la funzione di business "Costruzione". Questa utilizza il Threat modeling per identificare i potenziali rischi. Successore diretto di questo framework è l'OWASP SAMM[7] rilasciato per consentire alle organizzazioni di misurare e migliorare la postura di sicurezza del software prodotto. OpenSAMM non si lega ad alcun approccio di modellazione delle minacce e raccomanda l'uso di STRIDE della Microsoft o TRIKE come possibili opzioni.

3 https://ww2.bsa.org/~/media/Files/Policy/BSA_2019SoftwareSecurityFramework.pdf

4 https://ww2.bsa.org/

5 http://www.opensamm.org/

6 https://www.owasp.org/images/6/6f/SAMM_Core_V1-5_FINAL.pdf

7 https://owaspsamm.org/

5.1.3 Building Security in Maturity Model (BSIMM)

L'iniziativa di sicurezza BSIMM[8] è stata progettata per aiutare i team di sviluppo software a comprendere e pianificare la sicurezza in un ciclo di vita di sviluppo delle applicazioni, studiando le pratiche di cinquantuno importanti iniziative di sicurezza software. Aziende come Google, Adobe, Intel, Visa, Nokia, Sony e Microsoft hanno partecipato alla ricerca guidata da Gary McGraw (leader esperto di settore nella sicurezza del software, vicepresidente della Security Technology presso la Synopsys Inc. SNPS, autorità riconosciuta a livello mondiale per la sicurezza del software e autore di otto libri tra i più venduti su questa tematica). La metodologia risultante ha unito le migliori pratiche (parere del team BSIMM) in un'unica iniziativa. Si tratta di dodici pratiche raggruppate in quattro domini, Governance, Intelligence, SSDL Touchpoint (pratiche associate all'analisi e alla garanzia di particolari manufatti e processi di sviluppo del software. Tutte le metodologie di sicurezza del software includono l'analisi dell'architettura, la revisione del codice e i test di sicurezza) e Deployment, utilizzate per organizzare le attività del framework di sicurezza del software. La prima pratica all'interno del dominio "Intelligence" è costituita dai modelli di attacco. Il Threat modeling viene utilizzato durante questa fase per modellare gli attacchi e per creare una base di conoscenza relativa all'applicazione.
BSIMM non è un approccio innovativo per lo sviluppo sicuro del software. Questo framework ha raccolto dati su attività effettivamente svolte dai leader dell'industria promuovendo poi quelle migliori e più utilizzate, anche se per tradizione, le società di software erano riluttanti a divulgare informazioni riguardo le loro pratiche interne. L'impiego della metodologia BSIMM risulterebbe vantaggiosa per un'organizzazione che intende adottare un'iniziativa di sicurezza o migliorare/maturare le pratiche esistenti. Tuttavia, tale framework non fornisce sufficienti informazioni di dettaglio riguardo il Threat Modeling.

5.1.4 Comprehensive, Light-weight Application Security Process (CLASP)

Il CLASP[9], è un framework di sicurezza supportato dall'OWASP che contiene best practices formalizzate per attuare la sicurezza, in modo strutturato e ripetibile, nei cicli di vita di sviluppo di software in essere o in divenire. Il framework esiste dal 2005, ma non sono stati registrati recenti aggiornamenti del progetto. È stato originariamente sviluppato dalla Secure Software Inc. e successivamente donato a OWASP che lo ha rilasciato come soluzione completa di sicurezza a favore di quelle organizzazioni che intendono adottarlo. Insieme all'SDL di Microsoft, CLASP è stato riconosciuto come uno dei processi originali di alto profilo per lo sviluppo di software sicuro. CLASP, si basa su sette best practices che sono alla base di tutte le attività connesse alla sicurezza. La valutazione dell'applicazione è una di queste pratiche, che promuove un'analisi dei requisiti di sicurezza e la progettazione del sistema basata su l'utilizzo del Threat Modeling.
Il CLASP non è legato ad alcun metodo di modellizzazione specifico e non ha sviluppato un proprio approccio. Al fine di proporre un'iniziativa di sicurezza, a causa di una mancata evoluzione di questo progetto, è preferibile prendere in considerazione un framework maggiormente innovativo.

5.1.5 Microsoft's Security Development Lifecycle (SDL)

Il ciclo di vita di sviluppo sicuro (SDL[10]) è una metodologia introdotta dall'iniziativa "Trustworthy Computing" di Microsoft. SDL mira a ridurre i costi di manutenzione del software e ad aumentare l'affidabilità implementando la sicurezza in ciascuna fase del ciclo di vita dello sviluppo.

[8] https://www.bsimm.com/

[9] https://www.owasp.org/index.php/CLASP_Concepts

[10] https://www.microsoft.com/en-us/sdl/default.aspx

Il processo consiste in pratiche di sicurezza, raggruppate in sette fasi distinte: formazione, requisitazione, progettazione, implementazione, verifica, rilascio e monitoraggio/manutenzione. Uno degli aspetti chiave dell'SDL è l'introduzione del Threat Modeling nella fase di progettazione, che promuove l'individuazione preventiva delle vulnerabilità presenti nelle applicazioni e alcune volte persino i potenziali difetti di progettazione. Queste informazioni vengono poi utilizzate per attuare eventuali piani di mitigazione o modifiche progettuali.

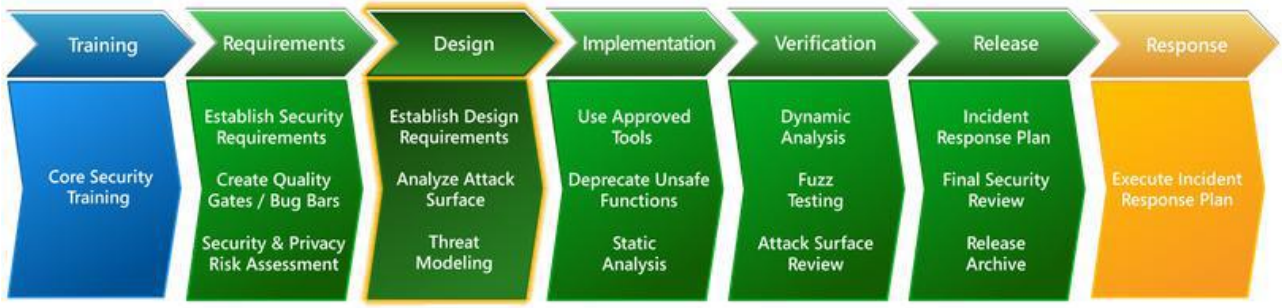


Figura 1 - Processo del ciclo di sviluppo sicuro di Microsoft

Sono disponibili diversi strumenti, non solo per il Threat Modeling ma per ciascun aspetto dell'SDL, tutorial online, documentazione, forum e blog di supporto. Il processo può essere utilizzato anche con le metodologie di sviluppo Waterfall e Agile e può essere applicato a qualsiasi piattaforma e implementato da qualsiasi organizzazione indipendentemente dalla sua dimensione.
L'obiettivo che tutte le metodologie di modellazione delle minacce condividono è lo sviluppo di un processo a passi iterativi che un team di sviluppo può facilmente seguire durante la valutazione di un sistema software.
Microsoft ha sviluppato e pubblicato una propria metodologia di modellazione delle minacce denominata STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) e diversi approcci per l'analisi dei rischi, tra cui la DREAD[11] (Damage potential, Reproducibility, Exploitability, Affected users, Discoverability). Hussain, Erwin e Dunne hanno indicato la STRIDE[12] come la metodologia di Threat modeling più ampiamente utilizzata [1].
Al termine dell'attività STRIDE, viene prodotto un elenco di vulnerabilità. Tali vulnerabilità vengono poi classificate secondo un indice di priorità (basso/medio/alto) utilizzando una analisi qualitativa del rischio come la DREAD che a sua volta consente di definire la priorità delle opportune contromisure . Il prodotto finale dell'analisi STRIDE/DREAD può inoltre indirizzare in una fase successiva un eventuale Penetration Test (sulla base delle vulnerabilità riscontrate è possibile delimitare il perimetro che sarà poi oggetto di PT).
La Figura che segue illustra le fasi principali nella preparazione e nell'esecuzione di una modellazione delle minacce.

[11] https://resources.infosecinstitute.com/qualitative-risk-analysis-dread-model/

[12] https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

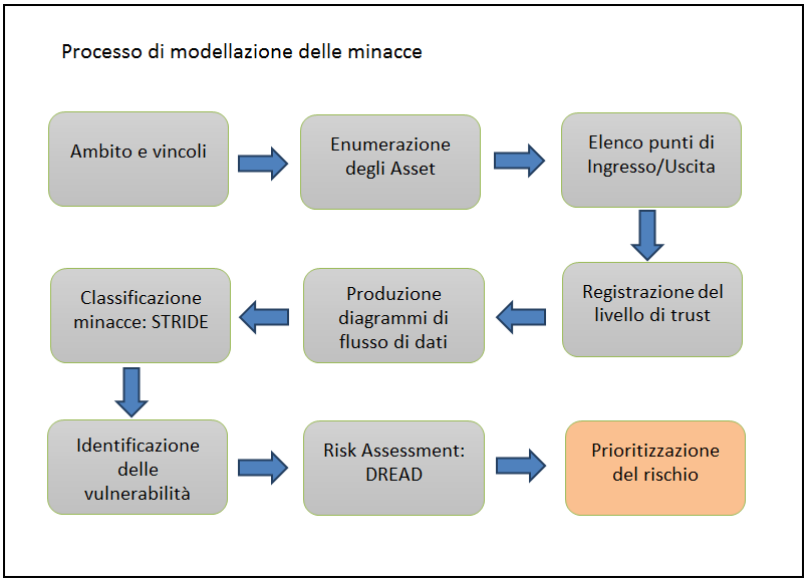


Figura 2 - SDL: Passi nella modellazione delle minacce

I processi riguardanti le diverse metodologie di Threat Modeling possono differire fra loro, ma hanno tutti in comune la raccolta delle informazioni riguardanti l'applicazione, il suo ambiente operativo, come questa viene utilizzata e distribuita, come eventuali vulnerabilità vengono identificate e come i relativi rischi vengono valutati.
La tabella che segue riassume le caratteristiche principali di un processo di modellazione delle minacce:

Caratteristica/ambito	Descrizione
Ambito e vincoli	La modellazione delle minacce può essere un processo che richiede tempo, soprattutto se si modella un'applicazione di scala enterprise. Avere chiari gli obiettivi, aiuta a focalizzare l'attività di modellazione delle minacce. L'obiettivo del modello dovrebbe essere ben definito fin dall'inizio. Nel caso di sistemi più grandi, il processo può dare in modo più rapido maggior valore, riuscendo a limitare il campo di azione solo ad aree specifiche. Nella fase iniziale del processo di Threat Modeling, quando l'ambito e i vincoli sono ben definiti, informazioni aggiuntive come scenari d'uso, dipendenze e note di sicurezza forniscono supporto a una migliore comprensione del sistema stesso.
Scenari d'uso	Viene creato un elenco di possibili scenari di utilizzo o di trascorsi che dimostrano l'uso previsto dell'applicazione. Ciò può aiutare nell'identificazione di potenziali aree di abuso presenti nel sistema.
Dipendenze esterne/interne	Le applicazioni software spesso dipendono da altri sistemi o componenti. Questi possono avere un impatto diretto sulla sicurezza del sistema in oggetto. È importante elencare tali dipendenze e individuare l'impatto che questi possono avere sul sistema stesso. Alcuni esempi posso essere l'utilizzo di: Antivirus, Firewalls, librerie di terze parti e sistemi di autenticazione.
Note di sicurezza	Vengono registrati dettagli come le informazioni già conosciute relative agli aspetti di sicurezza di un sistema, eventuali ipotesi fatte sull'applicazione, o trade-off a livello di disegno. Queste note di sicurezza possono aiutare il processo decisionale.

Asset	Vengono elencati gli asset che potrebbero essere di interesse per un potenziale avversario. Questi possono essere di tipo fisico o logico. Possibili esempi di asset sono: le credenziali di accesso, i dettagli di una carta di credito, le chiavi di crittografia o i dettagli dell'utente.
Punti di Ingresso/Uscita	I punti di ingresso/uscita sono quei punti del sistema dove i dati entrano o escono. Questi vengono talvolta indicati come "punti di attacco" poiché soggetti ad uno o più potenziali attacchi. Ciascuno di questi, corrisponde ad una parte del sistema per la quale dovrebbero essere implementate opportune misure di sicurezza.
Livelli di Fiducia o di Trust	Un livello di trust viene utilizzato per definire i privilegi che un'entità esterna deve avere per accedere al sistema. Questi possono essere classificati in base ai privilegi assegnati o alle credenziali fornite e fanno riferimento a punti di ingresso/uscita di risorse protette. Possibili esempi sono l'utente anonimo del sistema, l'utente autenticato del sistema e l'amministratore. I livelli di trust vengono applicati ad ogni punto di entrata/uscita.

5.2 Secure by Design

Le applicazioni la cui architettura non viene valutata dal punto di vista della sicurezza sono applicazioni 'fragili'. L'analisi, fatta in concerto sia sull'architettura dell'applicazione che sulle interazioni, accresce la visibilità dei team IT circa gli ambiti della sicurezza e del risk management. Gli architetti delle applicazioni software sono responsabili della realizzazione del progetto per garantire l'adeguata copertura ai rischi derivanti sia da un uso standard del sistema, sia da un attacco dannoso.
Nella fase di design di una nuova applicazione o di re-ingegnerizzazione di un'applicazione esistente, per ciascuna caratteristica funzionale, è necessario analizzare aspetti aggiuntivi quali:

- Gli aspetti di sicurezza nel processo riguardante tale funzione sono stati massimizzati?
- Premeditatamente, come si potrebbe abusare di tale funzione?
- La funzione deve essere attivata per impostazione predefinita? In caso affermativo, esistono limiti o alternative che potrebbero contribuire a ridurre il rischio derivante da tale funzione?

Andrew van der Stock chiama il suddetto processo "Thinking Evil", e raccomanda di mettersi nei panni dell'attaccante e di pensare a tutte le possibili alternative per abusare di ogni singola funzione, considerando i tre pillar fondamentali (Riservatezza, Integrità, Disponibilità) e utilizzando a sua volta il modello STRIDE.
Utilizzando il modello di rischio di minaccia STRIDE/DREAD discusso nel presente documento, è possibile intraprendere la giusta strada per adottare formalmente un'architettura sicura nello sviluppo applicativo del software.
Il concetto di sicurezza nell'architettura inizia ad esistere nel momento in cui vengono modellati i requisiti di business e prosegue fino a quando l'ultima istanza dell'applicazione viene dismessa.

5.2.1 Principi base del secure design

"Security by design" è un concetto base dell'ingegneria del software e definisce un software progettato espressamente per essere sicuro (anticipando e minimizzando a priori gli impatti delle vulnerabilità che potrà manifestare in produzione) capace di garantire i seguenti caposaldi della sicurezza dell'informazione:

- **R**iservatezza: consentire l'accesso solo ai dati per i quali l'utente è autorizzato.
- **I**ntegrità: garantire che i dati non vengano manomessi o alterati da utenti non autorizzati.

- **D**isponibilità: garantire che i sistemi e i dati siano sempre disponibili agli utenti autorizzati quando questi ne hanno bisogno.

I principi base del Secure Design possono essere sintetizzati come segue:

1) Ridurre al minimo la superficie di attacco.
   Ogni funzione aggiunta a un'applicazione immette una certa dose di rischio nell'applicazione. L'obiettivo di uno sviluppo sicuro è quello di ridurre il rischio complessivo riducendo la superficie di attacco. Ad esempio, un'applicazione web implementa un Help on line con una funzione di ricerca. La funzione di ricerca può essere vulnerabile agli attacchi di SQL injection. Se l'accesso alla funzione di Help viene limitata ai soli utenti autorizzati, la probabilità di attacco è certamente ridotta. Se la funzione di ricerca relativa all'Help è stata implementata attraverso routine di convalida dei dati centralizzate, la capacità di eseguire l'iniezione SQL viene drasticamente ridotta. Tuttavia, se la funzione di Help è stata riscritta per eliminare la funzione di ricerca (ad esempio attraverso un miglioramento dell'interfaccia utente), ciò elimina quasi totalmente la superficie di attacco, anche se la funzione di Help in generale rimane disponibile su Internet.

2) Stabilire valori predefiniti sicuri.
   Esistono diversi modi per offrire agli utenti un'esperienza "out of the box". Tuttavia, per impostazione predefinita, tale esperienza dovrebbe essere sicura e dovrebbe spettare all'utente ridurre il proprio livello di sicurezza, se consentito. Ad esempio, la scadenza e la complessità della password dovrebbero essere abilitate per impostazione predefinita. Gli utenti potrebbero comunque essere autorizzati a disattivare tali funzioni per semplificare l'uso dell'applicazione di conseguenza incrementando il loro livello di rischio.

3) Adottare il principio del minimo privilegio.
   Il principio del "minimo privilegio" suggerisce che agli account venga dato il livello minimo di privilegio richiesto necessario per poter eseguire i relativi processi di business. Ciò include i diritti dell'utente, i permessi di accesso alle risorse come la limitazione nell'uso della quota di CPU, i permessi di accesso alla memoria, alla rete e al file system. Ad esempio, se un server middleware richiede solo l'accesso alla rete, l'accesso in lettura a una tabella di database e la possibilità di scrivere su un log, ciò definisce tutte le autorizzazioni che dovrebbero essere concesse. In nessun caso al middleware devono essere concessi privilegi amministrativi.

4) Adottare il principio di difesa a più livelli o in profondità.
   Il principio della difesa a più livelli anche detta "Defense in Depth", suggerisce che laddove un controllo può risultare adeguato, più controlli che affrontano i rischi in modi diversi possono rappresentare l'approccio migliore. Quando tali controlli vengono eseguiti in profondità, questi possono rendere le più gravi vulnerabilità presenti nel sistema incredibilmente difficili da sfruttare e quindi improbabili. In termini di codifica sicura, lo si può tradurre in una forma di validazione basata su più livelli, in controlli di auditing centralizzati, richiedendo che tutti gli utenti vengano autenticati su tutte le pagine dell'applicazione. Ad esempio, è improbabile che un'interfaccia amministrativa che presenta delle difettosità sia vulnerabile a possibili attacchi anonimi se l'accesso alla rete di gestione dell'ambiente di produzione viene opportunamente bloccata, viene controllata l'autorizzazione amministrativa dell'utente e vengono registrati su log tutti gli accessi.

5) Gestire gli errori in modo sicuro.
   A causa di diversi motivi, le applicazioni software falliscono durante la propria esecuzione. La modalità in cui queste falliscono determina il fatto che siano sicure o meno. La gestione sicura degli errori è un aspetto fondamentale per le applicazioni sicure e resilienti. Esistono due tipi principali di errori che richiedono una particolare attenzione:
   - le eccezioni che si verificano durante l'elaborazione di un controllo di sicurezza,

- le eccezioni generate dal codice che non sono "rilevanti per la sicurezza".

È importante che queste due distinte tipologie di eccezioni non permettano un comportamento che una contromisura software normalmente non consentirebbe. In termini di sviluppo, è opportuno considerare il fatto che un meccanismo di sicurezza possa dare seguito a tre distinti possibili risultati:

- consentire l'operazione;
- bloccare l'operazione;
- sollevare un'eccezione.

In generale, il meccanismo di sicurezza dovrebbe essere progettato in modo tale che un eventuale fallimento segua lo stesso percorso di esecuzione implementato per il blocco dell'operazione. Ad esempio, metodi di sicurezza come "isAuthorized" o "isAuthenticated" dovrebbero restituire tutti "false" se è stata sollevata un'eccezione durante l'elaborazione. Se gli stessi controlli di sicurezza prevedono delle eccezioni, è opportuno che ci sia chiarezza su cosa significhi esattamente ciascuna condizione di errore.

6) Non affidarsi ai servizi di terze parti o a risorse esterne.
Molte organizzazioni utilizzano le capacità di elaborazione di terzi parti, che molto spesso adottano politiche e postura di sicurezza significativamente diverse. È altamente improbabile che si possa influenzare o controllare tali parti, siano esse utenti privati, grandi fornitori o partner. Pertanto, è opportuno non affidarsi implicitamente a quei servizi/sistemi che vengono gestiti esternamente. Tutti i servizi/sistemi esterni devono essere trattati allo stesso modo.

7) Separare i ruoli.
Un controllo antifrode fondamentale è la separazione dei ruoli. Ad esempio, chi richiede un finanziamento non può essere l'unico garante, né dovrebbe ricevere direttamente il finanziamento. Ciò impedisce al soggetto di richiedere numerosi prestiti e di affermare poi di non averli mai avuti. Alcuni ruoli hanno livelli di fiducia diversi rispetto ai normali utenti. In particolare, gli amministratori sono diversi dagli utenti normali. In linea generale, gli amministratori non dovrebbero essere anche utenti dell'applicazione. Ad esempio, un amministratore dovrebbe essere in grado di accendere o spegnere il sistema, impostare le politiche sulle password, ma non dovrebbe essere in grado di accedere al frontend del sistema come utente super privilegiato, ed essere in grado di compiere un'azione per conto di altri utenti. La regola generale nella separazione dei ruoli è che l'entità che approva un'azione, l'entità che svolge un'azione e l'entità che controlla tale azione devono necessariamente essere separate. La separazione dei ruoli è un controllo importante sia nell'ambito finanziario che in quello informatico. L'obiettivo è quello di eliminare la possibilità che un singolo utente non compia e nasconda un'azione a lui non consentita. Separando i ruoli di autorizzazione, implementazione e monitoraggio, il tentativo di frode può essere portato a termine con successo solo attraverso la collusione delle parti. Lo sviluppo sicuro di una applicazione software dovrebbe sempre prevedere la separazione dei ruoli (ad esempio, implementando l'accesso basato sui ruoli) e tale concetto dovrebbe anche essere integrato nel ciclo di vita di sviluppo del dell'applicazione (ad esempio, limitando l'accesso degli sviluppatori alle librerie di produzione).

8) Segmentare l'infrastruttura di rete.
La segmentazione e la segregazione delle reti sono strategie altamente efficaci che un'organizzazione può attuare per limitare l'impatto dovuto ad un'intrusione di rete. Se implementate correttamente, tali strategie possono rendere significativamente più difficile per un avversario localizzare e accedere alle informazioni più sensibili dell'organizzazione aumentando la probabilità di rilevare tale attività in modo tempestivo. Storicamente, la segmentazione e la segregazione della rete veniva implementata attraverso l'impiego di un firewall perimetrale; oppure attraverso l'utilizzo di una coppia di firewall che delimitando una zona demilitarizzata fornendo un ambiente separato tra zone totalmente attendibili e non attendibili.

9) Ridurre i “single point of failure”.
   Un “single point of failure” è un componente critico del sistema che ha la capacità di arrestare le operazioni del sistema stesso in caso di guasto. L’adozione di tali elementi diviene inaccettabile in quei sistemi che richiedono affidabilità e disponibilità, come le applicazioni software, le reti o le supply chains. Dal punto di vista della sicurezza, la presenza di un “single point of failure” fa sì che nel momento in cui questo viene a mancare (a seguito di una qualsiasi causa) anche il sistema smette di funzionare inficiandone la disponibilità. Pertanto è opportuno progettare il software evitando per quanto possibile la scelta dei “single point of failure” definendo gli opportuni requisiti di ridondanza (load balancer, cluster, etc.).

10) Mantenere la semplicità nell’uso del sistema.
   È essenziale che la user interface sia progettata considerando la facilità d'uso, tale che gli utenti possano regolarmente e automaticamente applicare gli opportuni meccanismi di protezione. Inoltre, gli errori vengono ridotti al minimo nel momento in cui la percezione dell'utente rispetto ai propri obiettivi di protezione corrisponde ai meccanismi di sicurezza messi in campo e che l’utente stesso è obbligato ad usare. Tale principio stabilisce che la soluzione di sicurezza adottata deve essere comprensibile per gli utenti finali, sia nella sua fase di progettazione che durante il suo utilizzo (facilità di installazione, configurazione e usabilità), nascondendo la complessità introdotta dai meccanismi di sicurezza.

11) Non affidarsi ad una sicurezza basata sull’oblio.
   Questo principio afferma che l’adozione di tecniche basate sul concetto dell’occultazione nella risoluzione di problematiche di sicurezza non dovrebbero mai essere considerate affidabili. Se l’applicazione prevede come requisito di sicurezza che il suo URL di amministrazione venga nascosto in modo che possa essere sicuro, in realtà non lo è affatto. Diversamente, è più appropriato prevedere gli opportuni controlli di sicurezza al fine di mantenere sicura l'applicazione senza nasconderne le funzionalità di base o il codice sorgente. Implementare la sicurezza nascondendo le funzioni del sistema ritenute vulnerabili è da considerarsi un controllo di sicurezza debole, e quasi sempre fallisce quando questo è l'unico controllo messo in campo. Ciò non significa che mantenere la segretezza sia sbagliato, significa semplicemente che la sicurezza di quei sistemi ritenuti chiave non deve dipendere esclusivamente dal fatto che ne vengono tenuti nascosti i dettagli. Ad esempio, la sicurezza di un'applicazione non dovrebbe basarsi sulla segretezza del suo codice sorgente. Questa dovrebbe basarsi su numerosi altri fattori, tra cui politiche adeguate in materia di password, difesa in profondità, limitazione delle transazioni finanziarie o comunque critiche, architettura di rete solida, controlli antifrode e di audit. Un esempio pratico può essere considerato il sistema operativo Linux. Il codice sorgente di Linux è ampiamente disponibile, eppure, se adeguatamente protetto, può essere considerato un sistema operativo sicuro e robusto.

12) Mantenere i controlli di sicurezza semplici e chiari.
   Superficie d'attacco e semplicità vanno di pari passo. Alcune tendenze dell'ingegneria del software preferiscono approcci eccessivamente complessi alla semplicità e alla chiarezza del codice. Chi sviluppa dovrebbe evitare l'adozione di architetture complesse quando un approccio più chiaro e lineare sarebbe più rapido e semplice da proteggere. Quando si pensa alla sicurezza del software, la prima domanda che ci pone è: "In quanti modi diversi il nostro software potrebbe essere attaccato? Ovvero, quante "vie d'accesso" esistono nella nostra applicazione? E' un po' come chiedere "Quante porte e finestre ci sono in un edificio? Se l'edificio ha solo una porta esterna, è molto facile proteggere quella porta. Se diversamente, si dispone di mille porte, allora sarà quasi impossibile mantenere sicuro l'edificio, non importa quanto siano resistenti le porte o di quante guardie di sicurezza si dispone. E' importante dunque limitare le "vie d'accesso" al software ad un numero ragionevole, altrimenti questo non sarà mai sicuro. Ciò lo si ottiene rendendo l'intero sistema relativamente

semplice, o scomponendolo in componenti molto semplici e totalmente separati. Una volta limitate le vie d'accesso, è necessario iniziare a pensare a "Quanti attacchi diversi sono possibili contro ciascuna via d'accesso? Anche in questo caso, è opportuno limitare il tutto rendendo le vie di per sé piuttosto semplici. Come una porta con una chiave unica, invece di una porta che può essere aperta con cinque chiavi diverse. Una volta fatto ciò, è necessario contenere i possibili danni che un attacco potrebbe arrecare se portato con successo. Ritornando all'esempio dell'edificio, dovremmo fare in modo che una singola porta consenta l'accesso ad una sola stanza. Il fatto di aumentare la complessità del codice determina con buona probabilità una crescita del livello di entropia e di conseguenza un proporzionale incremento dell'esposizione del software a possibili vulnerabilità. La complessità rende la sicurezza facile da ignorare e quindi aumenta la probabilità che questa possa fallire. All'atto pratico:

- Riutilizzare quei componenti di cui è nota l'affidabilità.
- Evitare architetture complesse e codifiche che presentano l'uso di doppi negativi, es: un costrutto condizionale del tipo "if(!item.isNotFound())" dovrebbe essere ricondotto nella sua forma semplice "if(item.isFound())".

13) Risolvere nel modo corretto le problematiche di sicurezza.
Una volta che un problema di sicurezza viene identificato, è importante attuare un test e comprendere la causa che origina il problema al livello di massima 'profondità'. Quando si utilizzano schemi predefiniti di progettazione, è probabile che eventuali problematiche di sicurezza presenti su un modulo vengano diffuse in tutte le baseline di codice, per cui è essenziale sviluppare la giusta risoluzione senza introdurre regressioni. Per esempio, un utente ha scoperto di poter vedere le informazioni di un altro utente manipolando il proprio cookie. La correzione sembra essere relativamente semplice, ma poiché il codice di gestione dei cookie è condiviso tra tutte le applicazioni, una modifica ad una sola applicazione può essere estesa a tutte le altre. La correzione, ed il contesto su cui opera (una correzione può avere effetti diversi su implementazioni differenti) deve quindi essere verificata su tutte le applicazioni interessate.

5.2.2 Pratiche di secure design

5.2.2.1 Best practice di secure design per le applicazioni web

Il seguente elenco definisce, a titolo esemplificativo e non esaustivo, un insieme di buone pratiche che, se adottate in fase di design, consentono di risolvere a priori vulnerabilità applicative note considerate maggiormente critiche.

- Gestione degli errori e attività di logging
  - Mostrare all'utente finale messaggi di errore di carattere generico.
    - Descrizione: I messaggi di errore non devono mai rivelare dettagli sullo stato interno dell'applicazione. Ad esempio, percorsi del file system e informazioni sullo stack non devono essere mai esposte all'utente attraverso messaggi di errore. (Vedi CWE 209).
  - Gestire tutte le eccezioni nel codice sorgente.
    - Descrizione: Non consentire mai che si verifichi un'eccezione non gestita attraverso i linguaggi e i framework impiegati nello sviluppo di applicazioni web. La gestione degli errori deve essere implementata in modo tale da comprendere anche gli errori imprevisti restituendo l'output all'utente sempre in modo controllato. (Vedi CWE 391).
  - Nascondere, bloccare o gestire gli errori generati da eventuali framework o software di terze parti in uso.
    - Descrizione: L'uso di un eventuale framework o piattaforma di sviluppo può generare messaggi di errore predefiniti. Questi dovrebbero essere bloccati o

sostituiti con messaggi di errore personalizzati, in quanto tali messaggi potrebbero rivelare informazioni sensibili all'utente. (Vedi CWE 209).

- Registrare su log tutte le attività di autenticazione e validazione.
  - Descrizione: Registrare su log tutte le attività di autenticazione e di gestione delle sessioni eseguite lato software insieme a tutti gli errori di convalida dell'input. Tutti gli eventi relativi alla sicurezza devono essere registrati. Questi possono essere utilizzati successivamente per rilevare attacchi passati o in corso. (Vedi CWE 778).
- Registrare su log tutte le attività di cambiamento dei privilegi.
  - Descrizione: Registrare su log tutte le attività o le occorrenze in cui il livello di privilegi di un utente subisce un cambiamento. (Vedi CWE 778).
- Registrare su log tutte le attività amministrative.
  - Descrizione: Registrare su log tutte le attività eseguite a livello amministrativo sull'applicazione o su uno qualsiasi dei suoi componenti. (Vedi CWE 778).
- Registrare su log qualsiasi accesso a informazioni sensibili.
  - Descrizione: Registrare su log qualsiasi accesso ai dati sensibili. Ciò è particolarmente importante per quelle organizzazioni che devono soddisfare i requisiti normativi come HIPAA, PCI o SOX. (Vedi CWE 778).
- Non registrare su log dati inappropriati o non necessari.
  - Descrizione: Sebbene la registrazione su log degli errori e l'accesso per l'auditing siano importanti, i dati sensibili non dovrebbero mai essere registrati in forma non cifrata. Ad esempio, sotto HIPAA e PCI, costituirebbe una violazione del log in termini di dati sensibili in esso presenti, a meno che il log non sia criptato sul disco. Inoltre, potrebbe dar luogo ad un pericoloso punto di esposizione nel caso in cui l'applicazione stessa venga compromessa. (Vedi CWE 532).
- Persistere i dati di log in modo sicuro.
  - Descrizione: I log devono essere conservati e mantenuti in modo appropriato per prevenire eventuali perdite di informazioni o la possibile compromissione da parte di intrusi. La conservazione dei log deve inoltre rispettare le politiche di conservazione stabilite dall'organizzazione al fine di soddisfare i requisiti normativi e fornire le sufficienti informazioni necessarie per le attività forensi e di risposta agli incidenti. (Vedi CWE 533).

- Protezione dei dati
  - Prevedere ovunque l'HTTPS per applicazioni web.
    - Descrizione: L'HTTPS dovrebbe essere impiegato preferibilmente in tutti i punti di accesso all'applicazione web. Se è necessario limitarne l'uso, quantomeno lo si deve prevedere per le pagine di autenticazione e tutte le pagine a valle dell'autenticazione. Se è previsto l'invio di informazioni sensibili (ad esempio, informazioni personali) prima dell'autenticazione, anche queste devono essere necessariamente trasmesse tramite HTTPS. Utilizzare sempre se disponibile, la versione HTTPS prevista dal server indicata attraverso l'URL. Consentire il re indirizzamento da HTTP a HTTPS aumenta la possibilità per un possibile attaccante di perseguire un attacco man-in-the-middle senza destare il minimo sospetto dell'utente. (Vedi CWE 311, 319, 523).
  - Disattivare l'accesso tramite HTTP a tutte le risorse protette.
    - Descrizione: Per tutte le pagine che richiedono l'accesso protetto tramite HTTPS, lo stesso URL non deve essere accessibile tramite canale HTTP. (Vedi CWE 319).
  - Impiegare una configurazione forte del TLS.
    - Descrizione: Disattivare su tutti i server eventuali schemi di cifratura considerati deboli. Ad esempio, i protocolli SSL v2, SSL v3 e TLS precedenti alla versione 1.2 presentano debolezze note e non sono considerabili sicuri. Disattivare inoltre, le suite di cifratura NULL, RC4, DES e MD5. Assicurarsi che la lunghezza di tutte le chiavi

sia superiore a 128 bit, utilizzare una rinegoziazione sicura e disabilitare la compressione se presente.

- Utilizzare l'intestazione "Strict-Transport-Security".
  - Descrizione: L'intestazione Strict-Transport-Security garantisce che il browser non parli con il server tramite HTTP. Ciò contribuisce a ridurre il rischio di attacchi di downgrade dell'HTTP.
- Memorizzare le password utente utilizzando un algoritmo forte di salted hashing iterativo.
  - Descrizione: Le password utente devono essere memorizzate utilizzando tecniche di hashing sicuro con algoritmi forti come PBKDF2, bcrypt o SHA-512. Il semplice hashing della password eseguito una sola volta non protegge a sufficienza la password stessa. Per rendere l'hash forte, utilizzare un algoritmo di hashing basato su funzione adattiva, combinato con un salt generato randomicamente per ciascun utente. (Vedi CWE 257).
- Scambiare in modo sicuro le chiavi crittografiche.
  - Descrizione: Se le chiavi di crittografia vengono scambiate o preimpostate nell'applicazione, allora qualsiasi impostazione o scambio di chiavi deve essere effettuato su un canale sicuro.
- Definire e configurare un processo sicuro di gestione delle chiavi crittografiche.
  - Descrizione: Quando le chiavi crittografiche sono memorizzate nel sistema, queste devono essere opportunamente protette e accessibili solo al personale autorizzato rispettando il criterio del "need-to-know". (Vedi CWE 320).
- Utilizzare certificati HTTPS validi rilasciati da una autorità certificante con buona reputazione.
  - Descrizione: I certificati HTTPS devono essere firmati da un'autorità di certificazione di buona reputazione. Il nome sul certificato deve corrispondere all'FQDN del sito web. Il certificato stesso deve essere valido e non scaduto.
- Disabilitare il data caching attraverso l'uso delle intestazione di controllo della cache e disabilitare anche l'auto completamento.
  - Descrizione: Disabilitare la cache dei dati del browser usando le intestazioni HTTP di controllo della cache o i meta tag presenti all'interno della pagina HTML. Inoltre, i campi di input che contengono dati sensibili, come quelli presenti nel form di login, dovrebbero avere nel relativo form HTML, l'attributo "autocomplete "impostato su "off" al fine di forzare il browser a non mettere in cache le credenziali. (Vedi CWE 524).
- Cifrare i dati sensibili persistiti.
  - Descrizione: Crittografare i dati sensibili o critici prima che questi vengano archiviati. (Vedi CWE 311, 312).
- Limitare l'utilizzo e la memorizzazione dei dati sensibili.
  - Descrizione: Condurre una appropriata valutazione al fine di accertarsi che i dati sensibili non vengano inutilmente trasportati o conservati. Ove possibile, utilizzare la tokenizzazione per ridurre i rischi di esposizione dei dati.

- Configurazione e operatività
  - Automatizzare il processo di distribuzione.
    - Descrizione: Con l'avvento dei processi di Continuous Integration e di Continuous Delivery Il ciclo di vita del software si integra sempre più con l'ambiente di produzione. Grazie a ciò, l'automazione del processo di distribuzione del software, garantisce che le modifiche vengano apportate in modo coerente e ripetibile in tutti gli ambienti.
  - Definire e attuare un processo rigoroso di "Change Management".
    - Descrizione: Nell'ambito dell'operatività, è opportuno mantenere un processo rigoroso di gestione dei cambiamenti. Ad esempio, le nuove release di software

dovrebbero essere distribuite solo dopo il completamento degli appropriati test e della relativa documentazione. (Vedi CWE 439).

- Definire gli opportuni requisiti di sicurezza.
  - Descrizione: Incaricare il business owner per la definizione dei requisiti di sicurezza del software. Ciò include gli aspetti che vanno dalle regole di convalida basate su whitelist fino ai requisiti non funzionali, come le prestazioni di una funzione specifica del sistema. La definizione anticipata di tali requisiti garantisce che la sicurezza sia integrata nel sistema sin dall'inizio.
- Condurre una Design Review.
  - Descrizione: L'integrazione delle pratiche di sicurezza nella fase di progettazione consente di risparmiare tempo e denaro. Condurre un'analisi dei rischi con i professionisti della sicurezza (ad esempio attraverso il tool di Risk Management fornito da AGID) e modellare l'applicazione per identificare i rischi maggiormente rilevanti. Ciò consente di integrare le appropriate contromisure nella progettazione e nell'architettura dell'applicazione software. (Vedi CWE 701, 656).
- Eseguire una Code Review.
  - Descrizione: La revisione del codice incentrata sugli aspetti di sicurezza possono rappresentare uno dei mezzi più efficaci per individuare i bug di sicurezza. E' importante revisionare regolarmente il codice alla ricerca di problematiche comuni di sicurezza come SQL Injection e Cross-Site Scripting. Avvalersi degli strumenti automatizzati per massimizzare l'ampiezza della copertura e la coerenza dei findings. (Vedi CWE 702).
- Eseguire gli opportuni test di sicurezza.
  - Descrizione: Condurre dei test di sicurezza sia durante che dopo lo sviluppo al fine di garantire che l'applicazione soddisfi gli standard di sicurezza imposti. Tali test dovrebbero essere condotti anche dopo le major release per garantire che eventuali vulnerabilità non siano state introdotte durante il processo di aggiornamento. Avvalersi dell'automazione includendo i test di sicurezza nella pipeline CI/CD.
- Hardenizzare l'infrastruttura.
  - Descrizione: Tutti i componenti dell'infrastruttura a supporto dell'applicazione devono essere configurati secondo le migliori pratiche di sicurezza e le linee guida di hardening. In una tipica applicazione web ciò può includere router, firewall, switch di rete, sistemi operativi, server web, application server, database e framework applicativi. (Vedi CWE 15, 656).
- Definire un piano di gestione degli incidenti.
  - Descrizione: E' opportuno elaborare e regolarmente verificare un piano di gestione degli incidenti. Si deve ben definire e mantenere aggiornato, l'elenco delle persone da coinvolgere in un incidente di sicurezza che riguarda l'applicazione software.
- Formare il team sugli aspetti della sicurezza.
  - Descrizione: La formazione aiuta a definire un linguaggio comune che il team può utilizzare per migliorare la sicurezza dell'applicazione software. La formazione non dovrebbe essere limitata esclusivamente agli sviluppatori di software, tester e architetti. Chiunque sia coinvolto nel processo di sviluppo, come gli analisti funzionali e i project manager, dovrebbe essere soggetto a formazione periodica sulla sicurezza del software.

5.2.2.2 Best practice di secure design per il cloud

In questo paragrafo, vengono sinteticamente descritte alcune delle migliori pratiche da adottare in fase di design, per fronteggiare le vulnerabilità e i rischi che potrebbero nascere a seguito dello spostamento di applicazioni e dati verso il cloud.

Tali pratiche si rivolgono a tutte le organizzazioni, indipendentemente dalle dimensioni, che intendono migliorare la sicurezza dei propri servizi in cloud. Si fa notare che queste best practice non sono da intendersi esaustive e che devono essere integrate con le linee guida rilasciate dai fornitori di servizi cloud (CSP), con le best practice generali di sicurezza informatica, con i requisiti di conformità normativa e con le pratiche definite dalle associazioni di categoria del cloud (ad esempio, la Cloud Security Alliance).

- Utilizzo non sicuro delle credenziali di sviluppo
  - Le credenziali dello sviluppatore consentono al team e ad eventuali integrazioni di accedere all'account cloud. Queste dovrebbero essere conservate e utilizzate in modo sicuro per garantire che solo le persone e i casi d'uso autorizzati ne abbiano accesso. Se possibile, prevedere un monitoraggio adeguato e una scadenza automatica dopo un determinato periodo di tempo o di inattività delle credenziali.
- Storage pubblicamente accessibili
  - I provider cloud normalmente prevedono diversi metodi di memorizzazione di oggetti e dati. E' importante revisionare regolarmente le relative configurazioni al fine di garantire che solo i componenti previsti siano pubblicamente accessibili.
- Uso improprio della configurazione di default
  - I provider cloud di solito pre-configurano i criteri di default di controllo degli accessi. Tale impostazione a volte può essere utile, ma spesso introduce dei rischi in quanto le offerte di servizi del fornitore spesso sono soggette a cambiamento. Quindi, le regole preconfigurate spesso vengono modificate per consentire l'accesso ai nuovi servizi che sono fuori del contesto di ciò che è di fatto necessario o in uso.
- Compromissione delle regole di controllo accesso
  - Quando si progetta l'accesso ad un servizio in cloud è buona norma seguire sempre il principio del minimo privilegio. E' necessario considerare la granularità dell'accesso ai servizi, ai sistemi e alla rete. Pertanto è opportuno revisionare regolarmente o automaticamente tali tipologie di accesso al fine di garantire che il principio del minimo privilegio venga rispettato.
- Costrutti di rete non configurati correttamente
  - La maggior parte dei provider cloud dispone di metodi per controllare l'accesso alla rete che vanno al di là delle semplici regole basate sugli indirizzi IP. Considerare l'utilizzo di tali costrutti per controllare l'accesso a livello granulare. Considerare l'utilizzo di componenti di rete basti su provider cloud per segmentare il traffico in modo ponderato.
- Monitoraggio e Logging inadeguato
  - Attivare e monitorare regolarmente la registrazione su log degli accessi attraverso API. Adottare una strategia di logging basata sul rischio per quei servizi che non vengono loggati attraverso i core logging services.
- Mancanza di un Inventory Management
  - L'accesso basato su API risolve molti problemi di inventory management. Adottare le opportune strategie al fine di arricchire l'ambiente con informazioni aggiuntive su proprietà, casi d'uso e sensibilità.
- Dirottamento del dominio
  - Spesso esiste un rapporto di fiducia transitoria tra i servizi cloud e le entries DNS. Revisionare regolarmente le configurazioni DNS e cloud per prevenire situazioni di acquisizione di controllo fraudolento.
- Mancanza di un piano di Disaster Recovery
  - Gli ambienti cloud non risolvono automaticamente i problemi di Disaster Recovery. Considerare l'appropriato livello di investimento per gli eventi catastrofici che avvengono all'interno dell'ambiente cloud. Progettare un programma di Disaster Recovery per il ripristino abilitato da account esterni, provider o da locale.
- Configurazione manuale degli account

- Gli interventi manuali limitano la capacità di scalare e sfruttare al meglio gli strumenti e i controlli di sicurezza nativi del cloud. Considerare il software sviluppato in ottica di sicurezza e l'automazione come i migliori alleati in ambito cloud.

5.2.2.3 Best practice di secure design per le architetture serverless

Il serverless computing, noto anche come function as a service (FaaS), è una tendenza emergente nello sviluppo di applicazioni web che offre numerosi vantaggi rispetto all'architettura tradizionale basata su cloud: riduzione dei costi operativi, semplificazione dei processi, maggiore flessibilità, scalabilità e distribuzione più rapida. Tuttavia, per quanto vantaggioso, il serverless computing è comunque vulnerabile alle minacce alla sicurezza proprio come i servizi basati sul cloud, anche se in modi diversi. Da una parte, una architettura di questo tipo, riduce i rischi che coinvolgono i server, come ad esempio attacchi DDoS (distributed-denial-of-service). Elimina inoltre la necessità per gli sviluppatori di gestire le patch del sistema operativo. D'altro lato, il serverless computing solleva una nuova serie di problematiche di sicurezza che consentono agli hacker di eseguire azioni dannose sulle applicazioni. Tutte le funzioni serverless sono prive di stato, il che significa che i dati sensibili, comprese le sessioni utente, vengono trasmessi e memorizzati in una posizione esterna. Tale configurazione inevitabilmente aumenta il rischio di perdite di dati quando questi vengono spostati all'esterno del server.
Pertanto è opportuno disporre all'interno dell'organizzazione di controlli e pratiche di sicurezza da attuare in ambito serverless. Di seguito alcune pratiche utili ad incrementare il livello di sicurezza in tale ambito.

- Funzione di iniezione di dati relativi ad un evento
  - Le architetture serverless costituiscono una moltitudine di fonti di eventi, che possono innescare l'esecuzione di una funzione serverless. Tali funzioni possono elaborare input proveniente da qualsiasi tipologia di fonte di evento, e tale input può includere diversi formati di messaggio, a seconda del tipo di evento e della fonte di provenienza. Le parti costituenti di tali messaggi possono contenere dati controllati da possibili aggressori o comunque dannosi.
- Compromissione del processo di autenticazione
  - Le architetture Serverless promuovono una progettazione di sistema orientata ai microservizi e sono costituite da funzioni che vengono combinate e orchestrate per realizzare la logica di business complessiva del sistema. Alcune funzioni serverless possono esporre web API pubbliche, mentre altre possono consumare eventi provenienti da diversi tipi di fonti, come eventi di cloud storage, eventi di database NoSQL, segnali di telemetria generati dai dispositivi IoT o anche messaggi di notifica SMS. Pertanto è opportuno adottare schemi di autenticazione forti, capaci di fornire l'adeguato livello di controllo degli accessi e di protezione, a tutte le funzioni maggiormente rilevanti, i tipi di eventi e i trigger.
- Configurazione di deployment non sicura in ambito Serverless
  - I servizi cloud in generale, e le architetture serverless in particolare, per essere adattate ad ogni specifica esigenza, compito o ambiente circostante, offrono numerose personalizzazioni e impostazioni di configurazione. Alcune di queste impostazioni hanno implicazioni critiche sulla postura di sicurezza generale dell'applicazione e come tali, devono essere tenute in considerazione. Pertanto, non affidarsi alle impostazioni predefinite fornite dai fornitori di architetture serverless.
- Permessi di funzioni e ruoli eccessivamente privilegiati
  - Le applicazioni Serverless dovrebbero sempre seguire il principio del "minimo privilegio". Ciò significa che ad una funzione serverless dovrebbero essere concessi solo quei privilegi, che sono essenziali per eseguire la logica prevista. In un sistema in

cui tutte le funzioni condividono lo stesso set di permessi sovra-privilegiati, la presenza di una vulnerabilità in una singola funzione può degenerare in una catastrofe di sicurezza a livello di sistema.

- Monitoraggio e Logging di funzioni inadeguato
  - Rafforzare le impostazioni base o di default di logging per fornire un audit trail completo degli eventi di sicurezza. Tale configurazione dovrebbe includere elementi quali l'uso riuscito/fallito della chiave di accesso API, tentativi di invocazione di funzioni serverless con autorizzazioni inadeguate, distribuzione riuscita/fallita di nuove funzioni o configurazioni serverless, modifiche delle autorizzazioni di funzioni o dei ruoli di esecuzione, interazioni anomale o flussi irregolari tra funzioni serverless, connessioni in uscita avviate da funzioni serverless ed esecuzione di funzioni serverless o accesso ai dati da parte di un account esterno di terze parti non correlato al main account.
- Dipendenze non sicure da terze parti
  - Definire un processo per mantenere aggiornato un inventario riportante i pacchetti software e altre dipendenze con le relative versioni, scansionare il software alla ricerca di dipendenze vulnerabili note, rimuovere le dipendenze inutili, aggiornare le versioni dei pacchetti deprecate alle versioni più recenti e applicare tutte le patch software necessarie.
- Memorizzazione non sicura delle informazioni sensibili trattate dall'applicazione
  - Memorizzare tutti i dati sensibili trattati dalle applicazioni in un archivio crittografato sicuro e garantire che le chiavi crittografiche siano mantenute attraverso un'infrastruttura o servizio di gestione centralizzata. Tale tipologia di servizi sono normalmente disponibili nella maggior parte delle soluzioni architetturali serverless e cloud di mercato, che forniscono agli sviluppatori anche API sicure da poter utilizzare per integrarsi facilmente e perfettamente con gli ambienti serverless.
- Indisponibilità del servizio ed esaurimento delle risorse finanziarie
  - I vendors di soluzioni architetturali serverless definiscono i limiti predefiniti per l'esecuzione delle funzioni serverless. A seconda del tipo di limite e di attività, eventuali applicazioni mal progettate o configurate possono essere violate in modo tale da rendere la latenza nella risposta inaccettabile o addirittura per alcuni utenti inutilizzabile. Inoltre, un aggressore può spingere l'applicazione serverless per lunghi periodi di tempo, ad un "sovrautilizzo", causando essenzialmente un incremento della fatturazione mensile e infliggendo una significativa perdita finanziaria per l'organizzazione destinataria.
- Manipolazione o compromissione del flusso di esecuzione di una funzione Serverless
  - La possibilità di poter manipolare il flusso applicativo può aiutare eventuali aggressori a sovvertire la logica applicativa. Attraverso questa tecnica, un attaccante può a volte bypassare i controlli di accesso, elevare i privilegi dell'utente o persino sferrare un attacco di Denial of Service. In un sistema in cui esistono una moltitudine di funzioni e ciascuna di queste può invocarne un'altra, l'ordine di invocazione può essere critico nel raggiungimento della logica desiderata. Inoltre, la progettazione dell'applicazione potrebbe presumere che alcune funzioni siano invocate solo in scenari specifici e solo da invocazioni autorizzate. Assicurarsi che per ciascuna funzione vengano impostati gli adeguati permessi e controlli di accesso e, lì dove applicabile, impiegare una solida infrastruttura di gestione dello stato dell'applicazione.
- Gestione impropria degli errori e eccessiva verbosità dei messaggi di errore
  - Le possibilità di eseguire il debug linea per linea delle applicazioni serverless sono piuttosto limitate e di maggiore complessità rispetto alle classiche opzioni di debug

disponibili per lo sviluppo di applicazioni standard. Ciò costringe alcuni sviluppatori ad adottare l'uso di messaggi di errore con elevata verbosità, ad abilitare le variabili dell'ambiente di debug e a volte a dimenticare di ripulire il codice quando lo si sposta nell'ambiente di produzione. I messaggi di errore particolarmente verbosi come lo stack trace o gli errori di sintassi, che vengono esposti agli utenti finali, possono rivelare dettagli sulla logica interna della funzione serverless, e a sua volta rivelare potenziali debolezze, difettosità o addirittura dare luogo a perdite di dati sensibili. Se l'ambiente serverless consente di definire messaggi di errori personalizzati, come nel caso degli API gateway, è opportuno generare messaggi di errore semplici che non rivelano dettagli implementativi o il contenuto di eventuali variabili di ambiente.

1.1.1.3. Best practice di secure design per le architetture basate su registri distribuiti (DLT)

I registri distribuiti (DLT o più notoriamente Blockchain) sono sistemi informatici che gestiscono dati, transazioni o codici eseguibili (Smart Contracts) in modo il più possibile indipendente da un'autorità centrale attraverso l'utilizzo di data storage distribuito in correlazione con processi crittografici e sistemi decisionali decentralizzati. Le DLT sono in rapido sviluppo per le più svariate applicazioni, da nuovi sistemi di identità digitale (Self Sovereign Identity) a sistemi di certificazione garantita di dati e certificazioni (Verifiable Claims), alla creazione e gestione di nuovi mercati digitali (vedi i mercati peer to peer di scambio energetico) fino alla creazione e gestione di nuove entità (vedi ad esempio le DAO/DAC Distributed Autonomous Organizations/Corporations). In questo documento evitiamo volutamente di menzionare le criptovalute, da cui le DLT sono nate, ma che esulano dal contesto di questa analisi.
Queste caratteristiche delle DLT hanno permesso la gestione di applicazioni in cui differenti partecipanti, con obiettivi in conflitto, possono comunque operare di concetto attraverso processi predefiniti ed immodificabili.
Bisogna comunque tenere in considerazione che questa classe di tecnologie annovera strumenti con design alquanto differenti, sia per la gestione die permessi di lettura/scrittura (permissioned/unpermissioned DLT) che per la specifica implementazione del processo di maggiore criticità, cioè quello della gestione del consenso.
I concetti espressi in questo paragrafo prevedono una buona conoscenza della tecnologia.
È fondamentale prima di tutto valutare la tematica dei dati memorizzati in una DLT relativamente all'integrità, alla disponibilità ed alla riservatezza.
Affrontando per primo il tema dell'integrità dei dati, uno dei fattori fondamentali che assicurano tale caratteristica in una DLT è la struttura a blocchi concatenati attraverso la funzione di hash. Ma questa di per sé non basterebbe se non fosse affiancata da un adeguato algoritmo di consenso che garantisca l'immediata evidenza di una qualsivoglia modifica di questa struttura in confronto a strutture uguali esistenti nei vari nodi componenti la DLT.
Tenendo conto che la struttura a blocchi amplifica la sicurezza della funzionalita' di hash attraverso la concatenazione (cioe' la modifica di un blocco in posizione <n> implicherebbe la conseguente modifica di tutti i blocchi successivi fino alla testa della catena) il fattore critico da considerare relativamente all'integrità dei dati è chiaramente quello dato dall'algoritmo di consenso. Riportiamo quindi la tematica alla sezione relativa alla sicurezza di tale algoritmo.
In una DLT la disponibilità dei dati è intrinsecamente garantita dall'infrastruttura sottostante: difatti ogni nodo di una DLT (a meno di nodi specifici, normalmente chiamati 'light nodes') detiene una intera copia della struttura dati e quindi l'indisponibilita' degli stessi si potrebbe avere solo nel caso in cui tutti i nodi fossero allo stesso istante inattivi. Attacchi che mirino ad un eventuale centro non avrebbero senso in una struttura decentralizzata.

Un possibile attacco sulla disponibilità quindi dovrebbe essere mirato all'intera infrastruttura e quindi avere ad esempio come obiettivo il software di base, al fine di rendere indisponibili le informazioni negandone l'accesso attraverso la modifica della modalita' con cui tutti i nodi operano.
In una DLT la riservatezza dei dati varia ampiamente a seconda della specifica tecnologia utilizzata: alcune DLT sono intrinsecamente trasparenti (vedi bitcoin) in quanto chiunque ha accesso a tutte le transazioni fatte da qualsivoglia partecipante e la confidenzialità è demandata all'anonimità dei partecipanti che vengono identificati da un indirizzo generico. Il problema è che, una volta questo indirizzo venga correlato ad un utente, tutte le transazioni di questo divengono immediatamente visibili.
Alcune DLT sono invece orientate all'anonimità dei partecipanti e delle transazioni (es. zcash, monero) ed ad un osservatore esterno è praticamente impossibile risalire agli attori parte di una qualsiasi transazione proprio per le tecnologie utilizzate (es. ring transactions in monero).
In qualsiasi caso è necessario provvedere ad una adeguata gestione della riservatezza dei dati in modo disgiunto dalla gestione dell'infrastruttura: il livello di riservatezza da implementare sarà come al solito legato alla ripologia dei dati e le metodologie le stesse applicate a qualsiasi altra tipologia di data storage.

Analizziamo qui di seguito la sicurezza delle DLT, approcciando i vari elementi fondamentali che le compongono:

1. Rete infrastrutturale peer-to-peer
2. Struttura dati (concatenazione di blocchi)
3. Algoritmi di consenso (e.g. PoW, PoS, etc)
4. Smart contracts e logiche dinamiche (e.g. token, DAC/DAO, etc)

Riguardo l'infrastruttura, un possibile attacco di Distributed Denial of Service potrebbe provenire dall'utilizzo di wallet/accounts per generare grandi numeri di transazioni al fine di rallentare la rete. Un evento del genere, anche se non rappresentante un attacco, è avvenuto nella rete ethereum all'apparire dello smart contract CryptoKitties: questo smart contract ha raggiunto volumi superiori al 10% dell'intero volume di transazioni della rete, causando un indesiderato rallentamento dell'intera rete.
Nel mese di Marzo 2016 la rete Bitcoin si è quasi arrestata a causa di un wallet che generava larghi volumi di transazioni con un costo di transazione piu' elevato della media: i minatori, prioritizzando questo wallet piu' remunerativo, tendevano ad ignorare altre transazioni nella generazione di nuovi blocchi.
La struttura dati è correlata all'algoritmo di consenso che decide l'introduzione di nuove transazioni in coda alla catena, studiamo quindi queste due tematiche in modo congiunto.
Riguardo l'algoritmo di consenso, dobbiamo tenere prima conto del teorema CAP, che indica che è impossibile per una DLT garantire contemporaneamente piu' di due dei seguenti elementi:

- *Consistenza:* una DLT è consistente quando i fork sono evitati, caratteristica detta anche '*finalizzazione del consenso'* . In pratica i nodi devono tutti avere la stessa copia del ledger nello stesso momento
- *Disponibilita':* una DLT è disponibile se le transazioni generate dai client sono gestite e quindi committate (cioe' aggiunte alla catena di blocchi)
- *Tolleranza alle partizioni*: quando una partizione della rete avviene, i nodi autoritativi sono divisi in gruppi disgiunti affinche' i nodi di un gruppo non possono comunicare con i nodi di un altro gruppo

L'imposizione delle proprieta' suddette è responsabilita' dell'algoritmo di consenso, tale da garantire il funzionamento dell'intero sistema ed è compito di chi definisce l'architettura di decidere quali caratteristiche siano fondamentali per lo specifico caso d'uso.

Due forme di inconsistenza possono avvenire nell'algoritmo di consenso lasciando blocchi validi al di fuori di una DLT: la prima forma è quella dello "stale block", in cui un blocco minato con successo non viene accettato dall'attuale bestBlockchain (cioè la chain più complessa da ricreare): gli staleblock avvengono più frequentemente nelle blockchain pubbliche a causa delle consizioni di competizione tra i minatori (raceconditions).
L'altra forma di inconsistenza è basata sugli "orphaned block", cioè blocchi in cui il blocco padre ha un hash che punta ad un blocco non autentico.

Alcuni attacchi di cui possono essere vittime le DLT:

1. Il block withholding attack (BWH) ha come obiettivo le mining pools e fa in modo che il Sistema di reward delle stesse pool non sia più corretto, gratificando alcuni partecipanti al pool (che normalmente sono anche gli attaccanti) ricevere dei premi non proporzionali con il lavoro di mining effettuato.
2. L'attacco denominato 'selfish mining' prevede che un attaccante possa guadagnare dei premi non proporzionali con il lavoro di mining effettuato generando deliberatamente dei fork.
3. L'attacco denominato 'Fork after withholding' (FAW), similarmente al selfish mining, utilizza false fork: a differenza però il reward di un attaccante FAW è sempre maggiore od uguale a quello di un attaccante BWH ma è utilizzabile fino a 4 volte più spesso.
4. Il famoso attacco del 51% (principalmente orientato al consenso PoW) prevede che un gruppo di nodi con più del 51% della capacità di minare possa decidere in ogni caso qual è la prossima transazione indipendentemente dalla realtà (da considerare che un simile attacco su altri algoritmi quali il PBFT potrebbe essere portato anche con il 33% dei nodi).
5. L'attacco denominato 'eclipse attack' prevede un attacco non a tutta la rete ma solo ad uno o pochi nodi. L'attaccante isola il nodo e poi introduce transazioni malevole facendo in modo che lo stesso nodo non si accorga della non-sincronizzazione con i suoi peer.
6. L'attacco denominato 'spatial partitioning' prevede l'isolamento di un autonomous system che gestisce uno o più sistemi di mining e ridurre il mining power globale: questo attacco è portato in generale per facilitare altri attacchi attraverso proprio la riduzione del mining powrer
7. L'attacco denominato 'consensus delay' è associato con la natura P2P delle blockchain: in questo attacco si iniettano blocchi falsi con lo scopo di incrementare la altenza e quindi prevenire i nodi dal raggiungere un consenso sullo stato della DLT.
8. L'attacco denominato 'Finney Attack' prevede che il minatore possa minare un blocco che contenga una delle proprie transazioni e mantenerlo nascosto (stealth): c'è la possibilità di un double-spending nel cao in cui merchant accetti la transazione non confermata. A seguire il blocco nascosto viene pubblicato prima che la transazione venisse confermata dalla rete.

Non si vuole dare qui un resoconto esaustivo di tutti gli attacchi ma sottolineare le tipologie degli stessi e quindi le azioni preventive da tenere in considerazione a riguardo.
Bisogna in ogni caso tenere conto che la maggioranza degli attacchi è orientata verso le applicazioni (smart contracts) che girano sulla DLT e non sulla chain stessa: ci si deve quindi concentrare sulla qualità delle applicazioni che operano sulla DLT e prevedere per le stesse lo stesso ciclo di controllo di un qualsiasi software sviluppato come da linee guida AGID.

5.3 Threat Intelligence e Threat Modeling

Se l'obiettivo che ci si prefigge è quello di individuare gli attacchi a cui il software è correntemente esposto, allora il possesso di informazioni sulle minacce possono favorire nel porre l'attenzione sulle precise azioni necessarie che potrebbero essere intraprese nell'immediato. Diversamente, se l'obiettivo è ridurre la superficie di attacco e indirizzare gli investimenti in modo proattivo, allora la modellazione delle minacce può essere sicuramente di maggiore supporto. Quest'ultima non è in grado di fornire una risposta rapida al singolo problema di sicurezza che invece può essere data dalla Threat Intelligence, ma può sicuramente essere di aiuto nel guidare un programma maggiormente strategico che ha come obiettivo quello di elevare il livello di resilienza del software.

	Threat Modeling	**Threat Intelligence**
Finestra temporale	Proattivo	Reattivo
Estensione	Individuazione delle problematiche di sicurezza	Individuazione degli attaccanti
Supporto dal mercato	Consulenza e formazione	Feeds & Tools

Table 1 – Differenze tra Threat Modeling e Threat Intelligence

5.4 Threat Modeling e Threat Assessment

L'attività di Threat Assessment si concentra sull'identificazione delle minacce nelle applicazioni. Tale pratica è orientata all'individuazione e alla accurata comprensione di potenziali attacchi al software per recepire meglio i rischi e facilitarne la gestione. Difatti, la "software assurance" consiste nell'identificare i rischi presenti nelle applicazioni trattandoli quindi di conseguenza.
I rischi per un'applicazione possono essere relativi al business dell'applicazione (si pensi agli attacchi alla logica di business) o alla configurazione tecnica dell'applicazione. Lo stream del profilo di rischio dell'applicazione si occupa del primo, mentre il Threat Modeling si concentra sul secondo. Di seguito una sintesi dei livelli di maturità di un'organizzazione riguardo la valutazione delle minacce in relazione al profilo di rischio applicativo e all'attività di Threat modeling:

	Profilo di rischio applicativo	**Threat Modeling**
Livello di maturità 1 – Identificazione del Best-effort delle minacce di alto livello per l'organizzazione e per i singoli progetti.	Valutazione base del rischio applicativo	Modellazione delle minacce ad hoc del Best-effort
Livello di maturità 2 - Standardizzazione e analisi a livello aziendale delle minacce legate al software all'interno dell'organizzazione.	Comprensione del rischio per tutte le applicazioni dell'organizzazione	Modellazione delle minacce standardizzata
Livello di maturità 3 - Miglioramento proattivo della copertura delle minacce in tutta l'organizzazione	Revisione periodica dei profili di rischio dell'applicazione	Miglioramento della qualità grazie all'automazione del processo di analisi

Table 2 - Threat Assessment Overview

A seguire si descrivono in modo maggiormente dettagliato i singoli livelli di maturità sopra indicati, considerando che il tool di Risk Management di AGID può essere utilizzato per gestire questa tematica a tutti e tre i livelli, in base alla completezza delle informazioni gestite:

1) Profilo di rischio applicativo

- Livello di maturità 1
  - Benefici: capacità di classificare le applicazioni in base al rischio
  - Attività: come organizzazione, si vuole spendere il proprio budget nella sicurezza lì dove necessario. Il rischio applicativo è un valido strumento per guidare la spesa per la sicurezza. Una classificazione dei rischi aiuta a identificare quali applicazioni possono rappresentare una seria minaccia per l'organizzazione se queste venissero attaccate o violate. Adottare un metodo semplice per valutare il rischio applicativo per applicazione, stimando il potenziale impatto aziendale che essa rappresenta per l'organizzazione in caso di attacco. A tal fine, valutare l'impatto di una violazione della riservatezza, dell'integrità e della disponibilità dei dati o del servizio. Considerare l'utilizzo di un questionario di 5-10 domande per comprendere le caratteristiche importanti dell'applicazione, come ad esempio se l'applicazione tratta dati finanziari, se esposta a Internet o se sono coinvolti dati relativi alla privacy. Il profilo di rischio dell'applicazione indica se questi fattori sono applicabili e se possono avere un impatto significativo sull'organizzazione. Successivamente, utilizzare uno schema per classificare le applicazioni in base a tale rischio. Uno semplice schema qualitativo (ad es. alto/medio/basso) che traduce queste caratteristiche in un valore risulta essere spesso efficace. E' importante utilizzare tali valori per rappresentare e confrontare il rischio di applicazioni diverse l'una rispetto all'altra. Organizzazioni mature altamente orientate al rischio potrebbero utilizzare schemi di rischio maggiormente quantitativi. Non inventare un nuovo schema di rischio se l'organizzazione ne ha già uno e questo funziona bene. Valutate il rischio in base alla serie di domande e assegnare un livello di rischio ad ciascuna applicazione.
- Livello di maturità 2
  - Benefici: conoscenza solida del livello di rischio di un'applicazione
  - Attività: l'obiettivo di questa attività è quello di comprendere a fondo il livello di rischio di tutte le applicazioni presenti all'interno delle organizzazioni per concentrare gli sforzi dove è veramente importante. Dal punto di vista della valutazione del rischio, il questionario di base non è sufficiente per valutare a fondo il rischio di tutte le applicazioni. E' opportuno creare un modo ampio e standardizzato per valutare il rischio dell'applicazione, tra l'altro attraverso il loro impatto sulla sicurezza dell'informazione (riservatezza, integrità e disponibilità dei dati). Oltre alla sicurezza, si vuole anche valutare il rischio per la privacy dell'applicazione. Comprendere i dati che l'applicazione elabora e quali potenziali violazioni della privacy sono rilevanti. Infine, valutare l'impatto che l'applicazione ha su le altre applicazioni presenti all'interno dell'organizzazione (ad esempio, l'applicazione potrebbe modificare dati che sono stati considerati di sola lettura in un altro contesto). Valutare tutte le applicazioni all'interno dell'organizzazione, comprese quelle legacy. Considerare l'uso di schemi quantitativi per classificare il rischio applicativo. Un semplice schema qualitativo (ad esempio alto/medio/basso) non è sufficiente per gestire e confrontare efficacemente le applicazioni su scala Enterprise. Sulla base di tale input, costruire un inventario centralizzato dei profili di rischio che utilizza i risultati delle valutazioni del rischio per definire il profilo. Questo inventario fornisce a tutte le parti interessate una visione allineata del livello di rischio di un'applicazione per assegnare un'adeguata priorità alle attività legate alla sicurezza.
- Livello di maturità 3
  - Benefici: aggiornamento tempestivo della classificazione dell'applicazione in caso di modifiche

- Attività: Il portafoglio di applicazioni di un'organizzazione cambia, così come le condizioni e i vincoli in cui opera un'applicazione (ad esempio, guidato dalla strategia aziendale). Eseguire una revisione periodica dell'inventario dei rischi per garantire la correttezza delle valutazioni dei rischi delle diverse applicazioni. Operare tale revisione a livello aziendale. Inoltre, via via che l'organizzazione evolve e matura nella garantire la sicurezza del software, è importante incoraggiare i team a chiedersi continuamente quali cambiamenti nelle condizioni potrebbero avere un impatto sul profilo di rischio. Per esempio, un'applicazione interna potrebbe essere esposta a Internet a seguito di una decisione aziendale. Ciò dovrebbe indurre i team a ripetere la valutazione del rischio e ad aggiornare di conseguenza il profilo di rischio dell'applicazione. In un'implementazione matura di questa pratica, è altrettanto importante formare e aggiornare continuamente i team sulle esperienze e sulle migliori pratiche derivanti da queste valutazioni del rischio. Ciò porta ad una migliore esecuzione e ad una rappresentazione più accurata del profilo di rischio dell'applicazione.

2) Threat modeling
   - Livello di maturità 1
     - Benefici: comprensione di base delle potenziali minacce
     - Attività: lo scopo del Threat Modeling è quello di identificare in modo proattivo le potenziali problematiche presenti nella progettazione tecnica dell'applicazione. Una configurazione imprudente potrebbe dar luogo a importanti vettori di attacco in un'applicazione che può essere sfruttata per colpire l'organizzazione da cui viene realizzata o utilizzata. L'esperienza dimostra che la progettazione dell'architettura può essere una fonte rilevante di problemi di sicurezza e le conseguenze possono essere significative. La pratica della modellazione delle minacce include sia la raccolta che la gestione delle minacce. Per individuare le minacce e buona norma adottare quelle che sono riconosciute come buone pratiche di sicurezza o un approccio maggiormente strutturato come STRIDE. La modellizzazione delle minacce è spesso più efficace se eseguita da un gruppo di persone, prevedendo un brainstorming. Una delle sfide principali nella modellizzazione delle minacce è quella di riuscire ad ottenere, attraverso un esercizio di efficienza, un elenco contenuto di minacce rilevanti, evitando processi lunghi ed elenchi eccessivamente dettagliati di minacce di scarsa rilevanza. L'esperienza come sempre, aiuta a trovare il giusto equilibrio. Eseguire la modellazione delle minacce in modo iterativo per allinearsi ai paradigmi di sviluppo maggiormente iterativi. Se si aggiungono nuove funzionalità ad un'applicazione esistente, esaminare solo le nuove funzioni aggiunte piuttosto di cercare di coprire l'intero ambito. Eseguire la modellazione delle minacce su progetti importanti (vedi sopra: Application Risk Profile) in modalità best effort per identificare le minacce più importanti per l'applicazione. Ad esempio, un buon punto di partenza potrebbe essere quello di prendere nota degli schemi di rete esistenti descritti durante i vari workshop.
   - Livello di maturità 2
     - Benefici: miglioramento della raccolta e gestione delle minacce
     - Attività: stabilire un approccio standard per eseguire in modo strutturato la modellazione delle minacce e aumentare la qualità e l'efficienza di tale attività all'interno dell'organizzazione facendo sì che lo sforzo investito sia utile e ben speso. Una modellazione strutturata delle minacce tiene conto dei diversi attori, risorse e flussi per identificare un ampio spettro di potenziali minacce all'applicazione. Questa definisce gli input necessari per avviare l'attività (ad esempio, una descrizione di massima dell'architettura tecnica e un diagramma di flusso dei dati), i diversi passaggi per identificare le minacce e i formalismi per descrivere o annotare le

minacce. È possibile aggiungere al modello delle minacce gli opportuni controlli di mitigazione al fine di guidare il progettista nella gestione di particolari minacce. A livello di organizzazione, definire quali azioni devono essere innescate durante la fase di modellazione. Ad esempio, un cambiamento di architettura o la distribuzione di un'applicazione in un nuovo ambiente. Allo stesso tempo, riflettere su come attuare e supportare in modo scalabile il processo di modellazione delle minacce in tutta l'organizzazione. Notificare i risultati dell'attività di modellazione delle minacce al processo di gestione dei difetti del software per dare seguito ad un adeguato follow-up. Adottare un sistema di valutazione per misurare e confrontare la rilevanza delle diverse minacce riscontrate. Considerare l'utilizzo di uno strumento per la gestione dei modelli di minacce delle distinte applicazioni. Istruire gli addetti ai lavori a concentrarsi sulle minacce maggiormente significative, poiché una dei principali problemi nella modellazione delle minacce è quello di individuare un numero elevato di minacce poco significative. Gli strumenti offrono un notevole supporto nell' identificare le potenziali minacce ma, alla fine, la modellazione delle minacce richiede un'intelligenza umana che difficilmente può essere automatizzata.

- Livello di maturità 3
  - Benefici: aggiornamento tempestivo e gestione qualitativa delle minacce applicative
  - Attività: nell'ambito di un processo maturo di modellazione delle minacce, revisionare regolarmente (ad esempio, annualmente) i modelli di minacce esistenti al fine di verificare un eventuale presenza di nuove minacce rilevanti per le applicazioni analizzate. Utilizzare un processo di analisi automatizzato per valutare la qualità e individuare eventuali difettosità nei modelli delle minacce. Revisionare le categorie di minacce maggiormente rilevanti per l'organizzazione. Quando si identificano nuove categorie di minacce, informare tempestivamente l'organizzazione al fine di garantirne una appropriata gestione.

5.5 Modellazione e Individuazione delle minacce: Threat Modeling

I processi di sviluppo sicuro del software, analizzati nel capitolo precedente, prevedono la modellazione delle minacce e suggeriscono l'inclusione dell'attività di Threat modeling nella metodologia, al fine di migliorare la pratica di identificazione delle vulnerabilità in materia di sicurezza del Software.
La tecnica di modellazione e individuazione delle minacce si rivolge alle seguenti figure professionali:

- Progettisti e Architetti del software;
- Threat Modeling SME o Security Assessors, responsabili dell'analisi della sicurezza di tutti i componenti dell'intera applicazione.

Le informazioni adoperate per stabilire i requisiti di sicurezza necessari devono includere i principi di progettazione sicura, descritti a seguire, e valutati da un programma prestabilito di gestione delle vulnerabilità, che può anche richiedere l'intervento di terze parti interessate, come un team di conformità (ad esempio, se l'applicazione deve essere conforme a standard quali HIPAA, PCI, GDPR, ecc.) o un team di gestione e distribuzione, in quanto dove e come viene utilizzata l'applicazione può incidere sulle reali esigenze di sicurezza. Pertanto, prima di iniziare il processo di modellazione delle minacce, è importante identificare gli obiettivi di business delle applicazioni e identificare i requisiti di sicurezza e conformità. Ciò è molto importante da definire in anticipo al

fine di fornire l'adeguato supporto nella valutazione dell'impatto di qualsiasi vulnerabilità durante il processo di analisi dei rischi[13].
Ad alto livello, il processo dovrebbe prevedere:

1) l'identificazione delle minacce, dei rischi e dei requisiti di conformità a cui l'applicazione è soggetta;
2) l'individuazione degli adeguati requisiti di sicurezza capaci di fronteggiare le minacce e i rischi di cui sopra;
3) la comunicazione dei requisiti di sicurezza agli appropriati team operativi;
4) la convalida dell'attuazione di ciascun requisito di sicurezza;
5) l'audit, se necessario, al fine di dimostrare la conformità a qualsiasi politica o regolamento applicabile.

5.5.1 Introduzione e concetti base

Nella sua forma più elementare la modellazione delle minacce è un approccio strutturato che identifica le potenziali minacce alla sicurezza, valutandone il rischio e fornendo le più opportune contromisure.
La modellazione delle minacce comporta l'identificazione degli asset in un processo strutturato, individuando le possibili minacce che insistono su tali asset, categorizzandole e determinando le appropriate strategie di mitigazione.
Quando si approccia con la modellizzazione delle minacce, è importante avere una corretta comprensione della terminologia di base:

- **Asset**, è qualcosa di valore su cui un avversario pone particolare interesse. I dati presenti in un database sono un esempio di Asset.
- **Threat Agent**, un individuo o un gruppo di individui che possono manifestare una minaccia.
- **Superfice di attacco**, l'insieme dei diversi punti (i vettori di attacco) in cui un utente non autorizzato (l'aggressore) può tentare di introdurre dati in un ambiente o di estrarre dati da un ambiente.
- **Probabilità**, possibilità che si verifichi un evento ostile in cui un attore minaccioso sfrutta la presenza di una debolezza. La probabilità di eventi di minaccia che si traducono in impatti negativi determina la possibilità che un evento di minaccia si traduca in un risultato effettivo.
- **Impatto**, danno potenziale (fisico, logico, finanziario, ecc.) dovuto a un evento di minaccia.
- **Controllo**, protezione o contromisura per prevenire, rilevare, contrastare o ridurre al minimo i rischi per la sicurezza delle informazioni, dei sistemi informatici o di altri beni.
- **Mitigazione**, riduzione sistematica del rischio o dell'impatto su un asset.
- **Minaccia** (threat), è un evento che può o non essere dannoso all'origine ma che possa danneggiare o compromettere un'attività a seguito di un attacco.
- **Vulnerabilità**, è un difetto nella sicurezza di una o più parti di un sistema che rende possibile una minaccia.
- **Attacco**, è un tentativo da parte di un avversario di sfruttare una vulnerabilità.
- **Rischio**, è la probabilità di essere bersaglio di un attacco.
- **Contromisura**, è un'azione o uno strumento che contrasta una minaccia e mitiga il rischio.

La modellazione delle minacce può essere definita come la "revisione sistematica delle caratteristiche e dell'architettura dell'applicazione da un punto di vista della sicurezza". Tale processo fornisce un approccio strutturato per identificare e classificare le minacce basate sui componenti del software, sui flussi di dati e sui confini di fiducia (confini entro i quali esistono dei criteri di sicurezza).

[13] Per l'attività di Risk Assessment si deve far riferimento alla metodologia e al tool adottato da AGID a tale scopo (**Cyber Risk Management** - **https://www.sicurezzait.gov.it/Home**).

La modellazione delle minacce è una parte importante del ciclo di vita del software che identifica le minacce che potrebbero non essere riconosciute nelle tradizionali sessioni di brainstorming sulla sicurezza.
A differenza delle tecniche di test del software come il penetration testing o fuzzing, la modellazione delle minacce può essere eseguita durante la fase di progettazione del sistema rendendola totalmente indipendente dallo sviluppo del codice. Introdotto nel ciclo di vita dello sviluppo del software, il Threat Modeling può significativamente contribuire a garantire la sicurezza già nella fase di progettazione e disegno di un'applicazione, riducendo i costi e minimizzando le necessarie successive correzioni di sicurezza nel progetto. Anche i sistemi in essere possono beneficiare di tale processo. Possono essere identificate nel sistema le difettosità di sicurezza sconosciute o non risolte e può essere applicata una classificazione del rischio alle vulnerabilità individuate. Questo processo può anche essere adattato alle pratiche di sviluppo come Agile.

5.5.2 Motivazioni nell'uso del Threat Model

5.5.2.1 Ricerca preventiva dei bug di sicurezza

Si pensi alla costruzione di una casa; le decisioni prese all'inizio del progetto impatteranno drasticamente sulla sicurezza del prodotto finale. La scelta di pareti di legno e un consistente numero di finestre al piano terra, espongono la casa a maggiori rischi rispetto a una costruzione fatta in mattoni e con poche finestre. A seconda di dove si sta costruendo e di altri fattori, potrebbe anche essere una scelta ragionevole. Comunque, una volta effettuata la scelta, possibili futuri cambiamenti avrebbero sicuramente un impatto significativo sui costi. Certo, si possono sempre mettere delle inferriate sulle finestre, ma non sarebbe meglio concepire una soluzione più efficace e appropriata sin dall'inizio? È possibile applicare gli stessi tipi di compromessi alla tecnologia. La modellazione delle minacce aiuta a trovare problematiche di progettazione anche prima della stesura del codice e questa fase risulta essere il momento migliore per scoprire tali problemi.

5.5.2.2 Comprensione dei requisiti di sicurezza

Individuare le minacce e decidere come s'intende procedere, rende chiari i requisiti. Con i requisiti più chiari, è possibile dedicare maggior energia a un insieme consistente di funzionalità e proprietà di sicurezza. Esiste un'importante interazione tra requisiti, minacce e mitigazioni. Durante la fase di modellazione delle minacce, è possibile scoprire ad esempio, che alcune minacce non sono conformi ai requisiti aziendali e come tali potrebbero non essere prese in considerazione. Altresì i requisiti di sicurezza potrebbero dover tener conto di ulteriori minacce la cui risoluzione potrebbe però, essere troppo complessa e dispendiosa, pertanto, si dovrà scegliere tra l'indirizzare parzialmente tali minacce o accettarle comunicando che non è possibile affrontarle.

5.5.2.3 Ingegnerizzazione e rilascio di prodotti più sicuri

Considerando i requisiti e la progettazione effettuati all'inizio del processo, è possibile ridurre drasticamente le probabilità di dover ri-progettare o ricostruire il sistema, o manutenere un flusso costante di bug di sicurezza. L'effort risparmiato in tal senso potrebbe andare a favore di un processo di costruzione di un prodotto migliore, più veloce, più economico o più sicuro, lasciando spazio a una maggiore concentrazione nella fase realizzativa dei requisiti funzionali.

5.5.3 Processo di modellazione del sistema da proteggere

Il processo di modellazione delle minacce s'identifica in un insieme di passi che realizzano dei sotto-obiettivi, piuttosto che come una singola attività. Essenzialmente, le domande da porsi al fine di identificare i sotto-obiettivi, sono:

- Cosa si sta realizzando?

- Cosa potrebbe andare storto una volta realizzato?
- Cosa è necessario fare per contrastare eventuali problemi?
- È stato svolto un lavoro di analisi accettabile?

Queste domane indirizzano puntualmente i quattro step del processo illustrato graficamente nella figura che segue e che possono essere così riassunti:

- Modellare il sistema che si sta costruendo, rilasciando o modificando.
- Identificare le minacce usando il modello e gli approcci descritti nel Paragrafo 5.5.4.
- Indirizzare le minacce utilizzando gli approcci descritti nel Paragrafo 5.6.
- Convalidare il lavoro per completezza ed efficacia sulla base delle best practices riportate nel Paragrafo 5.5.4.5.

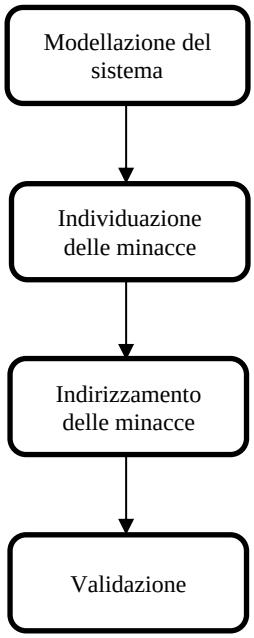


Figura 3 - I quattro step del Framework

Il framework si concretizza come una modalità strutturata per modellare le minacce.

5.5.3.1 Diagrammi DFD

I diagrammi sono un buon metodo per rappresentare ciò che si sta realizzando. Questi sono il modo più efficace per pensare a ciò che si sta costruendo. Ci sono diversi modi per diagrammare il software, e si può iniziare con un diagramma disegnato su una lavagna che rappresenta i flussi di dati e come questi attraversano il sistema.
La modellazione delle minacce si concentra sui dati, su come questi vengono fruiti (flussi) e su come si muovono tra i vari componenti del sistema. I diagrammi di modellazione forniscono una rappresentazione visiva del modo in cui i singoli sottosistemi operano e lavorano insieme. I diagrammi di flusso di dati (DFD) vengono normalmente utilizzati in molti processi di modellazione delle minacce. Questi forniscono un modo coerente e compatto per modellare i flussi di dati presenti in un'applicazione attraverso l'utilizzo di sei forme distinte che rappresentano: il processo, i processi multipli, l'entità esterna, l'archivio dati, il flusso di dati e il perimetro privilegiato (Figura sottostante).

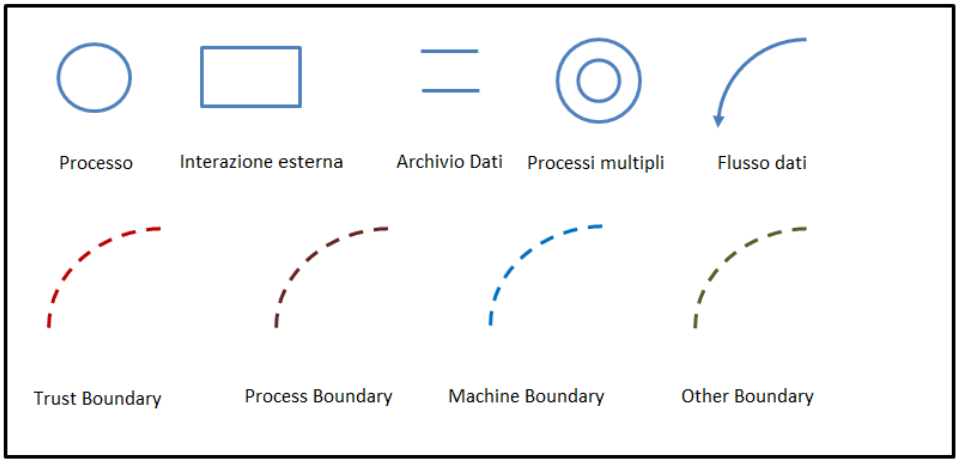


Figura 4 - Simbolismo nel Threat Modeling

I Trust boundaries (confini di fiducia) vengono identificati da una linea tratteggiata rossa, i Process Boundary (confini di processo) vengono invece identificati da una linea tratteggiata marrone, i Machine Boundary (confini della macchina) vengono identificati da una linea tratteggiata blu mentre gli altri confini vengono identificati da una linea tratteggiata verde.
L'utilizzo del colore consente ai team di riconoscere facilmente durante la valutazione del DFD del sistema quale tipo di confine il flusso di dati attraversa. Qui alcuni esempi di elementi che costituisco il DFD:

- Processo o Processi multipli
  - File di libreria dinamica
  - File eseguibile
  - Componente SW/FW
  - Web Service
  - Assemblies
  - Etc.
- Interazione o Entità esterna
  - Persona
  - Altro sistema
  - Sito web
  - Etc.
- Archivio dati
  - Database
  - File
  - Registro
  - Memoria condivisa
  - Code e Stack
  - Etc.
- Flusso dati
  - Chiamata a funzione
  - Traffico di rete
  - Etc.
- Trust boundaries
  - Perimetro del processo
  - File system
  - Etc.

Segue una tabella riepilogativa in cui, per ciascun elemento DFD viene riportato l'aspetto che assume nel diagramma, il significato e alcuni brevi esempi:

ELEMENTO DFD	ASPETTO	SIGNIFICATO	ESEMPIO
Processo	Rettangolo arrotondato , cerchio o cerchio concentrico	Qualunque codice in esecuzione	Codice scritto in C, C#, Python o PHP, etc.
Flusso dati	Freccia	Comunicazione tra processi o tra processi e archivi di dati	Connessioni di rete, HTTP, RPC, LPC, etc.
Archivio dati	Due linee parallele con etichetta nel mezzo	Supporti di memorizzazione dati	File, database, registro di Windows, segmenti di memoria condivisi.
Entità esterna	Rettangolo con angoli retti	Persone o codice al di fuori del nostro controllo	Un utente, un sito web esterno.

Tabella 5 - Caratteristiche degli elementi DFD

L'impiego di diverse tipologie di diagramma concepiti come diversi blocchi di costruzione aiutano a modellare ciò che si sta realizzando.
Nell'esempio che segue, viene rappresentato un modello di una semplice applicazione web che implementa una su logica di business che interagisce con un browser web, con un server web e con un database (vedi figura a seguire).

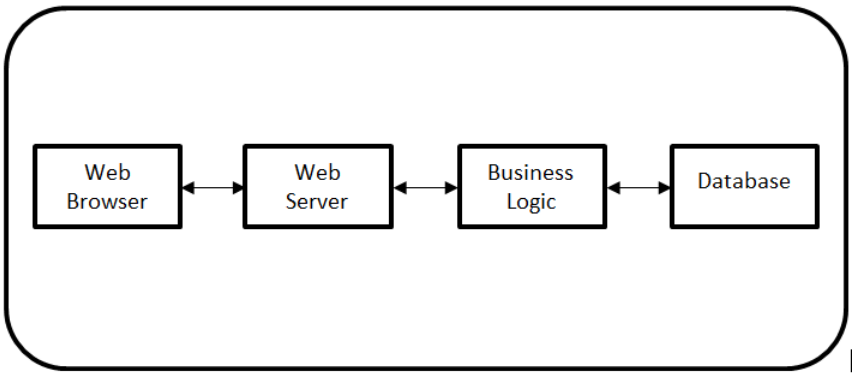


Figura 5 - Diagramma del sistema

Un modo semplice per migliorare il diagramma consiste nel delineare i confini per dare evidenza di "chi controlla cosa". Si può facilmente comprendere che le minacce che attraversano questi confini sono probabilmente le più importanti e possono essere un buon punto di partenza nel processo di identificazione. Questi confini prendono il nome di "trust boundaries" (confini di fiducia) e dovrebbero essere sicuramente disegnati ovunque esiste un controllo da parte di persone. Alcuni esempi significativi:

- Account (User ID sui sistemi Unix o i Security Identifiers su sistemi Microsoft Windows);
- Interfacce di rete;
- Macchine fisiche;
- Macchine virtuali;
- Perimetri organizzativi;
- Ovunque possa essere messa in discussione la diversificazione dei privilegi.

Nel diagramma riportato nella Figura che segue, si aggiungono i "trust boundaries" (rappresentati da rettangoli tratteggiati) e la descrizione di ciò che il perimetro contiene:

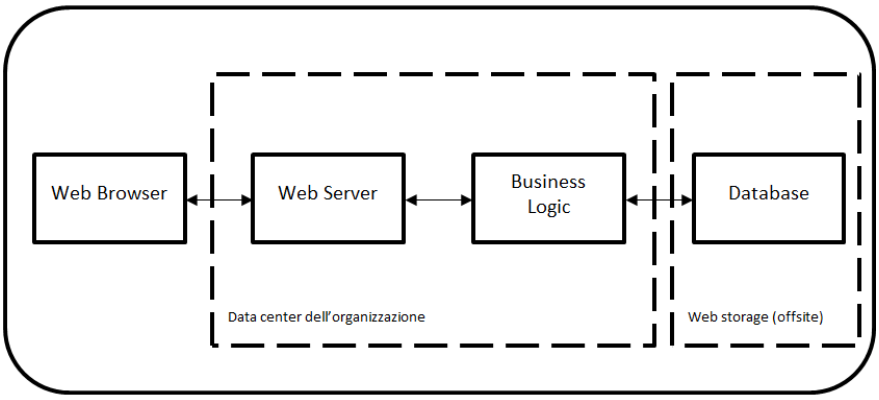


Figura 6 - Aggiunta dei "Trust boundaries" al diagramma

Quando il diagramma diventa più grande e più complesso, può essere molto utile numerare ogni processo, flusso dati e archivio dati presenti nel diagramma, come mostra la figura che segue (Ciascun "trust boundary" dovrebbe avere un identificativo univoco in rappresentanza del suo contenuto):

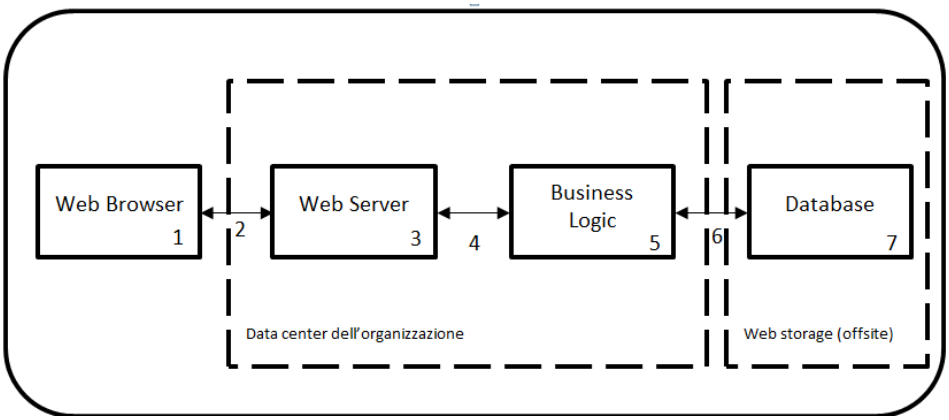


Figura 7 - Numerazione degli elementi del diagramma

Si deve pensare al diagramma del modello come parte integrante del processo di sviluppo, quindi deve essere messo sotto il controllo di versionamento così come tutto il resto del materiale relativo al progetto. Da questo modello, si procede con l'individuazione delle minacce sulla base delle metodologie e delle tecniche descritte nel paragrafo successivo.

5.5.4 Tecniche di modellazione e individuazione delle minacce

L'obiettivo che tutte le metodologie di modellazione delle minacce condividono è lo sviluppo di un processo di passi iterativi che un team può facilmente seguire durante la valutazione di un sistema software.

5.5.4.1 Microsoft SDL – STRIDE

La STRIDE è un processo metodologico che aiuta a individuare le minacce di sicurezza in un sistema complesso. L'elemento mnemonico STRIDE è acronimo di Spoofing, Tampering, Repudiation, Information disclosure, Denial of service ed Elevation of privilege.
Si riporta a seguire la descrizione delle classi di minaccia rappresentate dalla STRIDE:

- **Spoofing** (falsificazione di identità): è la pretesa di essere qualcos'altro o qualcun altro che non si è. Classifica l'insieme delle minacce che consentono a un attaccante di interagire con il sistema

utilizzando un'altra identità. Per esempio, un server di phishing che pretende di essere il server della nostra banca.

- **Tampering** (manomissione dei dati): è l'alterazione di qualcosa che si presuppone non sia oggetto di modifica. Ciò può includere pacchetti di rete (sia fissa che mobile), dati persistiti su supporto di massa o in memoria nonché il codice applicativo. Classifica l'insieme delle minacce che permettono di modificare in modo fraudolento, dati e codice delle applicazioni. Per esempio, un attaccante sfruttando un bug software riesce a cambiare il codice di un'applicazione per aprire una back-door nel sistema.
- **Repudiation** (ripudio di una azione): significa dichiarare di non aver fatto qualcosa (indipendentemente dal fatto che sia stato fatto o no). Classifica l'insieme delle minacce che permettono ad un attaccante di negare di aver compiuto un'azione sul sistema. Per esempio, un utente compie un'azione illegale sul sistema e il sistema non è in grado di rilevare l'azione o di identificare l'utente.
- **Information Disclosure** (divulgazione di informazioni): riguarda l'esposizione delle informazioni a persone non autorizzate alla loro visione. Classifica l'insieme delle minacce che causano l'esposizione di informazioni ad utenti/individui a cui non è consentito l'accesso in lettura. Per esempio, un utente legge un file per il quale non ha ricevuto i diritti di lettura oppure un attaccante legge i dati in transito sulla rete.
- **Denial of Service** (diniego di servizio): sono attacchi designati all'interruzione del servizio erogato da sistemi. Questi includono come effetto il crashing, il rallentamento che porta alla non usabilità del sistema e il riempimento degli storage. Classifica l'insieme delle minacce che permettono di negare o degradare la fornitura di un servizio. Per esempio, un attaccante invia numerosi pacchetti al fine di ostruire la banda di rete di un server il quale non potrà a sua volta essere contattato e/o fornire i suoi servizi agli utenti legittimi.
- **Elevation of Privilege** (elevazione dei privilegi): avviene quando un programma o un utente è tecnicamente abilitato a fare cose che si presuppone non debba fare. Classifica l'insieme delle minacce che permettono ad un utente di ottenere privilegi non previsti per il suo ruolo. Per esempio, un utente anonimo sfrutta un bug software per ottenere i privilegi di amministratore.

In riferimento al diagramma (Figura 7):

- Come si fa a sapere se il browser web verrà utilizzato solo dalle persone che ci si aspetta?
- Cosa accade se qualcuno altera i dati presenti nel database?
- È corretto far transitare i dati da una componente all'atra del sistema senza che questi vengano cifrati?

Questi sono esattamente e rispettivamente esempi di spoofing, tampering e information disclosure che possono essere facilmente individuati con l'ausilio della STRIDE. Con una scarsa conoscenza della sicurezza, ma con l'impiego delle giuste tecniche, è possibile trovare le minacce più importanti in modo veloce e con maggiore affidabilità. Se si sta impiegando un processo di modellizzazione delle minacce, la documentazione prodotta da tale processo, può incrementare il livello di fiducia nel realizzare un software più sicuro.
Per ciascuna minaccia vengono evidenziati gli elementi del diagramma su cui impattano (normalmente si ha maggiore impatto sul software, sui flussi dati o gli storage piuttosto che sui trust boundary).
La lista che segue fornisce alcuni esempi di minacce per ciascuna categoria (l'elenco non vuole essere esaustivo):

SPOOFING	Qualcuno potrebbe fingere di essere un altro utente del nostro sistema, quindi serve un modo per autenticare tutti gli utenti.

	Qualcuno potrebbe fingere di essere il nostro sito web, quindi è necessario assicurarsi di avere un certificato SSL e di utilizzare un singolo dominio per tutte le nostre pagine (per aiutare quel sottoinsieme di utenti che leggono gli URL per vedere se si trovano nel posto giusto).
	Qualcuno potrebbe mettere un collegamento nascosto in una delle nostre pagine, ad esempio logout.html o placeorder.aspx. Dobbiamo controllare il campo HTTP "Referer" prima di intraprendere qualsiasi azione. Non è una soluzione definitiva per contrastare il CSRF (Cross Site Request Forgery), ma è un inizio.

TAMPERING	Qualcuno potrebbe manomettere i dati del back-end.
	Qualcuno potrebbe manomettere i dati in transito tra il data center e il consumer.
	Chi sviluppa potrebbe rilasciare il codice del front-end dell'applicazione senza verificarlo, pensando che sia in fase di caricamento nell'area di staging. Ciò consentirebbe ad uno sviluppatore malintenzionato di aggiungere codice malevolo.

REPUDIATION	Le azioni precedenti potrebbero richiedere una attenta analisi per comprendere ciò che è accaduto. E' necessario porsi delle domande quali: Esistono i log di sistema? Nel log di sistema vengono registrate le giuste informazioni? Il log di sistema è protetto da tampering?

INFORMATION DISCLOSURE	Cosa accade se qualcuno legge i dati presenti nel database? E' possibile che qualcuno possa connettersi al database per leggere o scrivere informazioni?

DENIAL OF SERVICE	Cosa accade se migliaia di utenti si connettono contemporaneamente alla nostra applicazione? E se il sistema va giù?

ELEVATION OF PRIVILEGE	Magari il front-end è l'unico punto di accesso al nostro sito da parte degli utenti, ma cosa lo impone? Cosa previene l'accesso diretto da parte degli utenti alla logica di business in esecuzione sul server o al caricamento di un nuovo codice? Se è presente un firewall, questo è stato correttamente configurato? Chi controlla l'accesso al database, o cosa accade se un impiegato commette un errore o premeditatamente modifica i file a livello di configurazione?

Microsoft ha inizialmente previsto l'applicabilità della STRIDE solo per i punti di ingresso/uscita del sistema in analisi. Tale approccio è stato poi affinato applicando la STRIDE a tutti gli elementi DFD del modello.
Segue una tabella che indica l'esposizione in termini di vulnerabilità secondo la STRIDE rispetto agli elementi DFD utilizzati nel processo di modellazione:

Elemento DFD	S	T	R	I	D	E
Entità Esterna	**X**		**X**			
Flusso Dati		**X**		**X**	**X**	
Archivio Dati		**X**	*	**X**	**X**	
Processo	**X**	**X**	**X**	**X**	**X**	**X**

X – possibile classe di minaccia per l'elemento indicato
*** – applicabile solo se l'archivio dati contiene dati di logging o di auditing**

Tabella 6 - STRIDE per elemento DFD

Le minacce STRIDE sono esattamente l'opposto di alcune delle proprietà che il nostro sistema dovrebbe avere, ovvero: autenticità, integrità, non ripudio, riservatezza, disponibilità e autorizzazione. La seguente tabella, mostra la correlazione tra la categoria di minacce STRIDE, la corrispettiva proprietà che si desidera mantenere, e gli elementi del sistema che sono tipicamente esposti alla classe di minacce indicata.

MINACCIA	PROPRIETA' VIOLATA (requisito o controllo di sicurezza)	TIPICA VITTIMA
Spoofing	Autenticazione	Processo Entità esterna Persona
Tampering	Integrità e Privacy (*)	Processo Archivio dati Flusso dati
Repudiation	Non ripudio/(Autenticazione+Integrità) (*)	Processo
Information Disclosure	Confidenzialità e Privacy (*)	Processo Archivio dati Flusso dati
Denial of Service	Disponibilità e Privacy (*)	Processo Archivio dati Flusso dati
Elevation of Privilege	Autorizzazione	Processo

Tabella 7 - STRIDE proprietà violate

(*) Articoli 25 e 32 del GDPR.

Ad esempio, alcuni elementi quali un flusso di dati tra due componenti, un processo o un archivio di dati; in base alla classificazione STRIDE sono potenzialmente vulnerabili a manomissioni, divulgazione di informazioni e negazione del servizio. Il processo di modellazione delle minacce prende in considerazione ogni minaccia per elemento e valuta se esistono tecniche di mitigazione adeguate per quel tipo di minaccia. Analizzando ciascun elemento del sistema come identificato nei DFD, viene creato un profilo di minaccia dell'applicazione.
La classificazione STRIDE contiene elementi orientati ai dati e alle transazioni. La sicurezza orientata ai dati è rappresentata dalla "triade CIA o RID" e delinea tre distinti principi di base che i sistemi sicuri dovrebbero seguire. Tali principi garantiscono la riservatezza, l'integrità e la disponibilità dei dati o delle comunicazioni. Le minacce STRIDE di divulgazione delle informazioni, manomissione e negazione del servizio sono direttamente correlabili alla triade CIA o RID. Segue una breve descrizione di queste tre tipologie di minaccia:

- Le minacce di divulgazione delle informazioni esistono nel momento in cui la parte malintenzionata può ottenere "un numero elevato di informazioni" in quanto il sistema non è in grado di far rispettare lo stato di confidenzialità delle informazioni stesse.
- Le minacce di manomissione si verificano quando il sistema è soggetto ad attacchi capaci di alterare i dati o le comunicazioni, invalidando così l'integrità dei dati a riposo o in transito.
- Il diniego del servizio è una minaccia alla sicurezza che riduce intenzionalmente la qualità del servizio, rendendolo non più performante, affidabile o disponibile.

La sicurezza orientata alla transazione invece definisce se un interlocutore del sistema ha un'identità ed è autorizzato ad effettuare le comunicazioni richieste. Una volta che tali comunicazioni hanno luogo, un sistema sicuro dovrebbe impedire a qualsiasi interlocutore di negarle o ripudiarle. A questo sottoinsieme appartengono le seguenti categorie di minacce:

- Le minacce di falsificazione dell'identità esistono a causa di una scarsa o inesistente verifica e autenticazione dell'identità stessa. Il sistema non è in grado di verificare l'autenticità della controparte che interagisce con esso.
- L'elevazione del privilegio è la minaccia alla sicurezza maggiormente dannosa. Un controllo di accesso scarso o inesistente (autorizzazione) dà la possibilità ad una possibile controparte malintenzionata di portare uno qualsiasi degli altri attacchi che la STRIDE rappresenta.
- Le minacce di ripudio si verificano quando non è possibile dimostrare l'integrità dei dati e quando l'autenticazione ad essi collegata non può essere verificata. Una parte ostile potrebbe pertanto negarne qualsiasi tipo di associazione per mancanza di prove.

5.5.4.1.1 Tecniche di mitigazione

Ogni minaccia viene mitigata o accettata. Per i non esperti di sicurezza, la STRIDE fornisce per ogni tipo di potenziale minaccia identificata una o più classificazioni delle tecniche di mitigazione da mettere in campo (vedi tabella che segue).

Tipo Minaccia	Tecnica di mitigazione o controlli di sicurezza
Spoofing	**Autenticazione**
Tampering	**Integrità**
Repudiation	**Servizi di non ripudio (Autenticazione + Integrità)**
Information disclosure	**Confidenzialità**
Denial of Service	**Disponibilità**
Elevation of Privilege	**Autorizzazione**

Tabella 8 - Tecniche di mitigazione

AUTENTICAZIONE – TECNICHE DI MITIGAZIONE DELLO SPOOFING

Le tecnologie per l'autenticazione di computer (o account di computer) includono ad esempio:

- IPSec,
- DNSSEC,
- SSH host keys,
- Kerberos authentication,
- HTTP Digest o Basic authentication,
- "Windows authentication" (NTLM),
- Sistemi PKI, come SSL o TLS con certificati.

Le tecnologie per l'autenticazione dei flussi a livello di bit (file, messaggi, ecc.) includono ad esempio:

- Digital signatures,
- Hashes.

I metodi per l'autenticazione delle persone possono coinvolgere uno qualsiasi dei seguenti elementi:

- Qualcosa che sai, come ad esempio una password,
- Qualcosa che hai, come una card di accesso,
- Qualcosa che sei, come ad esempio un dispositivo biometrico, comprese le immagini fotografiche,
- Qualcuno che conosci e che può autenticarti.

Le tecnologie per mantenere l'autenticazione tra le connessioni includono ad esempio:

- Cookies.
- Tokens (es: JWT – JSON Web Token),
- Accesso di terze parti (OAuth, API-token),
- OpenID (protocollo basato su http che utilizza un identity provider per validare l'utente),
- SAML (linguaggio di markup delle asserzioni di sicurezza che fa uso come nel caso OpenID di un identity provider, ma è basato su XML e quindi più flessibile. La versione consigliata per SAML è la 2.0. SAML fornisce anche un modo per implementare il Single SignOn, ovvero l'utente può utilizzare l'URL del fornitore di identità per accedere al sistema che reindirizza con dati XML alla pagina dell'applicazione che può essere decodificato per ottenere le informazioni dell'utente).

On top ai metodi di autenticazione di cui sopra, se necessario, è possibile anche implementare algoritmi One Time Password (OTP), Two Factor Authentication (2FA) e Email verification ecc.

INTEGRITA' – TECNICHE DI MITIGAZIONE DEL TAMPERING

Le tecnologie per la protezione degli asset includono includono ad esempio:

- ACLs o permissions,
- Digital signatures,
- Hashes,
- Windows Mandatory Integrity Control (MIC),
- Unix immutable bits.

Le tecnologie per la protezione del traffico di rete includono ad esempio:

- SSL,
- SSH,
- IPSec,
- Digital signatures.

NON RIPUDIO – TECNICHE DI MITIGAZIONE DELLA REPUDIATION

Le tecnologie che è possibile utilizzare per affrontare il problema del ripudio includono ad esempio:

- Logging,
- Log analysis tools,
- Secured log storage,
- Digital signatures,
- Secure time stamps,
- Trusted third parties,
- Hash trees,
- Strumenti per la prevenzione delle frodi.

CONFIDENZIALITA' – TECNICHE DI MITIGAZIONE DELLA INFORMATION DISCLOSURE

Le tecnologie per la riservatezza includono ad esempio:

- Protezione dei files:
  - ACLs/permissions,
  - Encryption,
  - Appropriata gestione delle keys.
- Protezione dei dati di rete:
  - Encryption,
  - Appropriata gestione delle keys.

- Protezione della comunicazione e delle headers di comunicazione:
  - Mix networks,
  - Onion routing,
  - Steganography.

DISPONIBILITA' – TECNICHE DI MITIGAZIONE DEL DENIAL OF SERVICE

Le tecnologie per la protezione degli asset includono ad esempio:

- ACLs,
- Filters,
- Quotas (rate limiting, thresholding, throttling),
- High-availability design/architetture,
- Bandwidth control (rate limiting, throttling),
- Cloud services. Secondo le ultime ricerche di McAfee[14], la maggior parte dei team di sicurezza afferma di poter ottenere una maggiore sicurezza negli ambienti in-the-cloud visto che, i principali provider di servizi in-the-cloud investono maggiori risorse nella sicurezza di quanto la maggior parte delle aziende possano permettersi per rendere sicuri i propri "hosted environment".

AUTORIZZAZIONE – TECNICHE DI MITIGAZIONE DELL'ELEVATION OF PRIVILEGE

Le tecnologie per migliorare l'autorizzazione includono ad esempio:

- ACLs,
- Group or role membership,
- Role based access control,
- Attribute based access control,
- Claims-based access control,
- Windows privileges (runas),
- Unix sudo,
- Chroot, AppArmor o altre unix sandboxes,
- The "MOICE" Windows sandbox pattern,
- Convalida degli input per uno scopo definito.

5.5.4.1.1.1 Indirizzamento dello Spoofing

La Tabella seguente elenca gli obiettivi di spoofing, le strategie di mitigazione per indirizzare lo spoofing e le tecniche per attuare tali mitigazioni:

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Spoofing di una persona	Identificazione e autenticazione (username e qualcosa che conosci/hai/sei)	Username, nomi reali, identificativi: • Password; • Token; • Biometria. Registrazione/Manutenzione/Scadenza.

[14]https://www.mcafee.com/enterprise/en-us/solutions/lp/cloud-adoption-risk-report-business-growth-edition.html?source=website&lsource=website&eid=BRXDXCDD&smcid=WW

Spoofing di un file su disco	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	• Percorsi assoluti; • Controllo ACL; • Assicurarsi che le pipes vengano create correttamente.
	Autenticatori crittografici	Firme digitali o autenticatori.
Spoofing di un indirizzo di rete	Crittografia	• DNSSEC; • HTTPS/SSL; • IPsec.
Spoofing di un programma in memoria	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	Molti sistemi operativi moderni hanno una qualche forma attuabile di identificazione dell'applicazione.

Tabella 9 - STRIDE: Indirizzamento dello Spoofing

Spoofing di una persona. Per evitare lo spoofing di una persona, è necessario che sia stato implementato un meccanismo di autenticazione e che a questa persona sia stato associato un nome utente univoco. Non è detto che questo sia sufficiente ad evitare lo spoofing: differenti tecniche di autenticazione con profili di sicurezza adeguati devono essere attivate basandosi sulla tipologia di servizio/dati gestiti: ad esempio un autenticazione a 2 fattori può aiutare a mitigare lo spoofing su servizi critici.
Spoofing di un file su disco. Quando si accede ad un file presente sul disco, non bisogna aprirlo utilizzando un percorso relativo come *open(file)*. Utilizzare il percorso assoluto *open(/percorso/assoluto/del/file)*. Se il file contiene dati sensibili, dopo averlo aperto, è necessario attuare un controllo di sicurezza sugli elementi del descrittore del file stesso (come il fully resolved name, i permessi e l'owner). Si potrebbe anche voler controllare il descrittore del file per prevenire eventuali *race conditions.* Ciò vale doppiamente quando il file è un eseguibile, ma il controllo dopo l'apertura potrebbe essere complicato. Ciò, può aiutare a garantire che le autorizzazioni sull'eseguibile non possano essere modificate da parte di un utente malintenzionato. In ogni caso, è quasi sempre sconsigliabile invocare *exec()* con il parametro file specificato in modo relativo "*./file"*.
Spoofing di un indirizzo di rete. Nel caso di spoofing di un indirizzo di rete, è consigliabile l'impiego di protocolli come DNSSEC, SSL, IPsec o una combinazione di questi per assicurarsi di colloquiare con la controparte attesa.
Spoofing di un programma in memoria. Si tratta di programmi che si nascondono mostrandosi come programmi legittimi ma con l'intento di fare danni o trafugare informazioni. Questi sono un tipo di Trojan, i quali duplicano le azioni di processi esistenti che vengono poi eseguite inconsapevolmente dall'utente. E' necessario tenere sempre aggiornato il software come il sistema operativo e il browser web. Mantenere la connessione a Internet il più sicura possibile, tenendo sempre attivo un firewall. Sia i firewall hardware che software sono eccellenti strumenti per controllare il traffico Internet dannoso. E' necessario installare inoltre un software antivirus o un Trojan remover.
La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[15] riconducibili alle minacce di spoofing e per ciascuna vulnerabilità indicata, le relative pratiche[16] e requisiti[17] di sicurezza consigliati da OWASP:

15 https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

16 https://www.owasp.org/index.php/OWASP_Proactive_Controls

17 https://github.com/OWASP/ASVS

OWASP TOP-10 2017 (Rischi di sicurezza delle applicazioni)	OWASP Proactive Controls 2018 v 3.0 (Pratiche di sicurezza proattive)	OWASP ASVS 3.0 (Requisiti di sicurezza applicativa)
A2 – Broken Authentication	C6 – Implementing Digital Identity	V2 - Authentication
		V3 - Session Management

Tabella 10 - Rischi di sicurezza OWASP relativi allo Spoofing

Esempi di minacce di spoofing:

- ARP Spoofing: Un attacco di spoofing ARP è un attacco che utilizza il protocollo di risoluzione degli indirizzi per catturare informazioni. In un attacco ARP spoofing l'aggressore invia messaggi ARP attraverso una rete nel tentativo di collegare il proprio indirizzo MAC con un indirizzo IP target. L'aggressore rimane in attesa sulla rete finché non riesce a violare l'indirizzo IP. Una volta che l'indirizzo IP è stato violato, l'aggressore può intercettare i dati in transito tra il computer relativo all'IP violato e il router. I dati inviati al target vengono quindi inviati all'indirizzo IP dell'aggressore. Il risultato finale è che i dati destinati al legittimo destinatario arrivano nelle mani dell'aggressore. L'aggressore può quindi utilizzare gli indirizzi IP della rete per lanciare un attacco DOS denial-of-service. Una cosa molto importante da tener presente sugli attacchi di ARP spoofing è che questi possono funzionare solo su LAN che utilizzano il protocollo ARP.
- IP Spoofing: Un attacco di IP spoofing si ha quando un aggressore tenta di impersonare un indirizzo IP in modo da poter fingere di essere un altro utente. Durante un attacco di spoofing dell'indirizzo IP, l'aggressore invia pacchetti da un falso indirizzo di origine.
- DNS Spoofing: Gli attacchi DNS o dei nomi di dominio di sistema sono quelli in cui gli aggressori confondono l'elenco degli indirizzi IP pubblici o dei nomi corrispondenti. I server DNS dispongono di un database di indirizzi IP pubblici e hostname che vengono utilizzati per facilitare la navigazione in rete. Quando si verifica un attacco DNS, l'aggressore modifica i nomi di dominio in modo che vengano reindirizzati a un nuovo indirizzo IP. Un esempio di ciò si ha quando s'inserisce un URL di un sito Web e si viene re diretti verso un dominio spoofed piuttosto che verso il sito Web atteso. Ciò rappresenta un mezzo comune per gli aggressori per diffondere worm e virus nelle reti.
- Email Spoofing: Gli attacchi di spoofing delle e-mail sono quelli in cui un attaccante invia un'e-mail emulando un altro mittente. In questi attacchi, il campo mittente viene alterato per mostrare dettagli falsificati di contatto. L'aggressore impersonifica il mittente scelto e poi invia un'e-mail con una richiesta di informazioni. Questi attacchi vengono spesso impiegati per spacciarsi come amministratori e chiedere agli altri membri dell'organizzazione dettagli sui propri account.

5.5.4.1.1.2 Indirizzamento del Tampering

La Tabella seguente mostra in elenco gli obiettivi di tampering, le strategie di mitigazione per indirizzare il tampering e le tecniche per attuare tali mitigazioni.

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Tampering di un file	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	ACLs.
	Crittografia	• Firme digitali; • Keyed MAC.
Concorrenza nella creazione di un file (tampering del file system)	Utilizzo di una directory protetta da manipolazione arbitraria di un utente	• ACLs; • Utilizzo di strutture private di directory;

		• La randomizzare dei nomi dei file contrasta l'esecuzione di possibili attacchi.
Tampering dei pacchetti di rete	Crittografia	• HTTPS/SSL; • IPsec.
	Anti-pattern	Isolamento della rete.

Tabella 11 - STRIDE: Indirizzamento del Tampering

Tampering di un file. Se un attaccante possiede un account su una macchina, questo può facilmente portare un attacco di tampering su di un file che risiede sulla stessa macchina, oppure quando questo transita sulla rete per essere ricevuto.
Tampering della memoria. Ciò avviene quando un processo con privilegi minimi di trust o non, altera in qualche modo la memoria fisica della macchina. Per esempio, se si stanno prendendo dati da un segmento di memoria condivisa, esistono delle ACL tali da consentirne agli altri processi la sola lettura? Per le applicazioni web che acquisiscono dati tramite AJAX, assicurarsi di validare tali dati prima di darli in pasto alla logica di business.
Tampering del traffico di rete. La prevenzione del traffico di rete richiede una gestione sia dello spoofing che del tampering. Diversamente, chiunque intenzionato ad alterare i dati in transito potrebbe semplicemente far finta di essere all'altra estremità, portando invece un attacco di tipo MITM (Man In The Midle). La soluzione più comune per contrastare questo problema è utilizzare il protocollo SSL o l'IPsec (IP Security) come infrastruttura di comunicazione. Entrambi, SSL e IPsec indirizzano le problematiche legate alla confidenzialità e all'integrità delle informazioni e possono anche aiutare ad indirizzare lo spoofing.
Tampering del traffico di rete attraverso l'anti-pattern. L'isolamento della rete non assicura la protezione dalle minacce di tampering poiché generalmente non si riesce a mantenere la rete costantemente isolata.

La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[18] riconducibili alle minacce di tampering e per ciascuna vulnerabilità indicata, le relative pratiche[19] e requisiti[20] di sicurezza consigliati da OWASP:

OWASP TOP-10 2017 (Rischi di sicurezza delle applicazioni)	**OWASP Proactive Controls 2018 v 3.0** (Pratiche di sicurezza proattive)	**OWASP ASVS 3.0** (Requisiti di sicurezza applicativa)
A1 – Injection	C5 – Validate All Inputs	V5 - Malicious Input Handling V11 - HTTP Security Configuration
	C4 – Encode and Escape Data	V5 - Malicious Input Handling
A4 - XML External Entities (XXE)	C6 – Validate All Inputs	V5 - Malicious Input Handling
	C4 – Encode and Escape Data	V5 - Malicious Input Handling
A7 - Cross-Site Scripting (XSS)	C5 – Validate All Inputs	V5 - Malicious Input Handling
	C4 – Encode and Escape Data	V5 - Malicious Input Handling

18 https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

19 https://www.owasp.org/index.php/OWASP_Proactive_Controls

20 https://github.com/OWASP/ASVS

A8 - Insecure Deserialization	C5 – Validate All Inputs	V5 - Malicious Input Handling
	C4 – Encode and Escape Data	V5 - Malicious Input Handling

Tabella 12 - Rischi di sicurezza OWASP relativi al Tampering

Esempi di minacce di Tampering:

- Manomissione dei Cookie: I cookie vengono utilizzati come meccanismo per memorizzare informazioni di dettaglio e preferenze dell'utente nonché altri dati, come i token di sessione. I cookie di tipo persistente e non, sicuri o non, possono essere manomessi dall'utente e inviati al server tramite richieste di Uniform Resource Locator, pertanto qualsiasi utente o hacker malintenzionato può modificare il contenuto dei cookie a suo vantaggio consentendo a se stesso di accedere ai file desiderati.
- Manomissione dei campi di Form HTML: Quando un utente effettua selezioni o modifiche su una pagina web o HTML, la selezione viene memorizzata come valore di un campo del form, che viene poi inviato all'applicazione attraverso una richiesta HTTP. L'HTML solitamente memorizza tali valori come campi nascosti (Hidden Fields), che non vengono mostrati sullo schermo dell'utente, ma vengono raccolti e inviati come stringhe o parametri durante la trasmissione del form. Se tali campi possono essere nascosti, preselezionati o liberi, possono anche essere manomessi o manipolati dall'hacker per inviare valori a sua scelta.
- Manomissione della stringa di Query URL: La manomissione degli URL comporta tutta una serie di problemi legati alla presenza di campi nascosti nei Form. Per sottomettere i dati all'applicazione viene utilizzato uno dei due metodi usati dai form HTML, POST o GET. Di solito viene usato il metodo GET, il quale mostra tutti i nomi degli elementi dei form e i relativi valori nella stringa di query dell'URL che l'hacker è in grado di vedere. Gli hacker trovano più facile manomettere le stringhe di query che manomettere i campi nascosti del form. Tutto quello che l'hacker deve fare è guardare l'URL presente nella barra degli indirizzi dell'utente. Ad esempio, una pagina web potrebbe consentire all'utente autenticato di selezionare uno dei suoi account pre-caricati da un campo con valori multipli e di addebitare all'account selezionato un importo unitario fisso. La scelta viene registrata premendo un pulsante di invio. La pagina memorizza effettivamente i valori scelti nei relativi campi del form e invia tali valori all'applicazione utilizzando un comando di submit del form. Tale comando sottomette la seguente richiesta HTTP: http://www.vittima.org/esempio?numeroconto=12345&addebito=1, ora tutto ciò che l'hacker deve fare è costruire il proprio numero di conto e modificare i parametri come segue: http://www.vittima.org/esempio?numeroconto=98760&addebito=100000000.
- Password Cracking: Un password cracker è un programma applicativo che viene utilizzato per supportare un hacker o un utente malintenzionato a identificare una password sconosciuta di accesso a un computer o a un dispositivo di rete con il fine di ottenere o consentire l'accesso non autorizzato alle sue risorse. L'hacker tenterebbe di ottenere le credenziali valide dal sistema di autenticazione attraverso un numero elevato di tentativi di autenticazione con password diverse ripetuti nel tempo. Il programma di Password cracking utilizza principalmente due metodi per cercare o identificare le password valide, che sono: la "forza bruta" e le "ricerche basate su dizionario". Quando il programma utilizza la "forza bruta", questo sottomette al sistema di autenticazione diverse combinazioni di tutti i tipi di caratteri con una lunghezza predeterminata fino a quando non viene identifica la combinazione corretta per il sistema informatico. Quando invece utilizza la "ricerca basata su dizionario", il programma applicativo utilizza ogni parola presente nel dizionario alla ricerca della password che il sistema informatico riconosce come valida.
- Manomissione dell'intestazione HTTP: Le intestazioni HTTP vengono utilizzate dal software del server web e dal client. Nella maggior parte delle applicazioni web tali intestazioni non vengono usate. Alcuni sviluppatori web scelgono di monitorare le intestazioni in entrata ed è importante notare che le intestazioni di richiesta provengono originariamente dal client, e come tali potrebbero essere manipolate da un potenziale aggressore. Normalmente le applicazioni web non consentono l'alterazione o la modifica delle intestazioni HTTP. Un hacker dovrà dunque scrivere il proprio

programma per sottomettere la propria richiesta HTTP o può utilizzare un proxy liberamente disponibile che gli consente di modificare facilmente qualsiasi dato inviato dall'applicazione web.

5.5.4.1.1.3 Indirizzamento della repudiation

Indirizzare la repudiation in genere significa garantire che il sistema venga progettato prevedendo il tracciamento e la registrazione delle operazioni svolte (logging), garantendo inoltre che, tali registrazioni vengano protette e preservate. Alcuni di questi registri possono essere gestiti utilizzando un trasporto affidabile specifico. In questo senso, il syslog su UDP presenta forti lacune in termini di sicurezza; il syslog su TCP / SSL invece è notevolmente migliore.
La Tabella seguente mostra in elenco gli obiettivi di repudiation, le strategie di mitigazione per indirizzare la repudiation e le tecniche per attuare tali mitigazioni:

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Non utilizzare un meccanismo di log vuol dire non poter provare nulla.	Log	Assicurarsi di tracciare tutte le informazioni rilevanti dal punto di vista della sicurezza.
Esposizione dei Logs a eventuali attacchi	Proteggere i logs	• syslog su TCP/SSL; • ACL.
Il Log come canale di attacco	Informazioni dettagliate sul log	Documentare la progettazione del log sin dall'inizio del processo di sviluppo.

Tabella 13 - STRIDE: Indirizzamento della repudiation

Non avere un log significa non poter provare nulla. Prevedere e mantenere i log, significa, poter investigare su quanto accaduto, al fine di acquisire un valido riscontro, nel momento in cui qualcuno nega di aver ottenuto o fatto qualcosa.
Esposizione del log a possibili attacchi. Eventuali aggressori faranno del tutto per invalidare le informazioni contenute nel log, contrastando a volte l'operazione stessa di scrittura o forzando il "roll over" del log, con il fine di rendere difficile l'individuazione dell'attacco. Questi inoltre, possono agire in modo tale da disattivare gli allarmi, facendo sì che, il vero attacco non venga alla luce.
Il log come canale di attacco. Da progettazione, è possibile che vengono raccolti dati provenienti da sorgenti 'malevoli' fuori dal nostro controllo, fornendo poi gli stessi dati a persone o sistemi che hanno dei privilegi di sicurezza. Un esempio di questa tipologia di attacco potrebbe essere l'invio di una e-mail indirizzata a "</html> destinatario@dominio.com", che causa problemi a quegli strumenti web-based che non prevedono l'HTML inline.
È possibile rendere il tutto più semplice scrivendo codice sicuro nell'elaborazione dei dati di log, dando chiara evidenza di ciò che questi non possono contenere, ad esempio "I dati di log sono tutti in chiaro, ed eventuali aggressori potrebbero inserire ciò che vogliono" oppure "I campi da 1 a 5 del tracciato del log sono sotto stretto controllo da parte del nostro software, mentre i campi da 6 a 9 sono facilmente iniettabili. Il campo 1 è un campo time GMT. I campi 2 e 3 sono indirizzi IP (v4 o v6) ... ".

La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[21] riconducibili alle minacce di repudiation e per ciascuna vulnerabilità indicata, le relative pratiche[22] e requisiti[23] di sicurezza consigliati da OWASP:

OWASP TOP-10 2017 (Rischi di sicurezza delle applicazioni)	OWASP Proactive Controls 2018 v 3.0 (Pratiche di sicurezza proattive)	OWASP ASVS 3.0 (Requisiti di sicurezza applicative)
A10 – Insufficient Logging & Monitoring	C9 - Implement Security Logging and Monitoring	V8 - Error Handling and Logging

Tabella 14 - Rischi di sicurezza OWASP relativi alla Repudiation

Alcuni esempi di minacce di repudiation:

- Potenziali debolezze nella protezione dei dati di audit: E' buona norma considerare cosa accade quando il meccanismo di audit viene attaccato, compresi eventuali tentativi di distruggere i log o i programmi di analisi dei log a supporto del processo di indagine sugli attacchi. E' opportuno assicurarsi costantemente che l'accesso al log sia strettamente monitorato e che esiste e venga utilizzato un meccanismo efficace capace di controllare la lettura e la scrittura separata su log.
- Auditing insufficiente: Il log deve raccoglie dati sufficienti per comprendere cosa è accaduto in passato. Questo deve acquisire dati sufficienti per essere in grado successivamente di riconoscere un possibile incidente di sicurezza. Tale acquisizione deve essere sufficientemente contenuta da poter essere lasciata sempre attiva. Si deve sempre disporre di dati sufficienti per gestire eventuali richieste di ripudio. Dunque è importante assicurarsi di loggare i dati sufficienti e appropriati per gestire eventuali controversie. Assicurarsi inoltre di tenere sempre in considerazione gli aspetti relativi alla privacy dei dati durante la verifica dei log.
- Dati di log provenienti da una fonte sconosciuta o da soggetti poco affidabili: Trattasi di dati di log provenienti da utenti o sistemi sconosciuti o debolmente autenticati. Un'altra entità presente al di fuori del più esterno confine di fiducia è stato autorizzato a scrivere su log. Di norma è buona pratica identificare e autenticare la fonte dei log prima che questi vengano accettati e consentire l'accesso al log solo da codice fidato.

5.5.4.1.1.4 Indirizzamento dell'information disclosure

La Tabella seguente mostra in elenco gli obiettivi dell'information disclosure, le strategie di mitigazione per indirizzare l'information disclosure e le tecniche per attuare tali mitigazioni:

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Monitoraggio della rete	Crittografia	• HTTPS/SSL; • IPsec.
Directory o nomi di file (per esempio "lettere-di-licenziamento/" o "NomeCognome.docx")	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	ACLs.
Contenuti di un file	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	ACLs.

21 https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

22 https://www.owasp.org/index.php/OWASP_Proactive_Controls

23 https://github.com/OWASP/ASVS

	Crittografia	Crittografia dei file come PGP, crittografia del disco (FileVault, BitLocker).
API information disclosure	Progettazione	Attento controllo nella progettazione, considerando il passaggio dei parametri per indirizzo o valore.

Tabella 15 - STRIDE: Indirizzamento dell'Information disclosure

Monitoraggio della rete. Il processo di monitoraggio della rete, si avvale dell'architettura per monitorarne il traffico (In particolare, la maggior parte delle attuali reti inviano i pacchetti sulla rete e ciascun listener presente su ogni end-point deve decidere se il pacchetto è per lui importante o meno). Quando le reti vengono progettate in modi diversi, esistono svariate tecniche per tracciare il traffico che va verso o attraversa la stazione di monitoraggio. Se non viene indirizzato lo spoofing, così come il tampering, un attaccante può mettersi nel mezzo attuando lo spoofing su ciascun end-point. La mitigazione dell'information disclosure sulla rete richiede la gestione delle minacce di spoofing e di tampering. Se non si indirizza il tampering, ci sono diversi modi intelligenti per ottenere informazioni. Quindi di nuovo, l'SSL e l'IPSec sono le migliori scelte da mettere in campo.
Nomi che rivelano informazioni. Quando il nome di una directory o di un file di per se forniscono informazioni utili ad un possibile attaccante, il modo migliore per proteggersi è creare una directory padre con un nome anonimo (cioè non correlato al servizio oppure ai dati trattati) e utilizzare le ACLs o i permessi del sistema operativo per proteggerle.
Contenuti sensibili nei file. Quando il contenuto di un file necessita di protezione, utilizzare le ACLs o la crittografia. Nel caso in cui la macchina (computer) dovesse cadere in mani non autorizzate, è necessario preventivamente utilizzare la crittografia al fine di proteggere tutti i dati in essa presenti. Le modalità di protezione crittografica che prevedono l'inserimento di una chiave o parola chiave da parte di una persona, sono più sicure ma meno convenienti. Esistono tecniche crittografiche per i file, filesystem e database, dipende da ciò che si deve proteggere.
API (Interfaccia di programmazione di una applicazione). Quando si progetta un'API, o diversamente si trasmettono informazioni oltre un confine di fiducia, è importante fare attenzione a quali informazioni si espongono. È necessario partire dal presupposto che le informazioni fornite vengono passate ad altri, quindi bisogna essere molto cauti e selettivi su ciò che viene fornito. Situazioni di errore generate da un sito web che mostrano il nome utente e la password di un database sono un esempio comune di questo problema.
La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[24] riconducibili alle minacce di information disclosure, e per ciascuna vulnerabilità indicata, le relative pratiche[25] e requisiti[26] di sicurezza consigliati da OWASP:

OWASP TOP-10 2017 (Rischi di sicurezza delle applicazioni)	**OWASP Proactive Controls 2018 v 3.0** (Pratiche di sicurezza proattive)	**OWASP ASVS 3.0** (Requisiti di sicurezza applicative)
A3 – Sensitive Data Exposure	C8 - Protect Data Everywhere	V7 - Cryptography at Rest
A3 – Sensitive Data Exposure	C8 - Protect Data Everywhere	V9 - Data Protection
A3 – Sensitive Data Exposure	C8 - Protect Data Everywhere	V11 - Http Security Configuration

[24] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

[25] https://www.owasp.org/index.php/OWASP_Proactive_Controls

[26] https://github.com/OWASP/ASVS

A3 – Sensitive Data Exposure	C10 - Handle All Errors and Exceptions	V8 - Error Handling and Logging
A6 – Security Missconfiguration	C10 - Handle All Errors and Exceptions	V19 - Configuration

Tabella 16 - Rischi di sicurezza OWASP relativi all'Information Disclosure

Alcuni esempi di minacce di information disclosure:

- Banner Grabbing/ricognizione attiva: Il Banner grabbing o ricognizione attiva è un tipo di attacco durante il quale gli aggressori inviano richieste al sistema che stanno tentando di attaccare per raccogliere maggiori informazioni riguardo il sistema stesso. Se il sistema non è ben configurato, può rivelare informazioni su se stesso, come la versione del server, la versione PHP/ASP.NET, la versione OpenSSH, ecc. Nella maggior parte dei casi, il banner grabbing non si traduce in una fuga di informazioni sensibili, ma piuttosto di informazioni che possono essere di aiuto per l'aggressore nella fase di sfruttamento dell'attacco. Ad esempio, se dal sistema target trapela la versione del PHP in esecuzione sul server, e tale versione risulta vulnerabile al Remote Command/Code Execution (RCE) in quanto non aggiornato, gli aggressori possono sfruttare la vulnerabilità nota e prendere il pieno controllo dell'applicazione web.
- Divulgazione del codice sorgente: I problemi di divulgazione del codice sorgente si verificano quando il codice dell'ambiente di backend di un'applicazione web viene pubblicamente esposto. La divulgazione del codice sorgente consente agli aggressori di comprendere come è fatta l'applicazione e come questa si comporta, semplicemente leggendo il codice e verificando le difettosità presenti nella logica applicativa, o rilevando le coppie username/password o le chiavi riservate delle API scolpite nel codice. La severità riguardo la sicurezza dell'applicazione web quindi dipende dal livello di esposizione del codice e dal livello di riservatezza delle linee di codice in esso contenute e divulgate. In breve, la divulgazione del codice sorgente trasforma un processo di penetration test black box in un approccio white box, dato che l'aggressore ha accesso al codice sorgente. Le problematiche di divulgazione del codice sorgente possono verificarsi in numerosi modi, di seguito se ne elencano alcuni:
  - **Repository di codice pubblico non protetto**: Numerose organizzazioni spesso ospitano il loro codice sorgente nel cloud per migliorare i metodi di sviluppo collaborativo. Tali repository a volte non vengono ben protetti e possono consentire agli aggressori di accedere al codice sorgente e alle informazioni in esso presenti. Inoltre, alcune organizzazioni che sviluppano software open source utilizzano repository pubblici in modo che chiunque possa contribuire allo sviluppo del progetto. In tal caso il codice sorgente è già pubblico, ma non è raro che il codice sorgente pubblicamente disponibile contiene informazioni sensibili.
  - **MIME Types non corretti**: I browser web sanno come analizzare le informazioni che ricevono dall'intestazione HTTP Content-Type, che viene inviata dal server web nella risposta HTTP. Per esempio se l'intestazione Content-Type è impostata su text/html, il browser analizzerà l'HTML e mostrerà il relativo output. Anche se il server web non è correttamente configurato, e ad esempio serve una pagina HTML inviando l'intestazione Content-Type: text/plain invece di Content-Type: text/html, tale pagina sarà resa come testo semplice nel browser web, permettendo all'attaccante di vedere il codice sorgente della pagina stessa.
- Trattamento inadeguato dei dati sensibili: Un altro errore comune è l'hardcoding di informazioni riservate o sensibili come le coppie username/password, gli indirizzi IP interni presenti negli script e i commenti nel codice e nelle pagine web. Nella maggior parte dei casi tali informazioni vengono lasciate all'interno dell'applicazione web di produzione. La divulgazione di tali informazioni può avere esiti devastanti per l'applicazione stessa; l'aggressore deve solo cercare queste informazioni nella fonte delle pagine (ad esempio, facendo un clic destro sulla pagina e selezionando "Visualizza fonte pagina", da non confondere con "Codice sorgente").
- Divulgazione di Nomi di file e di percorso: In alcune circostanze le applicazioni web possono rivelare nomi di file o percorsi di directory, divulgando così informazioni sulla struttura del sistema

sottostante. Questo può accadere a causa di un'errata gestione dell'input dell'utente, di eccezioni nel backend o di una configurazione inappropriata del server web. A volte tali informazioni possono essere individuate o identificate nell'output delle applicazioni web, pagine di errore, informazioni di debug, e quant'altro. Un esempio di divulgazione di nome di file e di percorso si ha quando a partire da un semplice test un attaccante può verificare se l'applicazione web rivela nomi di file o percorsi inviando un certo numero di richieste distinte che, a suo avviso, il server potrebbe gestire in modo diverso. Per esempio, quando si invia una richiesta del tipo che segue, l'applicazione web restituisce una risposta 403 (accesso negato):

https://www.esempio.org/%5C../etc/passwd

Ma quando l'attaccante invia la seguente richiesta, ottiene una risposta 404 (pagina non trovata):

https://www.esempio.org/%5C../etc/fake

Poiché per la prima richiesta l'attaccante ha ottenuto un errore 403 "accesso negato" e per la seconda un 404 "pagina non trovata", questo sa che nel primo caso il file in questione esiste. L'aggressore può quindi utilizzare a suo vantaggio il comportamento dell'applicazione web, come exploit per comprendere come è strutturato il server e per verificare se nel sistema esiste una certa directory o file. Un altro caso in cui il sistema è soggetto a divulgazione di informazioni riguardo gli elementi del file-system sottostante è rappresentato dal cosiddetto "Directory Listing" ovvero l'elenco delle directory e dei file presenti nel server web. Questa funzionalità viene normalmente fornita di default sui server web. Quando non viene definita una pagina web predefinita per il server web, nel momento in cui si tenta di accedere al server tramite una richiesta http, questo restituisce all'utente un elenco di file e directory presenti sul sito web. Quindi se il nome file predefinito su un server web Apache è index.php, e questo non è presente nella directory principale del sito web, il server mostrerà l'elenco di directory presenti nella directory principale invece di mostrare l'output del file php. Lasciare tale funzionalità abilitata negli ambienti di produzione è una cattiva pratica e può portare a numerosi problemi di sicurezza.

5.5.4.1.1.5 Indirizzamento del denial of service

La Tabella seguente mostra in elenco gli obiettivi del Denial Of Service, le strategie di mitigazione per indirizzare il Denial Of Service e le tecniche per attuare tali mitigazioni:

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Saturazione della rete (Network flooding)	Verificare le risorse esauribili	• Risorse flessibili (Elastic resources); • Progettare pensando che il consumo di risorse da parte di un futuro utilizzatore malintenzionato possa essere alto o comunque superiore a quello presunto.
		ACLs di rete.
Risorse utilizzate da un programma	Progettazione attenta e cautelativa	Gestione delle risorse flessibili.
	Evitare fattori moltiplicativi	Analizzare i punti in cui un eventuale attacco portato con uno sforzo minimo, potrebbe produrre una moltiplicazione nel consumo di CPU. Intervenire in modo tale da rendere maggiormente

		arduo il lavoro dell'attaccante abilitandone l'identificazione, come i client che applicano la crittografia o il login prima di procedere con il vero lavoro (ovviamente ciò non vuol dire che gli accessi non debbano essere crittografati).
Risorse di sistema	Sfruttare le funzionalità messe a disposizione dal Sistema Operativo	Utilizzare le impostazioni del sistema operativo.

Tabella 17 - STRIDE: Indirizzamento del Denial of Service

Saturazione della rete. Se si dispone di strutture statiche di connessione, cosa accade se queste si saturano? L'utilizzo di firewall può fornire un livello di protezione basato su ACLs per controllare l'accettazione (o l'invio) di dati e possono essere utili per mitigare gli attacchi di negazione di servizio di rete.
Individuazione delle risorse esauribili. Si possono identificare tre tipologie distinte di risorse esauribili: la prima è quella relativa alle risorse di rete; la seconda è quella relativa a quelle risorse direttamente gestite lato codice; la terza è quella relativa alle risorse gestite dal sistema operativo. In ogni caso, le risorse flessibili (elastic resources) risultano sempre essere una tecnica preziosa. Ad esempio, negli anni 90 alcuni stack TCP avevano un limite hardcoded di cinque connessioni TCP semiaperte (una connessione semiaperta è una connessione che viene attivata nel processo di avvio. Non bisogna preoccuparsi del fatto che ciò potrebbe non avere un senso, piuttosto bisognerebbe chiedersi il motivo per cui questo limite fu impostato a cinque). Oggi è spesso possibile ottenere risorse flessibili dai vari tipi di cloud provider.
Risorse di sistema. I sistemi operativi tendono ad avere limiti o quote per controllare il consumo di risorse a livello di codice applicativo. Considerare le risorse gestite dal sistema operativo, come la memoria o l'utilizzo del disco. Se il codice applicativo viene eseguito su server dedicato, può essere ragionevole consentirgli di utilizzare tutte le risorse di cui quel server dispone. Prestare attenzione a porre gli opportuni limiti al codice e assicurarsi di documentare quanto prestabilito.
Risorse del programma. Acquisire consapevolezza circa le limitazioni che si potrebbero avere nella gestione delle risorse applicative. Ad esempio, un attaccante potrebbe portare l'applicazione software ad un dispendio di lavoro e risorse inviando un flusso dati che richiederebbe costose operazioni crittografiche che potrebbero esporre il software stesso a vulnerabilità di "Denial Of service".
La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[27] riconducibili alle minacce di denial of service, e per ciascuna vulnerabilità indicata, le relative pratiche[28] e requisiti[29] di sicurezza consigliati da OWASP:

OWASP TOP-10 2017 **(Rischi di sicurezza delle applicazioni)**	**OWASP Proactive Controls 2018 v 3.0** **(Pratiche di sicurezza proattive)**	**OWASP ASVS 3.0** **(Requisiti di sicurezza applicative)**
A3 - Sensitive Data Exposure	C10 - Handle All Errors and Exceptions	V8 - Error Handling and Logging
A6 - Security Misconfiguration	C3 - Secure Database Access	V19 - Configuration

[27] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

[28] https://www.owasp.org/index.php/OWASP_Proactive_Controls

[29] https://github.com/OWASP/ASVS

A6 - Security Misconfiguration	C10 - Handle All Errors and Exceptions	V19 - Configuration
A9 - Using Components with Known Vulnerabilities	C3 - Secure Database Access	V13 - Malicious Controls

Tabella 18 - Rischi di sicurezza OWASP relativi al Denial Of Service

Alcuni esempi di minacce di denial of service:

- Ping of Death: Il comando ping viene solitamente utilizzato per verificare la disponibilità di una risorsa di rete. Il principio di funzionamento si basa sull'invio di piccoli pacchetti di dati alla risorsa di rete target. Il ping of death sfrutta tale principio e invia pacchetti di dati superando il limite massimo (65.536 byte) consentito dal TCP/IP. La frammentazione TCP/IP esegue una scomposizione in blocchi più piccoli che vengono poi inviati al server. A causa di questa tipologia di attacco, poiché i pacchetti di dati inviati sono più grandi di quelli che il server può gestire, il server può bloccarsi, riavviarsi o può andare in crash.
- Smurf: Questo tipo di attacco utilizza grandi quantità di traffico di tipo Internet Control Message Protocol (ICMP) nei confronti di un indirizzo Internet Broadcast. L'indirizzo IP di richiesta viene contraffatto con l'indirizzo IP della vittima. Ne consegue che tutte le risposte vengono inviate alla vittima piuttosto che all'IP utilizzato in origine dal ping. Poiché un singolo indirizzo Internet Broadcast Address può supportare un massimo di 255 host, un attacco smurf amplifica di 255 volte un singolo ping. L'effetto risultante è quello di rallentare la rete a un punto tale da rendere la rete stessa inutilizzabile.
- Buffer overflow: Un buffer è una area di memoria temporanea presente nella RAM che viene utilizzata per contenere dati che possono essere elaborati dalla CPU prima che questi vengano scritti su disco. I buffer hanno un limite di dimensione. Questo tipo di attacco sovraccarica il buffer con un numero di dati maggiore di quanto ne può contenere. In tal modo il buffer si riempie oltre il limite corrompendo i dati presenti in memoria e determinando conseguentemente un instabilità nel sistema.
- Teardrop: Questo tipo di attacco utilizza pacchetti di dati di grandi dimensioni. Il TCP/IP li scompone in frammenti che vengono poi assemblati sull'host ricevente. L'attaccante manipola questi pacchetti man mano che vengono inviati in modo tale da sovrapporli. Ciò può causare l'arresto anomalo dell'host vittima durante la fase di riassemblamento.
- Syn attack: SYN è una forma abbreviata di Synchronize. Questo tipo di attacco sfrutta le tre modalità di handshake (negoziazione iniziale) per stabilire una comunicazione TCP. L'attacco SYN funziona inondando la vittima con messaggi SYN incompleti. Ciò fa sì che l'host vittima assegni risorse di memoria che normalmente non vengono mai utilizzate negando conseguentemente l'accesso agli utenti legittimi per mancanza delle necessarie risorse.

5.5.4.1.1.6 Indirizzamento dell'elevation of privilege

La Tabella seguente mostra in elenco gli obiettivi dell'Elevation Of Privilege, le strategie di mitigazione per indirizzare l'Elevation Of Privilege e le tecniche per attuare tali mitigazioni.

OBIETTIVO DELLA MINACCIA	STRATEGIA DI MITIGAZIONE	TECNICA DI MITIGAZIONE
Confusione tra dati/codice	Adottare strumenti e architetture che inducono a separare i dati dal codice.	• Prepared Statement o Stored procedure per l'SQL; • Separatori chiari con forme canoniche; • Validare i dati prima di passarli al consumer.
Attacchi di compromissione del	Utilizzare un linguaggio di programmazione sicuro	L'utilizzo di un linguaggio di programmazione sicura nella stesura

flusso di controllo e della memoria		del codice ci protegge da intere classi di attacco.
	Sfruttare il sistema operativo per la protezione della memoria.	I sistemi operativi di ultima generazione possiedono meccanismi intrinsechi di protezione della memoria.
	Utilizzare il sandboxing	• I sistemi operativi moderni supportano il sandboxing in vari modi (AppArmor su Linux, AppContainer o il modello MOICE su Windows, Sandboxlib su Mac OS). • Non utilizzare per l'esecuzione l'account "nobody", crearne uno nuovo per ciascuna applicazione.
Attacchi di command-injection	Porre la dovuta attenzione durante la fase implementativa della logica di business	Validare l'input in termini di forma e dimensione attesa. Non bonificare. Tracciare l'input nel log e scartarlo se non viene riconosciuto.

Tabella 19 - STRIDE: Indirizzamento dell'Elevation of privilege

Confusione tra dati/codice. Accade spesso che i dati vengono trattati come codice. Attacchi come gli XSS sfruttano l'unione tra codice HTML e dati (un file html che contiene sia codice, come Javascript, che dati, come il testo da visualizzare e talvolta anche istruzioni di formattazione per il testo stesso). Esistono alcune strategie per affrontare tale problematica. Il primo consiste nell'adottare modalità/strumenti che aiutano a mantenere separati codice e dati (ad esempio, i prepared statement in SQL indicano al database quali dichiarazioni aspettarsi e dove saranno posizionati i dati). Un'altra strategia è validare i dati prima di inviarli. Ad esempio, se si stanno inviando dati attraverso una pagina web, è necessario assicurarsi che questi non contengano caratteri come <, >, # o & e quant'altro.
Attacchi di compromissione del flusso di controllo e/o della memoria. Questo insieme di attacchi generalmente sfrutta il "weak typing" e le strutture statiche presenti nei linguaggi simili al C per consentire ad un aggressore di introdurre del codice per poi eseguirlo. Se si utilizza un linguaggio sicuro, come Java o C#, molti di questi attacchi sono più difficili da portare. I sistemi operativi più moderni tendono a incorporare funzioni di protezione e di randomizzazione della memoria, come ad esempio l'Address Space Layout Randomization (ASLR). A volte queste funzioni sono facoltative e richiedono un compilatore o un linker switch. In molti casi, queste funzioni sono disponibili gratuitamente e pertanto dovrebbero essere utilizzate. L'ultima serie di controlli utili a contrastare la compromissione della memoria sono le sandbox. Le Sandbox sono funzioni del sistema operativo progettate per proteggere da un programma danneggiato il sistema operativo stesso e il resto dei programmi dell'utente in esecuzione su di esso.
Attacchi di command-injection. Gli attacchi di command-injection (iniezione di comando) sfruttano i dati di input come vettore di attacco (un aggressore fornisce un carattere di controllo, seguito da una serie di comandi). Per esempio, nella SQL injection, un apice chiude spesso un'istruzione dinamica SQL e quando si tratta di script shell unix, la shell può interpretare un punto e virgola come la fine dell'input, prendendo come comando qualsiasi cosa viene dopo.

La seguente tabella riporta le vulnerabilità della Top 10 OWASP 2017[30] riconducibili alle minacce di elevation of privilege, e per ciascuna vulnerabilità indicata, le relative pratiche[31] e requisiti[32] di sicurezza consigliati da OWASP:

OWASP TOP-10 2017 (Rischi di sicurezza delle applicazioni)	**OWASP Proactive Controls 2018 v 3.0** (Pratiche di sicurezza proattive)	**OWASP ASVS 3.0** (Requisiti di sicurezza applicative)
A5 - Broken Access Control	C7 - Enforce Access Controls	V4 - Access Control
A6 - Security Misconfiguration	C7 - Enforce Access Controls	V19 - Configuration
A9 - Using Components with Known Vulnerabilities	C7 - Enforce Access Controls	V13 - Malicious Controls

Tabella 20 - Rischi di sicurezza OWASP relativi all'Elevation Of Privilege

Alcuni esempi di minacce di elevation of privilege:

- Tasti permanenti in Windows: Questo attacco è abbastanza facile da eseguire e non richiede alcun tipo di conoscenza avanzata per poterlo portare. Per eseguire questo attacco è necessario un accesso fisico alla macchina e la possibilità di poterla avviare da un disco di ripristino. Una volta eseguito l'avvio, sarà necessario modificare il file di sistema associato alla funzione del tasto permanente (Sticky Key) (toccando cinque volte il tasto shift). Da prompt dei comandi, si fa una copia del file "sethc.exe" che si trova nella directory "%systemroot%\system32". Dopodiché, tutto ciò che bisogna fare è copiare il file "cmd.exe" nella cartella "%systemroot%\system32" rinominando il file come "sethc.exe". Dopo che l'eseguibile del file "cmd" è stato copiato nella corretta posizione, la macchina viene riavviata. Quando si presenta la schermata di accesso, toccando il tasto shift per cinque volte consecutive i "tasti permanenti" vengono attivati rendendo disponibile una command shell con accesso a livello di sistema. Da questo livello di accesso, un attaccante può creare una backdoor nel sistema e creare un account di amministratore locale.
- Suite Sysinternals su Windows: Un altro modo comune di eseguire escalation dei privilegi in Windows è attraverso l'uso della suite di strumenti Sysinternals. Dopo che un aggressore ottiene una backdoor nel sistema usando il metodo "Sticky Keys" sopra descritto, può aumentare ulteriormente i propri privilegi di accesso al sistema. Questo metodo di attacco richiede l'uso del comando "Psexec" e dei diritti amministrativi locali alla macchina. Dopo aver eseguito il login con l'account di backdoor, attraverso il semplice uso dello strumento "psexec.exe" è possibile aumentare i permessi di accesso al sistema. Ciò lo si può fare utilizzando il comando "psexec.exe -s cmd".
- Process Injection: Lo sfruttamento delle debolezze presenti nei processi è un altro modo per eseguire l'escalation dei privilegi. Ad esempio uno strumento usato per i test di penetrazione come "Process Injector" ha la capacità di enumerare tutti i processi in esecuzione su un sistema e nel contempo di rilevare il relativo account di esecuzione. Per sferrare questo tipo di attacco, è necessario accedere con un account con maggior livello di autorizzazione. Dopo aver identificato il processo che si desidera iniettare, ad esempio, "cmd.exe", è possibile procedere con l'injection eseguendo il comando "pinjector.exe -p [PID] cmd.exe [port]", dove PID rappresenta l'identificativo del processo dal quale si intende copiare i permessi.
- Enumerazione delle Password degli utenti su Linux: Un attacco base di escalation dei privilegi, comune in ambiente Linux, viene condotto attraverso l'enumerazione degli account utente configurati sulla macchina. Questo tipo di attacco richiede che l'attaccante acceda alla shell dei comandi di sistema. Ciò, normalmente lo si può fare attraverso lo sfruttamento di server ftp non

30 https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

31 https://www.owasp.org/index.php/OWASP_Proactive_Controls

32 https://github.com/OWASP/ASVS

correttamente configurati. Una volta che l'attaccante ha ottenuto l'accesso alla shell, eseguendo il comando "cat /etc/passwd | cut -d: -f1" potrà visionare l'elenco di tutti gli utenti della macchina.

- Metasploit su Android: Metasploit è uno strumento ben noto alla maggior parte degli hacker e contiene una libreria di exploit noti. Nel caso di dispositivi Android, Metasploit può essere utilizzato per attaccare i dispositivi Android rooted. Una volta che un dispositivo Android è rooted, su questo viene reso disponibile un file binario "SU" che consente di eseguire comandi come root.
- Utilizzo di file dannosi: Un attacco di questo tipo sfrutta una configurazione di sistema che consente all'attaccante di accedere direttamente a un file eseguibile, ad esempio attraverso l'accesso alla command shell; oppure, nel peggiore dei casi, permette all'attaccante di caricare un file per poi eseguirlo. I server web, i server ftp e i sistemi middleware message oriented che presentano numerosi punti di integrazione possono essere considerati particolarmente vulnerabili, in quanto sia chi sviluppa che gli amministratori devono essere costantemente allineati riguardo le interfacce e i relativi opportuni privilegi.
- Dirottamento dell'esecuzione di un thread privilegiato: Un attaccante può a volte dirottare un thread privilegiato dal sistema sottostante utilizzando metodi sincroni (chiamando una funzione privilegiata che fallisce a seguito di un input non previsto) o asincroni (modificando variabili di ambiente utilizzate dal processo che in qualche modo ne determinano il flusso di esecuzione). Ciò può consentire all'avversario di accedere a funzionalità che il progettista del sistema non aveva previsto, ma può anche consentire all'attaccante di passare inosservato o di negare ad altri utenti servizi essenziali causando seri problemi.
- Elevazione dei privilegi in ambito HTTP: L'utilizzo di metodi standard HTTP (Get, Put, Delete) potrebbero essere esposti verso l'esterno. Un'interpretazione rigorosa del metodo HTTP Get indica che quei servizi che espongono una interfaccia tramite HTTP Get non devono essere usati per cancellare informazioni sul server, ma non esiste alcun meccanismo di controllo degli accessi che garantisce tale logica. Ciò significa che a meno che i servizi non siano correttamente configurati con le opportune ACL e l'implementazione del servizio dell'applicazione segua tale principio, allora una qualsiasi richiesta HTTP può facilmente eseguire una cancellazione o aggiornamento lato server. L'aggressore identifica un URL HTTP Get come http://sitovittima/aggiornamentoOrdine, che a sua volta chiama un altro processo per aggiornare l'ordine presente su un database o per aggiornare un'altra risorsa. Non essendo il metodo Get idempotent (ovvero può essere chiamato più volte ottenendo sempre lo stesso risultato) la richiesta può essere presentata più volte dall'attaccante, inoltre, l'attaccante può essere in grado di sfruttare l'URL pubblicato ed accessibile attraverso il metodo Get per eseguire effettivamente aggiornamenti (invece di recuperare semplicemente dati). Ciò può comportare una modifica dolosa o involontaria dei dati lato server.
- Sovvertimento delle funzionalità di firma del codice: Numerosi linguaggi utilizzano la firma del codice per garantirne l'identità e quindi legare il codice ai privilegi assegnatigli all'interno di un ambiente. Sovvertire tale meccanismo può essere strumentale in un attacco di escalation dei privilegi da parte di un aggressore. Qualsiasi mezzo capace di sovvertire il modo in cui una macchina virtuale impone la firma del codice può essere associato a questo tipo di attacco.
- Programmi target con privilegi elevati: Questo tipo di attacco ha come obiettivo quei programmi che funzionano con privilegi elevati. L'attaccante potrebbe tentare di sfruttare un bug presente nel programma in esecuzione e di ottenere del codice arbitrario da eseguire con privilegi elevati. Per esempio l'attaccante potrebbe individuare delle vulnerabilità in quei programmi che scrivono nelle directory di sistema o nelle chiavi di registro (come la posizione di registro "HKEY_LOCAL_MACHINE", dove vengono memorizzate una serie di variabili critiche dell'ambiente Windows). Questi programmi vengono tipicamente eseguiti con privilegi elevati e di solito non sono stati progettati pensando agli aspetti di sicurezza. Tali programmi sono gli obiettivi perfetti per uno sfruttamento in quanto forniscono notevoli vantaggi all'attaccante quando vengono compromessi. L'utente malintenzionato cerca di eseguire del codice arbitrario con lo stesso livello di autorizzazione di una chiamata di sistema privilegiata.

- Cross Zone Scripting: Un aggressore è in grado di indurre una vittima designata a caricare contenuti nel proprio browser web che bypassa i controlli delle zone di sicurezza ottenendo così l'accesso per l'esecuzione con maggiori privilegi del codice di scripting o di altri oggetti web come i controlli ActiveX non firmati o gli applet. Si tratta di un attacco di elevazione dei privilegi mirato alla sicurezza dei browser web la cui sicurezza è basata su zone. In un modello basato su zone, le pagine appartengono a una delle zone corrispondenti al livello di privilegio assegnato a quella pagina. Le pagine in una zona non attendibile avrebbero un livello inferiore di accesso al sistema e/o sarebbero limitate nelle tipologie di contenuto eseguibile che queste sono autorizzate ad invocare. In un attacco di questo tipo, ad una pagina che dovrebbe essere assegnata ad una zona meno privilegiata vengono concessi i privilegi di una zona ritenuta maggiormente affidabile. Questo lo si può fare, sfruttando i bug presenti nel browser web o una configurazione errata nei controlli di zona, attraverso un attacco di cross-site scripting che fa sì che il contenuto dell'aggressore venga trattato come proveniente da una pagina attendibile. Questo attacco si differenzia dal "Restful Privilege Escalation" in quanto quest'ultimo minaccia lato server, l'inadeguata sicurezza dei metodi di accesso RESTful (come HTTP DELETE), mentre il "cross zone scripting" minaccia lato client, il concetto di "zone di sicurezza" implementato dal browser.
- Dirottamento di un processo privilegiato: Un aggressore ottiene il controllo di un processo a cui sono assegnati privilegi elevati per eseguire del codice arbitrario. Solitamente sul sistema operativo, ad alcuni processi vengono assegnati privilegi elevati tramite l'associazione ad un particolare utente, gruppo o ruolo. Se un aggressore è in grado di dirottare il processo, questo a sua volta sarà in grado di assumerne il livello di privilegi per eseguire il codice a sua scelta. I processi possono essere dirottati attraverso una gestione impropria dell'input da parte dell'utente (ad esempio, un buffer overflow o alcuni tipi di attacchi di iniezione) o utilizzando utility di sistema non adeguatamente protette che supportano il controllo del processo.

5.5.4.2 Attack tree

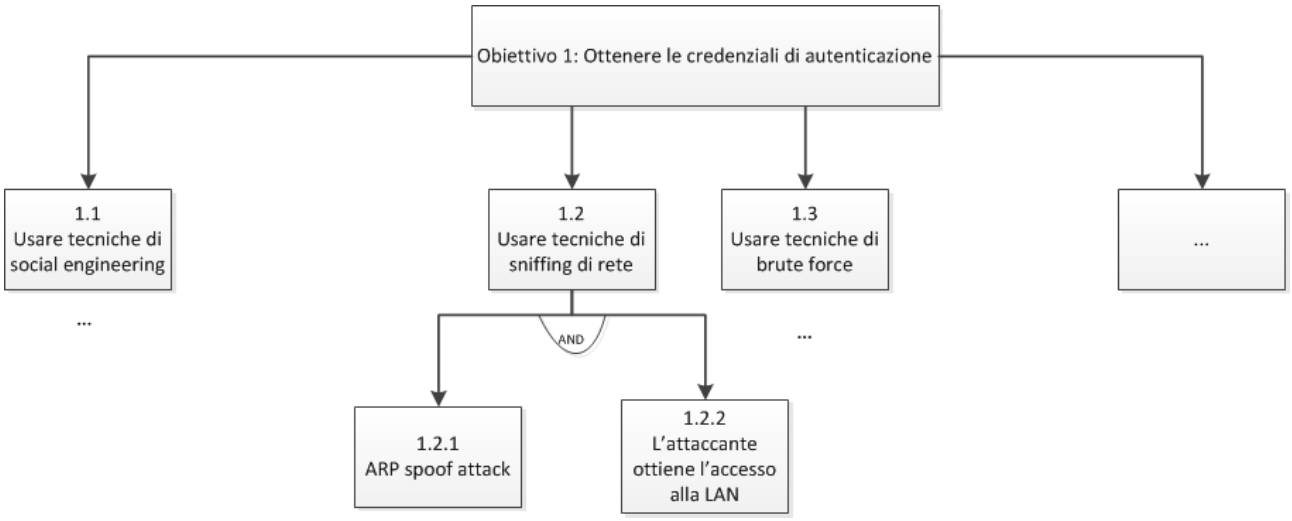
L'attack tree rappresenta un'ulteriore metodologia per raccogliere e documentare i potenziali attacchi in modo strutturato e gerarchico.
Un attack tree è modellato attraverso una struttura ad albero i cui elementi base sono:

- Il nodo radice che rappresenta *l'obiettivo finale* dell'attaccante,
- I nodi figli che rappresentano i *sotto-goals* che concorrono al raggiungimento dell'*obiettivo finale*,
- Le foglie che rappresentano gli *attacchi*.

La modellazione segue la seguente logica di base:

- I "nodi OR" indicano le alternative, ossia modi indipendenti, per raggiungere un goal/sotto-goal.
- I "nodi AND" indicano i passi che concorrono al raggiungimento dello stesso goal/sotto-goal.

PASSO 1 – Modellazione degli attacchi
Si modellano quindi, tanti attack tree quanti sono gli obiettivi di un attaccante. In questa fase l'analista deve identificare i possibili obiettivi di attacco e, per ciascuno di essi, i passi attraverso cui realizzarli (la capacità di identificazione degli obiettivi e dei passi è quindi assolutamente soggettiva).
Si riporta di seguito, un esempio di modellazione di attack tree:



Nell'esempio in figura:

- Il nodo radice rappresenta l'obiettivo finale dell'attaccante che consiste nell' "Ottenere le credenziali di autenticazione",
- L'obiettivo finale può essere raggiunto in vari modi ("OR"):

  1.1 - "Usare tecniche di social engineering",
  1.2 - "Usare tecniche di sniffing di rete",
  1.3 - "Usare tecniche di brute force", ecc.
- Sulla base della modalità di Livello-1 scelta, saranno diversi gli elementi (nodi foglie) che concorreranno al raggiungimento dell'obiettivo radice. Ad esempio, nel caso si scelga il percorso 1.2 - "Usare tecniche di sniffing di rete" l'obiettivo si raggiunge (sub-goal) attraverso 2 step correlati ("AND"):

  1.2.1 - "ARP spoof attack";
  1.2.2 - "L'attaccante ottiene l'accesso alla LAN".

Lo stesso attack tree può anche essere rappresentato in forma testuale:

Obiettivo 1 - Ottenere le credenziali di autenticazione
1.1 - Usare tecniche di social engineering OR
...
1.2 - Usare tecniche di sniffing di rete OR
1.2.1 - ARP spoof attack AND
1.2.2 - L'attaccante ottiene l'accesso alla LAN
1.3 - Usare tecniche di brute force OR
...
...

PASSO 2 – Analisi degli attributi di sicurezza

Dopo aver modellato i possibili attacchi al sistema, è necessario analizzare gli attributi di sicurezza del sistema quali, ad esempio:

1. la possibilità o l'impossibilità dell'attacco (P=Possibile, I=Impossibile)
2. il costo dell'attacco (valore in euro, es. 10K)
3. gli strumenti necessari per realizzare l'attacco (AS=Attrezzature Specifiche, SAS=Senza Attrezzature Specifiche)

Per determinare il costo di un attacco:

1. si determina il valore per ciascun nodo foglia.

2. si determina il valore dei nodi non foglia. Questo è ricavato a partire dal valore dei loro figli.

L'analista dovrà avere la capacità di attribuire dei "buoni" valori per i nodi foglia (anche in questa fase la capacità di identificazione dei valori è assolutamente soggettiva).

PASSO 3 – Analisi dei risultati

L'analisi è volta a determinare:

1. il costo dell'attacco più economico (si analizzano i valori degli attributi a livello del nodo radice);
2. gli attacchi il cui costo non supera una certa soglia (si analizzano i sotto-alberi i cui nodi rappresentano una determinata proprietà/soglia);
3. l'attacco più economico che non utilizza attrezzature speciali (si analizzano i sotto-alberi i cui nodi rappresentano un determinato insieme di proprietà).

L'analisi dei risultati deve tenere conto delle caratteristiche del potenziale attaccante in quanto queste determinano quali parti dell'attack tree devono essere prese in considerazione. Attaccanti diversi hanno diversi livelli di abilità, accesso, avversione al rischio, soldi e così via. Se il potenziale attaccante è la criminalità organizzata, occorre considerare la possibilità di attacchi costosi e "mezzi illeciti" (corruzione, ricatto, ecc.) che espongono l'attaccante fino al rischio di andare in prigione. Se il potenziale attaccante è un terrorista, occorre considerare "mezzi illeciti" che espongono l'attaccante fino al rischio di morire per raggiungere l'obiettivo. Se l'attaccante è un casual hacker si esclude la possibilità di attacchi costosi e "mezzi illeciti" (corruzione, ricatto, ecc.).

PASSO 4 – Analisi What-if

L'analisi "what if" è costruita a partire dalle diverse ipotesi di adozione delle contromisure. Introducendo contromisure, cambiano i valori attribuiti nel secondo step della metodologia e quindi cambiano i risultati dell'analisi.
In conclusione, una delle caratteristiche di valore degli attack tree è che sono riutilizzabili. Tornando all'esempio inziale, una volta completato l'albero di attacco relativo a "Ottenere le credenziali di autenticazione", è possibile utilizzarlo in qualsiasi situazione in cui sia interesse dell'attaccante l'acquisizione delle credenziali di autenticazione. L'attack tree relativo a "Ottenere le credenziali di autenticazione" può inoltre diventare parte di un attack tree più grande.

5.5.4.3 TRIKE

TRIKE è un framework open-source per l'auditing della sicurezza da un punto di vista di risk management basato sulla generazione di modelli di minacce in modo affidabile e ripetibile. Il progetto ebbe inizio nel 2005 come tentativo di migliorare l'efficienza e l'efficacia delle esistenti metodologie di modellazione delle minacce e da allora viene attivamente aggiornato e utilizzato. I creatori di TRIKE hanno anche sviluppato strumenti a supporto di questa metodologia come il foglio di calcolo TRIKE. Questo strumento si focalizza sull'automazione della generazione delle minacce e non prevede alcun brainstorming. I team di sicurezza non hanno la necessità di individuare le possibili minacce in quanto tali minacce sono già predefinite. TRIKE può essere utilizzato anche da uno sviluppatore di sicurezza inesperto per trovare le vulnerabilità di sicurezza in modo efficace ed affidabile. Il team TRIKE ha adottato l'analisi HAZOP (Hazardous Operations), ovvero, un metodo sistematico per identificare quali variazioni di processo devono essere mitigate. Questo framework può sostituire gli alberi di minaccia e d'attacco (attack tree).
TRIKE utilizza un approccio basato sul rischio con distinte implementazioni dei modelli di minacce e rischi. Approccia al Threat Modeling assumendo una posizione difensiva rispetto a quella di un attaccante. TRIKE ha anche proposto il concatenamento delle minacce, un'alternativa agli alberi delle minacce, nel tentativo di ridurre la natura ripetitiva di quest'ultimi.

5.5.4.4 P.A.S.T.A (Process for Attack Simulation and Threat Analysis)

Si tratta di un processo agnostico rispetto alla piattaforma, suddiviso in sette fasi distinte e applicabile alla maggior parte delle metodologie di sviluppo. Il suo obiettivo principale è quello di affrontare le minacce più gravi per un determinato obiettivo applicativo. Di seguito in elenco le fasi di cui sopra:

- Definizione degli obiettivi di business;
- Definizione dell'ambito tecnico;
- Scomposizione del sistema;
- Analisi delle minacce;
- Rilevamento della vulnerabilità;
- Enumerazione degli attacchi;
- Analisi del rischio delle minacce individuate e valutazione del relativo impatto[33].

Il processo è una combinazione di diversi approcci di modellizzazione delle minacce, definisce gli obiettivi di business, i requisiti di sicurezza e conformità e l'analisi dell'impatto delle minacce sul business. Similmente al processo adottato da Microsoft, nella rappresentazione del sistema, l'applicazione viene ridotta in componenti discreti attraverso l'utilizzo di casi d'uso e DFD. Durante l'analisi delle minacce e delle vulnerabilità vengono utilizzati anche gli alberi delle minacce e i casi di abuso. Si calcola quindi l'impatto delle minacce sul rischio e sul business, vengono quindi individuate le necessarie contromisure.

5.5.4.5 Best practices di carattere generale

- VALIDARE E NON BONIFICARE - Dobbiamo sapere cosa e quanto ci aspettiamo di ricevere e dobbiamo convalidare ciò che riceviamo. Se si riceve qualcos'altro rispetto al previsto, è necessario scartarlo restituendo un messaggio di errore. A meno che il nostro codice non sia perfetto, eventuali errori di bonifica potrebbero produrre ancor più danno, in quanto, dopo aver scritto la funzione di bonifica dell'input si farebbe esclusivo affidamento su di essa.
- AFFIDARSI AL SISTEMA OPERATIVO - Affidarsi al sistema operativo è una buona pratica per i seguenti motivi:
    - Il sistema operativo mette a disposizione funzionalità di sicurezza che consentono di concentrarsi sui propri obiettivi.
    - Il sistema operativo funziona con privilegi che probabilmente non sono disponibili per il programma o per un eventuale aggressore.
    - Se l'aggressore controlla il sistema operativo, è probabile che abbiamo problemi ben più gravi, indipendentemente da ciò che il codice tenta di fare.

  A seguito di quanto sopra indicato, ci si potrebbe chiedere: "quale bisogno c'è di modellare le minacce? Perché non affidarsi solo al sistema operativo?". Ebbene, sta a noi verificare di non impostare le autorizzazioni su un file a 777, o impostare gli ACL in modo tale da non consentire la scrittura agli account "guest". Sta a noi scrivere codice che funzioni bene con i permessi di un utente normale piuttosto che con i permessi di un utente sandboxed.
- FILE BUGS - È bene includere le minacce tra i bugs e contrassegnarle come bug di sicurezza. Bisogna inoltre dare una priorità a questi bugs. I bugs di tipo "Elevation-of-privilege" sono quasi sempre riconducibili alla categoria di più alta priorità poiché, quando vengono sfruttati causano molti danni. I bugs di tipo "Denial of service" spesso vanno considerati per ultimi.

[33] Da non confondersi con l'attività di Risk Assessment per la quale si deve far riferimento alla metodologia e al tool sviluppato da AGID a tale scopo (**Cyber Risk Management** - **https://www.sicurezzait.gov.it/Home**). Per ulteriori dettagli si rinvia all' *Allegato 1- Linee Guida per l'adozione di un ciclo di sviluppo di software sicuro*.

- VERIFICA DEL LAVORO SVOLTO - La convalida del modello è l'ultima cosa da fare come parte della modellazione delle minacce. Ci sono alcune attività da svolgere prima, ed è bene mantenerle allineate con l'ordine in cui è stato svolto il lavoro precedente. Pertanto, le attività di validazione includono il controllo del modello, la verifica di ciascuna minaccia e il controllo dei test. E' probabile che si desideri convalidare il modello anche una seconda volta, ovvero quando ci si avvicina al rilascio o all'installazione.
- CONTROLLO DEL MODELLO - È necessario assicurarsi che il modello finale corrisponda a quello costruito. Se così non fosse, come potremmo sapere se le minacce trovate sono corrette e rilevanti? Per fare ciò, è opportuno organizzare degli incontri durante i quali tutti, osservando e analizzando il diagramma, rispondano alle seguenti domande:
    - il diagramma è completo?
    - il diagramma è accurato?
    - il diagramma copre tutti le decisioni intraprese in termini di sicurezza?
    - è possibile procedere con la versione successiva del diagramma senza apportare modifiche?

  Una risposta affermativa a tutte le domande di cui sopra, indica che il diagramma è sufficientemente aggiornato e maturo per poter procedere. Diversamente sarà necessario apportare gli opportuni cambiamenti.
- DETTAGLI DEL DIAGRAMMA - Non disegnare mai un diagramma ad occhio, con un livello di dettaglio tale da rappresentare l'intero comportamento del sistema. Utilizzare un sotto diagramma che mostri il solo dettaglio di una particolare area del sistema stesso. Si deve cercare di escludere ciò che non è rilevante per il progetto. Per esempio, se si è davanti ad un processo molto complesso, forse tutto ciò che è in quel processo dovrebbe essere trattato in un diagramma, e tutto ciò che è al di fuori di esso in un altro. Se è presente un dispatcher o un sistema di code, questi sono un buon punto di suddivisione, e lo sono anche i database o i sistemi di fail-over. Esistono ancora elementi che devono essere maggiormente dettagliati? Bene, questi vanno esclusi. La cosa importante da ricordare è che il diagramma ha lo scopo di aiutare a comprendere e discutere il sistema.

5.6 Indirizzamento delle minacce

Una volta raccolte le minacce individuate in uno o più elenchi, il passo successivo nel processo di modellazione è quello di scorrere l'elenco o gli elenchi indirizzando ciascuna minaccia. Pertanto è possibile decidere se:

- **Mitigare** la minaccia - Si concretizza nel fare qualcosa per rendere più difficile la possibilità di poter essere sfruttata. Richiedere una password per controllare chi accede al sistema, mitiga la minaccia di spoofing. Aggiungere un controllo password che ne rafforza la complessità o la scadenza, riduce la probabilità che una password venga scoperta o venga utilizzata se rubata.
- **Eliminare** la minaccia - Avviene quasi sempre eliminandone le caratteristiche. Se si presenta una minaccia in cui qualcuno ha accesso ad una funzione amministrativa di un sito web entrando ad esempio in "/admin/URL", è possibile mitigarla impiegando delle password o altre tecniche di autenticazione, ma comunque, questa non verrà risolta. È possibile rendere più complessa la URL in modo da rendere meno probabile la possibilità che questa venga individuata, ma anche in questo caso la minaccia non verrà risolta. La si può eliminare rimuovendo l'interfaccia amministrativa, e gestendo l'attività di amministrazione tramite linea di comando. In questo caso esisterebbero comunque altre minacce riconducibili a come l'utente amministratore dovrebbe autenticarsi per eseguire l'attività di amministrazione da linea di comando. Lo spostamento dell'interfaccia dall'http a linea di comando rende facile la mitigazione della minaccia controllando la superficie di attacco.
- **Spostare** la minaccia - Consiste nel lasciare che qualcuno o qualcos'altro gestisca il rischio. Per esempio, potremmo demandare la gestione delle minacce relative all'autenticazione al sistema operativo, oppure rafforzare il perimetro di fiducia con un firewall. È anche possibile trasferire il

rischio direttamente all'utente utilizzatore, chiedendogli di navigare attraverso una moltitudine di finestre di dialogo incomprensibili prima che questo possa effettivamente utilizzare il sistema. Ovviamente questa non vuole essere assolutamente una tra le migliori soluzioni, ma in alcuni casi esiste da parte degli utilizzatori una conoscenza tale da poterli rendere partecipi per convenire ad un giusto compromesso di sicurezza. Se si pensa che esistano i presupposti per una soluzione del genere, si dovrebbe sostenere chi usa il sistema a prendere una decisione in tal senso.

- **Accettare** la minaccia - È l'ultimo approccio per indirizzare una minaccia. In alcuni casi, il costo necessario per impedire a qualcuno di inserire una back-door nella scheda madre di un hardware aziendale potrebbe risultare elevato, quindi in tal caso si potrebbe scegliere di accettare il rischio. Una volta che questo viene accettato, non c'è più bisogno di preoccuparsene. A volte la preoccupazione indica che il rischio non è stato pienamente accettato o che l'accettazione del rischio non sia appropriata.

5.7 Valutazione del rischio: tecniche di Risk Ranking

5.7.1 DREAD

Microsoft ha sviluppato la metodologia DREAD (tabella che segue) per valutare ciascun rischio individuato durante l'attività STRIDE. Ad ogni rischio viene assegnato un punteggio DREAD da parte del team di sicurezza/sviluppo i quali realizzano e applicano il modello delle minacce.
Esistono diverse varianti del sistema di valutazione e prioritizzazione del rischio:

DREAD	DESCRIZIONE
Damage potential	Classifica l'estensione del danno che si verifica se viene sfruttata la vulnerabilità individuata.
Reproducibility	Classifica quanto spesso un tentativo di sfruttamento della vulnerabilità individuata viene portato a termine con successo.
Exploitability	Assegna un valore numerico allo sforzo necessario per sfruttare la vulnerabilità individuata. Inoltre, la possibilità di sfruttamento considera come condizioni preliminari che l'utente deve essere autenticato.
Affected Users	Assegna un valore numerico che rappresenta la numerosità degli utenti del sistema che potrebbero essere interessati se un exploit divenisse ampiamente disponibile.
Discoverability	Misura la probabilità che la vulnerabilità possa essere individuata da soggetti esterni della sicurezza e/o dagli hacker, se questa non viene risolta tramite patch.

Tabella 21 - Modello DREAD

Nella valutazione del rischio ad ogni componente DREAD viene assegnato un punteggio. I punteggi dei singoli componenti vengono quindi calcolati per dare un 'punteggio DREAD' totale.
Il rischio viene quindi determinato in base al valore che il punteggio "DREAD" assume rispetto ad intervalli di valori predefiniti. Il risultato finale è un elenco di vulnerabilità classificate per rischio.
Il processo di applicazione della metodologia DREAD è estremamente soggettivo e richiede le necessarie competenze. È consigliabile avere almeno un membro del team che abbia competenze sulla sicurezza per dare il necessario supporto nell'assegnazione dei punteggi DREAD.
Come fase finale del processo di modellazione delle minacce, viene attuata una valutazione del ***rischio***[34], per dare una priorità a ciascuna vulnerabilità indentificata.

[34] Da non confondersi con l'attività di Risk Assessment per la quale si deve far riferimento alla metodologia e al tool sviluppato da AGID a tale scopo (**Cyber Risk Management** - **https://www.sicurezzait.gov.it/Home**). Per ulteriori dettagli si rinvia all' *Allegato 1- Linee Guida per l'adozione di un ciclo di sviluppo di software sicuro*.

In generale, in termini quantitativi, il rischio è definito come il prodotto tra la probabilità di accadimento dell'evento e l'impatto:

Rischio = Probabilità x Impatto

In effetti, se almeno uno dei due termini del prodotto tende a zero, percepiamo il rischio come basso. Viceversa percepiamo un rischio grave quando ambedue i termini sono elevati.
Nella metodologia DREAD il concetto di "impatto" viene declinato in termini di:

1. danno (Damage)
2. utenti interessati (Affected Users)

mentre il concetto di "probabilità" viene declinato in termini di:

3. riproducibilità (Reproducibility),
4. sfruttabilità (Exploitability)
5. rilevabilità (Discoverability).

DREAD è appunto l'acronimo che "fattorizza" il rischio rispetto a queste 5 distinte categorie che caratterizzano la minaccia:

1. **Damage potential**: Quanto sarebbe rilevante il danno nel caso in cui la minaccia[35] si concretizzasse?
2. **Reproducibility**: Quanto è facile che la minaccia possa ripetersi?
3. **Exploitability**: Quanto tempo, sforzo e conoscenza sono necessarie per concretizzare con successo la minaccia?
4. **Affected Users**: Nel caso in cui la minaccia si concretizzi, quale percentuale di utenti sarebbe coinvolta?
5. **Discoverability**: Quanto è facile per un attaccante scoprire la possibilità di minaccia?

A ciascuna categoria viene attribuito un peso. Il "DREAD score" è la media dei 5 pesi, ossia:

DREAD Score= (Damage + Reproducibility + Exploitability + Affected Users + Discoverability) / 5

Occorre quindi valutare e dare un peso numerico alle cinque categorie della tabella sopra mostrata. A seconda del dominio considerato, ci si può riferire o a una scala (semplificata) di tre soli valori o a una scala (più granulare) a dieci valori. I valori crescono rispettivamente al crescere del danno, della facilità di riproduzione, della facilità di sfruttamento, del numero di utenze coinvolte, della facilità di rilevamento.
A titolo di esempio, si consideri la categoria "Exploitability":

- nel caso si voglia adottare una scala a 3 valori si potrebbero voler considerare e pesare in modo diverso i seguenti casi (Quanto è difficile sfruttare la vulnerabilità?):
    - 1= L'attacco richiede una figura senior e una conoscenza profonda del sistema attaccato; un'utenza con diritti di amministrazione; dispendio di parecchie risorse per organizzare l'attacco (es. impiego di custom tool);
    - 2 = L'attacco richiede una figura senior; un'utenza autenticata con abilità non amministrative; tool di attacco disponibili in rete;
    - 3 = L'attacco è alla portata di una figura junior; senza necessità di autenticazione; attraverso un web browser.

35 Si ricordi che la minaccia è un evento potenziale, accidentale o deliberato. Se deliberato, la minaccia si configura come attacco.

- nel caso si voglia adottare una scala a 10 valori si potrebbero voler considerare e pesare in modo diverso i seguenti casi (Quanto è difficile sfruttare la vulnerabilità?):

    - 1 = Anche con conoscenza approfondita della vulnerabilità non si individua un percorso di attacco valido per l'exploit;
    - 2 = Sono richieste tecniche avanzate e tool custom. Sfruttabile solo dagli utenti autenticati;
    - 5 = La possibilità di exploit esiste, alla portata di skill medie da un'utenza autenticata;
    - 7 = La possibilità di exploit esiste, alla portata di un'utenza non autenticata;
    - 10 = Banale: basta un web browser.

Nel primo caso il calcolo del DREAD score:
DREAD Score= (Damage + Reproducibility + Exploitability + Affected Users + Discoverability) / 5
ricade in un numero compreso tra 1 e 3. Nel secondo caso ricade in un numero compreso tra 1 e 10. Il risultato finale è, in entrambi i casi, come desiderato, un elenco di vulnerabilità classificate per rischio ossia ordinate per priorità di intervento (a valori bassi corrisponde priorità bassa, a valori alti priorità alta).

5.7.2 Security Bulletin Severity Rating System (S.B.S.R.S)

Come alternativa alla metodologia di analisi DREAD, Microsoft si avvale anche del "Security Bulletin Severity Rating System" per valutare le vulnerabilità nei prodotti Microsoft. Il sistema di classificazione raggruppa le vulnerabilità in una delle quattro categorie sotto riportate. Microsoft utilizza questo sistema per valutare la necessità di patch di sicurezza per i propri prodotti.

CLASSIFICAZIONE	DEFINIZIONE
CRITICA	Una vulnerabilità il cui sfruttamento potrebbe consentire l'esecuzione di codice senza alcuna interazione da parte dell'utente.
IMPORTANTE	Una vulnerabilità il cui sfruttamento potrebbe compromettere la riservatezza, l'integrità o la disponibilità dei dati degli utenti o l'integrità o la disponibilità delle risorse necessarie all' elaborazione.
MODERATA	La sfruttabilità viene mitigata in misura significativa da fattori quali l'adozione della configurazione predefinita, l'auditing o la difficoltà di sfruttamento.
BASSA	Una vulnerabilità il cui sfruttamento è estremamente difficile, o il cui impatto è minimo.

Tabella 22 - Sistema di classificazione del S.B.S.R.S.

Questo sistema di rating può anche essere adattato per classificare le vulnerabilità identificate dalla STRIDE. L'obiettivo finale è quello di produrre un elenco di prioritizzazione delle vulnerabilità a supporto del processo decisionale. Sebbene Microsoft non utilizzi più l'analisi DREAD, le motivazioni potrebbero essere giustificate dal fatto che si tratta di un processo di valutazione del rischio più adatto ai non esperti di sicurezza o a team novizi nella modellazione delle minacce rispetto al Security Bulletin Severity Rating System. L'analisi DREAD prevede un calcolo basato sulla valutazione di diversi aspetti della vulnerabilità, quali il danno potenziale o la riproducibilità. Il Security Bulletin Severity Rating System classifica le vulnerabilità in una di quattro categorie.

Entrambi si basano sul parere del singolo individuo/team che effettua la valutazione del rischio. La DREAD ha perfezionato il suo sistema di valutazione del punteggio riducendo la probabilità di variazione dei risultati ed offre un approccio strutturato che i team possono adottare. Il processo è inoltre ben documentato e relativamente facile da attuare. Questa è una delle possibili motivazioni per cui scegliere l'analisi DREAD rispetto a S. B. S. R. S.

5.7.3 Altri processi di valutazione del rischio

Esistono altre opzioni disponibili per la valutazione del rischio, come l'US-CERT Vulnerability Metric, che utilizza una metrica quantitativa e valuta la gravità di una vulnerabilità assegnandole un valore compreso tra 0 e 180. Il Common Vulnerability Scoring System (CVSS) mira a fornire un framework open per misurare l'impatto delle vulnerabilità IT, mentre la scala SANS Critical Vulnerability Analysis classifica le vulnerabilità utilizzando diversi fattori chiave e variando il grado di peso. Il processo di valutazione del rischio da adottare dipende essenzialmente dalla scelta individuale del team che realizza il Threat Model.

5.8 Privacy by Design

5.8.1 Introduzione e concetti base

Il regolamento generale sulla protezione dei dati personali (UE) n. 679/2016 ("GDPR") è entrato in vigore, a partire dal 25 maggio 2018, come principale quadro giuridico in materia di protezione dei dati nell'UE direttamente applicabile in tutti gli Stati membri, abrogando la direttiva 95/46/CE sulla protezione dei dati. In particolare, in Italia, il decreto legislativo n. 101/2018 adegua la normativa nazionale al nuovo Regolamento, con l'entrata in vigore del 19 settembre 2018. Il regolamento prevede un'armonizzazione del regime giuridico di protezione dei dati in tutta l'UE, rafforzando i diversi principi e obblighi della direttiva che abroga e introduce nuove disposizioni quali la protezione dei dati per default e a partire dalla progettazione. Al fine di migliorare la trasparenza delle operazioni di trattamento da parte dei responsabili del trattamento e degli incaricati del trattamento, il regolamento introduce, inoltre, disposizioni specifiche in materia di certificazione, sigilli e marchi.
La garanzia di sicurezza di un'organizzazione rappresenta la base per la protezione della privacy degli individui. La privacy può essere compromessa a causa di un errore nella sicurezza, tuttavia, la privacy si riferisce all'utilizzo improprio e non, delle informazioni da parte di utenti autorizzati. La privacy è una politica di gestione delle informazioni più che una politica di controllo accesso. Insieme, privacy e sicurezza, rappresentano la base per creare un solido rapporto di fiducia. Una sicurezza solida è la base della protezione della privacy. La privacy richiede una sicurezza effettiva, ma quest'ultima da sola, non garantisce una privacy effettiva. La sicurezza garantisce ad esempio il controllo accessi alle risorse di un'organizzazione attraverso un'istruzione di controllo accessi come: all'entità X è consentito eseguire l'operazione Y sulla risorsa Z che tradotta in un esempio pratico, potrebbe essere: "Il personale del settore finanziario può interrogare la base dati della contabilità". La privacy si riferisce alla relazione tra un'organizzazione che raccoglie informazioni e il proprietario delle informazioni raccolte. Per stabilire e gestire questa relazione, è necessario creare una politica di riservatezza formata da diverse istruzioni. Un'istruzione della politica di riservatezza definisce:

- I tipi di informazioni raccolte e quelle accessibili;
- Chi può accedere alle informazioni raccolte;
- Per quali scopi è possibile accedere a tali informazioni.

Suddetta istruzione può assumere la seguente forma: all'entità X è consentito eseguire l'operazione Y sulla risorsa protetta Z del proprietario A per lo scopo B solo se il proprietario A esprime il proprio consenso esplicito. Una possibile traduzione di tale concetto in un esempio pratico,

potrebbe essere: "I membri del settore di ricerca e sviluppo possono consultare le informazioni riguardanti le abitudini di utilizzo di un prodotto di un singolo individuo per lo sviluppo di un nuovo prodotto se il soggetto in questione ha espresso esplicitamente il proprio consenso a rilasciare i propri dati in tal senso". Da notare che questa istruzione comprende la scelta da parte del proprietario delle informazioni di stabilire come utilizzare le proprie informazioni. È possibile, inoltre, applicare anche altre dipendenze per l'accesso alle informazioni personali. Ad esempio, una norma può imporre alcune restrizioni sull'utilizzo dei dati raccolti.
La privacy by Design è emersa come un approccio proattivo, integrativo e creativo per rafforzare i requisiti di privacy sin dalle prime fasi della progettazione applicativa. Tra le sfide legate all'ingegneria della privacy by design vi è la mancanza di metodologie olistiche, sistematiche e integrative che affrontino la complessità e la variabilità della privacy e sostengano la traduzione dei suoi principi fondamentali nelle attività di ingegneria. Per certi versi questo è comprensibile poiché l'approccio è stato sviluppato per tener conto di una serie di fonti normative e standard. Tuttavia, ne consegue che i suoi principi fondanti sono dati ad un alto livello di astrazione senza fornire a corredo metodologie e linee guida per ottenere requisiti concreti in materia di privacy. I principi fondamentali della Privacy by Design si basano sui Fair Information Practice Principles (FIPP) e fungono da framework universale per l'integrazione della privacy in tre principali aree di applicazione: tecnologie dell'informazione e della comunicazione, aree di business, progetti fisici e infrastrutturali.
La Privacy Engineering si è affermata come una nuova disciplina che mira ad applicare principi e processi di ingegneria nello sviluppo, nell'implementazione e manutenzione dei sistemi, in modo sistematico e ripetibile, per raggiungere un livello accettabile di protezione della privacy. Per distinguere tali concetti; la Privacy by Design (PbD) intende spiegare "Cosa fare" per raggiungere un livello adeguato di protezione della privacy, mentre la Privacy Engineering (PE) intende spiegare "Come farlo" definendo la privacy come un attributo di qualità nell'ingegneria dei sistemi. In altre parole, la PE si concentra sullo sviluppo e la valutazione di metodi, tecniche e strumenti che identificano e affrontano in modo sistematico le problematiche legate alla privacy durante il processo di sviluppo dei sistemi.
La tabella che segue, sintetizza i 'concetti' alla base della Privacy:

Concetto di Privacy	Descrizione
Personal Data	Informazioni che possono essere ricondotte ad un individuo.
Identifiable Natural Person	Individuo collegato al "Personal Data".
Item of Interest (IOI)	Informazioni relative ad un individuo (ad esempio soggetti, messaggi, azioni, ecc.).
Unlinkability	Impossibilità di distinguere se due IOI sono correlati.
Anonymity	Impossibilità di identificare il soggetto all'interno di un gruppo di soggetti.
Plausible deniability	Possibilità di negare di aver eseguito un'azione.
Undetectability	Impossibilità di distinguere se esiste un IOI.
Unobservability	Impossibilità nell'essere rintracciabili da tutti i soggetti coinvolti.
Confidentiality	Restrizioni autorizzative all'accesso e alla divulgazione delle informazioni.
Awareness	Essere consapevoli delle conseguenze della condivisione delle informazioni personali (vedi sopra, Personal Data).
Compliance	Aderenza alle normative e alle politiche interne di una organizzazione.

Tabella 23 - Concetti alla base della Privacy

Similmente alla miriade di normative sulla privacy disponibili, ci sono stati diversi tentativi di strutturare e classificare i concetti di privacy. Di seguito si riportano alcuni esempi di tassonomie basate su due approcci distinti:

- Classificazione dei concetti di privacy da un punto di vista giuridico;
- Classificazione dei concetti di privacy da un punto di vista di ingegneria del software.

Tassonomia di Solove. Presenta una tassonomia delle violazioni della privacy da un punto di vista legale. Anche se questa non tratta la privacy digitale, ma descrive la privacy in generale, fornisce comunque alcune informazioni utili in materia. Solove[36] opera una distinzione tra quattro gruppi di attività di base dannose:

- Raccolta dei dati. Include due tipi di violazioni della privacy: il controllo inteso come "osservazione, ascolto o registrazione delle attività di un individuo", e, l'investigazione che consiste in varie forme di sondaggio per ottenere informazioni.
- Trattamento dei dati. Include cinque tipi di violazioni dei dati raccolti al punto precedente: aggregazione (ovvero combinazione di dati relativi a un individuo), identificazione (ovvero collegamento dei dati per identificare un individuo), negligenza (poca attenzione nella protezione dei dati memorizzati), uso secondario (ovvero utilizzo dei dati per scopi diversi da quelli per i quali sono stati raccolti) ed esclusione (ovvero quando l'interessato non è a conoscenza di dati che lo riguardano che gli altri posseggono).
- Diffusione dei dati. Include sette categorie di violazioni: violazione della riservatezza (ovvero non mantenere riservate le informazioni di una persona), divulgazione (cioè rivelare informazioni "sensibili" veritiere su una persona), esposizione (cioè rivelare le nudità, il dolore o le caratteristiche fisiche di una persona), maggiore accessibilità (cioè amplificare l'accessibilità dei dati), appropriazione (cioè l'uso della propria identità per perseguire un'altra finalità).
- Invasione. A differenza dei gruppi precedenti, non riguarda necessariamente le informazioni personali, ma degli elementi che limitano la sfera personale e decisionale (ad esempio atti invasivi che violano la tranquillità di una persona e che impattano sulle decisioni private di una persona).

Linee guida FIPPs (Fair Information Practice Principles). La Privacy and Personal Information Protection raccoglie un insieme di indicazioni proposte dalla Federal Trade Commission degli Stati Uniti. Queste possono essere considerate come il fondamento di tutta la legislazione vigente negli Stati Uniti, in materia di protezione dei dati. Sono state utilizzate per la definizione delle linee guida dell'OCSE (Organizzazione per la Cooperazione e lo Sviluppo Economico) e per la direttiva europea sulla protezione dei dati. I principi si basano su cinque distinte categorie:

- Avviso/Consapevolezza. I soggetti interessati dovrebbero essere adeguatamente informati prima di procedere nella raccolta delle informazioni personali che li riguardano.
- Scelta/Consenso. I soggetti interessati devono essere in grado di scegliere come devono essere utilizzati i propri dati personali. In particolare laddove si fa un uso secondario dei dati (ad esempio, registrazione ad una mailing list o trasferimento di informazioni a terzi).
- Accesso/Partecipazione. Una persona dovrebbe essere in grado di accedere ai dati che la riguardano e di contestarne l'accuratezza e la completezza.
- Integrità/Sicurezza. I dati devono essere accurati e protetti.
- Enforcement/Redress. Dovrebbero essere messe in atto misure di enforcement per garantire il rispetto dei FIPP.

Le FIPP sono utilizzate anche per definire altre tassonomie di privacy. Se ne cita qualcuna a titolo di esempio:

[36] https://en.wikipedia.org/wiki/Daniel_J._Solove

- linee guida Microsoft per la privacy;
- tassonomia definita da Anton et al.[13]. Questa tassonomia però, non contiene solo i cinque FIPP come obiettivi di protezione della privacy, ma include anche una serie di obiettivi di vulnerabilità di privacy che sono correlati alle minacce esistenti. Questi obiettivi di vulnerabilità includono il monitoraggio delle informazioni, l'aggregazione, la memorizzazione, il trasferimento di informazioni, la raccolta e la personalizzazione.

European Data Protection Legislation. La legislazione è una questione complessa, spesso vaga e formulata in modo ambiguo; il che la rende molto difficile da attuare. La privacy non richiede solo misure tecnologiche, ma anche misure organizzative. Inoltre, è difficile prevedere tutti i potenziali domini e contesti (e le relative normative) in cui un prodotto software verrà poi utilizzato. Tuttavia, alcune disposizioni legislative in materia di protezione dei dati possono, con un minimo sforzo, essere integrate nella progettazione del sistema.
Guarda e Zannone [2] riassumono la Direttiva Europea sulla Protezione dei Dati nei seguenti nove principi:

- Elaborazione corretta e lecita. La raccolta e il trattamento dei dati personali non devono interferire in modo irragionevole con la privacy delle persone interessate né interferire in modo irragionevole con la loro autonomia e integrità e devono essere conformi al quadro giuridico generale.
- Consenso. I dati personali devono essere raccolti e trattati solo previo esplicito consenso al loro trattamento da parte degli interessati.
- Finalità. I dati personali devono essere raccolti per finalità specifiche, lecite e legittime e non devono essere trattati per finalità non compatibili con quelle per cui sono stati raccolti.
- Minimalità. La raccolta e il trattamento dei dati personali sono limitati al minimo necessario per il raggiungimento delle finalità specifiche. Ciò include che i dati personali vengono conservati solo per il tempo necessario a raggiungere lo scopo specifico.
- Informazione minima. La divulgazione di dati personali a terzi deve essere limitata e deve avvenire solo a determinate condizioni.
- Qualità dell'informazione. I dati personali devono essere accurati, pertinenti e completi rispetto alle finalità per le quali sono raccolti e trattati.
- Controllo dell'interessato. L'interessato deve essere in grado di controllare e condizionare il trattamento dei suoi dati personali.
- Sensibilità. Nel trattamento dei dati personali è necessario applicare misure di protezione maggiormente rigorose su quei dati ritenuti particolarmente sensibili per il soggetto interessato.
- Sicurezza delle informazioni. I dati personali devono essere trattati in modo da garantire un livello di sicurezza adeguato ai rischi connessi al trattamento e alla natura dei dati stessi.

Poiché il DPD è stato creato in un momento in cui Internet era ancora agli inizi, nel 2012 è stata elaborata una proposta di riforma della legislazione attuale per rafforzare i diritti della privacy online. Questa riforma indirizza:

- il "diritto all'oblio" ovvero, l'obbligo di fornire esplicitamente il consenso necessario al trattamento dei dati;
- il "diritto di portabilità dei dati" che consente un accesso più facile ai propri dati e una maggiore trasparenza sul modo in cui questi vengono gestiti.

Anche la responsabilità di coloro che trattano i dati personali è accresciuta dall'attuazione di principi quali "Privacy by Design".
La direttiva relativa alla privacy e alle comunicazioni elettroniche è stata promulgata nel 2002 e completa la direttiva DPD in quanto si concentra sulla protezione dei dati nell'era digitale. Essa disciplina il settore delle comunicazioni elettroniche ed è stata modificata nel 2009. È principalmente nota per la richiesta di consenso da parte dell'utente nella memorizzazione dei

cookie, ed è quindi spesso indicata come la "direttiva sui cookie". Inoltre, la direttiva disciplina lo spam online imponendo un regime di “opt-in”, in base al quale le e-mail indesiderate possono essere inviate solo previo accordo del destinatario. La direttiva comprende anche la conservazione e il trattamento dei dati relativi al traffico e dei dati relativi all'ubicazione. I dati relativi al traffico dovrebbero essere cancellati o resi anonimi non appena questi non sono più necessari ai fini della trasmissione. Il trattamento di tali dati può avvenire solo quando questi vengono resi anonimi o quando l'interessato ha fornito il proprio consenso.
Sebbene la legislazione sulla protezione dei dati sia complessa e spesso ambigua, alcune norme possono essere automatizzate nei sistemi software. Negli ultimi anni è emersa una ricerca che si propone di estrarre diritti e obblighi dai documenti legali e che fornisce la tracciabilità tra le politiche sulla privacy (scritte) e le loro controparti implementate nel software.

5.8.1.1 Proprietà

In quanto concetto astratto e soggettivo, la declinazione della privacy varia a seconda delle problematiche sociali e culturali, delle discipline di studio, degli interessi degli stakeholder e del contesto applicativo. Le norme di privacy più comuni sono volte a consentire agli individui di controllare, modificare, gestire e cancellare informazioni su se stessi e decidere quando, come e in quale misura tali informazioni possono essere comunicate agli altri.
La privacy si basa prevalentemente su due modelli di tutela:

- *hard privacy* (la privacy quale libertà negativa). Si basa sul concetto di libertà (e del relativo diritto) definendo un perimetro entro cui l’individuo può agire al riparo da invasioni esterne. Questa libertà dal controllo si esprime in una sfera di libertà di scelte e di comportamenti. Nella modellazione delle minacce è necessario tener conto delle seguenti entità: il fornitore di servizi, il titolare dei dati e l'ambiente con cui interagisce.
- *soft privacy* (privacy quale libertà positiva). Contrariamente al primo, si basa sul presupposto che l'interessato abbia concesso il controllo dei propri dati personali a terzi, e debba fidarsi dell'onestà e della competenza dei responsabili del trattamento. L'obiettivo di tale modello è quindi, di fornire la sicurezza dei dati ed elaborare questi con finalità e consenso specifici, tramite politiche, controllo degli accessi e audit. Il modello prevede che l'interessato fornisca i suoi dati personali e il responsabile del trattamento di tali dati è anche responsabile della loro protezione. Di conseguenza, si applica un modello di minaccia più debole, che include diverse parti con diversi poteri.

Alla base del modello di privacy, sono presenti alcune delle classiche proprietà di sicurezza quali:

- **confidenzialità,** garantisce che le informazioni siano accessibili solo da parte di persone autorizzate;
- **integrità,** garantisce la legittimità, l'accuratezza e la completezza delle informazioni e dei metodi di elaborazione;
- **disponibilità** (o resistenza alla censura), garantisce che le informazioni siano accessibili agli utenti autorizzati;
- **non ripudio,** garantisce che non si possa negare ciò che si è fatto.

Le caratteristiche di queste proprietà si trovano nella norma ISO 17799[37]. A queste si aggiungono ulteriori proprietà quali:

- **Unlinkability**. Si riferisce alla capacità di nasconde il legame tra due o più azioni (ad esempio, nascondere i link tra due messaggi anonimi inviati dalla stessa persona), identità (ad esempio, due pagine web visitate da parte dello stesso utente o due persone collegate da una relazione di amicizia in un social network) o informazioni (ad esempio, voci presenti in due distinti database relativi alla stessa persona). La unlinkability consiste nel separare due o più elementi di interesse, detti IOI, (quali ad esempio, soggetti, messaggi, azioni, etc). Questa separazione garantisce che all'interno del

[37] https://www.iso.org/standard/39612.html

sistema (comprendente questi ed eventualmente altri elementi), l'aggressore non sia in grado di identificare in modo efficace se tali IOI sono o meno tra loro collegati.

- **Anonymity**. Si riferisce alla capacità di nascondere il legame tra un'identità e un'azione o un'informazione (ad esempio, il mittente anonimo di una e-mail, l'autore di un testo, la persona che accede ad un servizio, la persona a cui si riferisce una voce presente in un database). Tale proprietà assicura che un eventuale attaccante non possa identificare in modo efficace il soggetto. L'anonimato può essere ricondotto anche alla precedente proprietà (unlinkability).
- **Pseudonymity**. Suggerisce che è possibile costruire una reputazione su uno pseudonimo e utilizzare pseudonimi multipli per scopi diversi.
- **Plausible deniability**. Si riferisce alla capacità di ostacolare la possibilità da parte di un attaccante di dimostrare che un soggetto conosce, ha fatto o ha detto qualcosa.
- **Undetectability and unobservability**. Si riferisce alla capacità di nascondere le attività dell'utente. L'*undetectability* è vista come l'impossibilità di rilevare un elemento di interesse (IOI) da un punto di vista di un attaccante, il che significa che quest'ultimo non può distinguere quando l'elemento esiste da quando invece non esiste. L'*unobservability* è vista come l'impossibilità di osservare un elemento di interesse (IOI) da parte di tutti i soggetti non coinvolti.
- **Confidentiality**. Si riferisce alla capacità di nascondere il contenuto dei dati o di controllare la divulgazione degli stessi (ad esempio, il trasferimento di e-mail crittografate, l'applicazione del controllo di accesso a un documento classificato o a un database contenente informazioni sensibili). NIST descrive la riservatezza come la capacità di mantenere le restrizioni autorizzative all'accesso e alla divulgazione delle informazioni, compresi i mezzi per proteggere la privacy e le informazioni proprietarie. Sebbene la riservatezza sia una proprietà di sicurezza, come si evince dalla definizione, essa è importante anche per preservare le proprietà dell'anonimato e di inscindibilità.
- **Content awareness**. Si riferisce alla capacità di garantire che gli utenti abbiano la consapevolezza dei propri dati personali e di limitare l'uso delle informazioni necessarie per consentire l'esecuzione della funzione a cui si riferiscono. Quanto più sono elevate le informazioni personali identificabili divulgate dagli interessati, tanto maggiore è il rischio di violazione della privacy. L'utente deve essere consapevole delle conseguenze della condivisione delle proprie informazioni. Tali conseguenze possono riferirsi alla violazione della privacy così come a risultati indesiderati ottenuti fornendo informazioni incomplete o errate.
- **Policy and consent compliance**. Si riferisce alle politiche in atto e alle caratteristiche di conformità sul consenso in merito al trattamento dei dati personali per informare l'interessato sulla politica di privacy del sistema, o consentire allo stesso di specificare i consensi in conformità con la legislazione, prima che gli utenti stessi accedono al sistema.

Le proprietà sopra descritte possono essere considerate tutte proprietà tipiche di sicurezza.
Relativamente ai modelli di hard/soft privacy possono essere inoltre così suddivise:

- unlinkability, anonimity, pseudonymity, plausible deniability, undetectability and unobservability e confidentiality possono essere considerate come proprietà di hard privacy;
- content awareness, policy and consent compliance, possono essere considerate come proprietà di soft privacy.

5.8.1.2 Principi

La **Privacy by Design** è un concetto sviluppato alla fine degli anni 90 da Ann Cavoukian [3], commissario per l'informazione e la privacy dell'Ontario. Si tratta di un approccio ingegneristico che si concentra sull'intero processo a partire dai principi di privacy e protezione dei dati.
La tutela della privacy non dovrebbe essere garantita solo dal rispetto delle norme e dai quadri normativi, in quanto non vi è alcuna utilità nelle norme giuridiche riguardo una codifica rigorosa se il sistema stesso non ha una solida base per la sicurezza e la tutela della privacy.

La privacy by Design può essere raggiunta applicando i sette principi su cui si basa:

- **Proattivo non reattivo, preventivo non correttivo**: le minacce alla privacy dovrebbero essere anticipate e prevenute, piuttosto che corrette dopo che queste si sono verificate.
- **Privacy come impostazione predefinita ("by default")**: la privacy dovrebbe essere lo standard. I dati personali dovrebbero essere protetti automaticamente, anche senza alcuna azione esplicita da parte dell'individuo interessato.
- **Privacy incorporata nella progettazione ("by design")**: la privacy non dovrebbe essere considerata un elemento aggiuntivo, ma dovrebbe essere integrata nella progettazione e nell'architettura dei sistemi software e delle attività di business in generale.
- **Piena funzionalità - somma positiva, non somma zero**: la privacy dovrebbe coesistere con altri interessi di business. Dovrebbero tuttavia essere evitati compromessi non necessari (ad esempio privacy anziché sicurezza o privacy piuttosto che performance). Si dovrebbe sempre cercare di ottenere una somma positiva.
- **Sicurezza end-to-end - Tutela dell'intero ciclo di vita**: la privacy richiede sicurezza durante l'intero ciclo di vita dei dati personali, garantendo una gestione sicura e completa di tutti i dati.
- **Visibilità e trasparenza**: gli obiettivi di privacy dichiarati dall'organizzazione e conseguentemente adottati dai sistemi informatizzati, devono essere visibili e trasparenti agli utenti.
- **Rispetto per la privacy degli utenti**: devono essere applicate misure adeguate per responsabilizzare l'utente nel processo di trattamento dei propri dati.

Il panorama tecnologico della privacy è stato classificato da Gürses [14] secondo tre paradigmi che tuttavia, non si escludono a vicenda:

- **Privacy come controllo**. In quanto controllo, mira a fornire agli interessati un mezzo per controllare la divulgazione dei propri dati. Rientrano in questa categoria anche gli strumenti adottati dall'organizzazione per definire e applicare politiche di sicurezza dei dati e prevenire l'abuso di accessi non autorizzati. Esempi di tecnologie correlate, includono: le impostazioni relative alla privacy, il controllo dell'accesso e la verifica dei dati.
- **Privacy come riservatezza**. Questo paradigma impedisce la divulgazione delle informazioni o perlomeno ne minimizza il più possibile la divulgazione al fine di evitare di collegarle all'interessato. Le tecnologie di esempio sono: i protocolli di autenticazione anonimi e le reti di comunicazione anonime.
- **Privacy come pratica**. Si concentra maggiormente sull'aspetto sociale della privacy e mira a rendere più trasparenti i flussi di informazioni attraverso strumenti di sensibilizzazione e feedback.

5.8.1.3 Riferimenti normativi

Il 27 aprile 2016 il Parlamento Europeo ha approvato il Regolamento generale sulla protezione dei dati personali afferenti a persona fisica (General Data Protection Regulation[38]) che abroga la direttiva 95/46/CE, al fine di armonizzare le legislazioni dei singoli paesi, consolidare il diritto alla privacy dei cittadini dell'UE e garantire un'adeguata sicurezza dei dati particolari trattati dalle organizzazioni.
Il regolamento si applica a tutti gli enti pubblici e le imprese che trattano dati personali e dati personali classificati (sensibili, biometrici, sanitari e giudiziari, etc.) e/o che raccolgono grandi quantità di dati personali.
Tra gli elementi di maggiore innovazione ed interesse troviamo:

- La responsabilizzazione del Titolare nell'adozione di comportamenti proattivi tali da dimostrare l'attuazione di misure concrete individuate attraverso una valutazione dell'impatto dei trattamenti previsti (Data Protection Impact Analysis – DPIA);

[38] https://gdpr-info.eu/

- La nomina del Responsabile della protezione dei dati (Data Protection Officer – DPO o Responsabile della Protezione dei Dati – RPD) con il compito di vigilare, sui processi interni alla struttura, l'osservanza del Regolamento, fornire pareri in merito alla DPIA, indicare le misure tecniche e organizzative per attenuare i rischi per i diritti e gli interessi delle persone interessate, servire da punto di contatto con l'autorità di controllo;
- La comunicazione all'Autorità Garante di eventuali violazioni della sicurezza dei dati personali (Data Breach);
- Il principio di "privacy by design" che prevede l'integrazione delle attività volte alla protezione dei dati personali in tutte le fasi del ciclo di vita dei sistemi e delle applicazioni IT, dalla fase di progettazione, messa in esercizio, utilizzo e dismissione finale;
- Il principio di "privacy by default" che prevede il rispetto dei principi generali della protezione delle informazioni, quali la minimizzazione dei dati e la limitazione delle finalità, nelle impostazioni dei servizi e dei prodotti che trattato dati personali.

La protezione della privacy richiesta dal GDPR presuppone quindi programmi di conformità sostenuti da tutta l'organizzazione, in modo da integrare i requisiti di sicurezza dei dati in tutte le fasi di ogni processo aziendale, dalla progettazione al rilascio.
L'Articolo 25 del GDPR[39] – **Data protection "by default" "by design"** – chiede al titolare del trattamento di mettere in atto misure tecniche e organizzative adeguate:

- volte ad attuare in modo efficace i principi di protezione dei dati, quali la pseudo-anonimizzazione e la minimizzazione, e a integrare nel trattamento le necessarie garanzie al fine di soddisfare i requisiti del regolamento e tutelare i diritti degli interessati (by design);
- per garantire che siano trattati, by default, solo i dati personali necessari per ogni specifica finalità del trattamento. Tale obbligo vale per la quantità dei dati personali raccolti, la portata del trattamento, il periodo di conservazione e l'accessibilità. In particolare, dette misure garantiscono che, di base (by default), non siano resi accessibili dati personali a un numero indefinito di persone fisiche senza previo intervento dell'interessato.

Nel ciclo di vita del software sicuro, deve essere posta quindi maggiore attenzione quando si sviluppano applicativi che dovranno trattare dati personali, inserendo controlli mirati per ciascuna fase, secondo le best practices di sicurezza e secondo le indicazioni fornite dall'analisi dei rischi privacy (DPIA). Tali controlli devono essere formalmente verificati in fase di test e collaudo.
Altri elementi da considerare durante le fasi del ciclo di sviluppo del software, si evincono dall'Articolo 32 del GDPR[40] – **Sicurezza del trattamento** – nel quale viene chiesto al titolare e al responsabile del trattamento di mettere in atto le opportune misure tecniche e organizzative per garantire un livello di sicurezza adeguato al rischio, che comprendono, tra le altre cose:

- la pseudo-anonimizzazione e la cifratura dei dati personali;
- la capacità di assicurare su base permanente la riservatezza, l'integrità, la disponibilità e la resilienza dei sistemi e dei servizi di trattamento;
- la capacità di ripristinare tempestivamente la disponibilità e l'accesso dei dati personali in caso di incidente fisico o tecnico;
- una procedura per testare, verificare e valutare regolarmente l'efficacia delle misure tecniche e organizzative messe in campo al fine di garantire la sicurezza del trattamento.

5.8.2 Requisiti di sicurezza applicativi nel GDPR

Per ciò che riguarda i requisiti di AppSec nel GDPR, gli articoli 25, 32, 33, 34 e 35 contengono la maggior parte delle indicazioni di dettaglio, da seguire affinché le organizzazioni sappiano come

[39] https://gdpr-info.eu/art-25-gdpr/

[40] https://gdpr-info.eu/art-32-gdpr/

proteggere i dati trattati da parte delle applicazioni in uso, così come cosa è necessario fare in caso di violazione dei dati personali. I requisiti generali riguardano i concetti di prevenzione, valutazione e monitoraggio. Di seguito i primi cinque punti chiave relativi alle sezioni del GDPR che indirizzano la sicurezza dei dati:

- Al fine di individuare eventuali punti di debolezza nel modo in cui i dati vengono elaborati o gestiti, il GDPR richiede che le organizzazioni valutino i propri sistemi e processi in merito alla capacità di gestire i dati ed eseguire l'analisi delle difettosità al fine di determinare cosa funziona e cosa deve essere cambiato o rimosso.
- Per garantire che i dati siano sempre protetti, da parte dell'applicazione o del sistema, è necessario che vengano considerati gli aspetti di Privacy/Security sin dalla fase di progettazione e per impostazione predefinita. Tale concetto descrive l'idea che la sicurezza e la privacy devono essere entrambe prese in considerazione a partire dalle prime fasi di pianificazione, piuttosto che durante la fase realizzativa.
- Come già precedentemente indicato nel precedente paragrafo, le organizzazioni devono "garantire un livello di sicurezza adeguato al rischio", attraverso le seguenti specifiche:
    - Crittografia e pseudonimizzazione dei dati personali;
    - Capacità nel ripristinare tempestivamente la disponibilità dei dati personali in caso di incidente di sicurezza o problemi tecnici;
    - Garantire la riservatezza, l'integrità e la disponibilità dei sistemi e dei servizi per il trattamento dei dati (il principio di InfoSec).
    - Istituire un processo per effettuare regolarmente test di sicurezza e valutare l'efficacia delle pratiche e delle soluzioni di sicurezza in atto.
- Le organizzazioni devono applicare ove possibile il principio del privilegio minimo, così come attuare specifiche politiche di bonifica periodica e rimozione dei dati che non sono più necessari.
- Infine, si raccomanda alle organizzazioni, soprattutto quelle più grandi, di creare un repository centralizzato di applicazioni e dati per mantenere un miglior controllo sui dati dei propri clienti.

5.8.3 Certificazioni

In accordo con l'articolo 42 del GDPR[41], "Gli Stati membri, le autorità di controllo, il consiglio di amministrazione e la Commissione incoraggiano, in particolare a livello UE, l'istituzione di meccanismi di certificazione della protezione dei dati e di sigilli e marchi di certificazione per la protezione di quest'ultimi, al fine di dimostrare il rispetto del regolamento europeo da parte dei responsabili e degli incaricati al trattamento".
Il regolamento recita come segue: "Al fine di migliorare la trasparenza e il rispetto del presente regolamento, è opportuno incoraggiare l'istituzione di meccanismi di certificazione e di sigilli e marchi di protezione dei dati, consentendo agli interessati di valutare rapidamente il livello di protezione dei propri dati, prodotti e servizi". In tal senso, conformemente al documento di programmazione 2017[42], l'ENISA ha avviato un progetto nell'ambito dei meccanismi di certificazione per la protezione dei dati, sigilli o marchi con l'obiettivo di imprimere il panorama attuale e fornire un orientamento per ulteriori lavori sulla tematica in questione.
Questo progetto è stato realizzato parallelamente alle attività svolte dall'Agenzia nel settore della certificazione riguardo la sicurezza informatica per poter poi trarre conclusioni, esperienze e best practices utili da entrambi i settori.

41 Regolamento (UE) n. 2016/679 del Parlamento europeo e del Consiglio, del 27 aprile 2016, relativo alla tutela delle persone fisiche con riguardo al trattamento dei dati personali, nonché alla libera circolazione di tali dati, e che abroga la direttiva 95/46/CE (regolamento generale sulla protezione dei dati), L 119/1 4.5.2016

42 ENISA (2016) Documento di programmazione ENISA 2017-2019: disponibile all'indirizzo web https://www.enisa.europa.eu/publications/corporate/enisa-programming-document-2017-2019/view

Il regolamento generale sulla protezione dei dati introduce disposizioni sulla certificazione per migliorare la trasparenza delle operazioni di trattamento da parte dei responsabili e degli incaricati a quest'ultimo. Il legislatore ha inoltre previsto un ruolo di certificazione nell'assistenza ai controllori e agli incaricati del trattamento per dimostrare la conformità al regolamento.
Di seguito si descrivo i principali aspetti della pratica di certificazione esistente, applicabile al regime di certificazione della protezione dei dati istituito nell'ambito del GDPR.
La certificazione è un'attività di valutazione della conformità. Questa comporta "la presentazione di una valutazione e di un'attestazione imparziale da parte di terzi che certifichi che è stato dimostrato il rispetto dei requisiti specificati". I requisiti sono generalmente derivati da norme tecniche o legislative. Quest'ultimo è il caso della certificazione nel settore della protezione dei dati, in cui la legislazione secondaria dell'UE che tutela il diritto alla protezione dei dati personali fornisce il quadro normativo come base per i requisiti di valutazione. È anche possibile che i requisiti siano incorporati in una direttiva tecnica ispirata alle disposizioni del GDPR. Le disposizioni del GDPR devono essere ulteriormente elaborate per essere idonee alla certificazione.
Poiché la norma ISO/IEC 17067:2013 prevede che "laddove sia necessario elaborare i requisiti per eliminare l'ambiguità, le motivazioni dovrebbero essere formulate da persone competenti e messe a disposizione di tutte le parti interessate".
Secondo la terminologia proposta da ISO e IEC, un requisito di certificazione è un requisito "che il cliente deve soddisfare come condizione per stabilire o mantenere la certificazione". Il termine comprende sia i requisiti sostanziali (chiamati anche requisiti "prodotto/processi/persona") sia i requisiti procedurali. I requisiti sostanziali derivano dalla base normativa. I diritti delle persone interessate (art. 15-22 GDPR), la sicurezza dei dati (art. 32 GDPR), la protezione dei dati fin dalla progettazione (art. 25 (1) GDPR) e la protezione dei dati per default (art. 25(2) GDPR) offrono, ad esempio, una base normativa che può essere ulteriormente specificata nei requisiti sostanziali di certificazione.
Al tempo stesso, anche i requisiti procedurali fanno parte di un sistema di certificazione e devono essere soddisfatti dalla parte che richiede la certificazione[43]. Tali requisiti procedurali dovrebbero ad esempio specificare a quali condizioni il responsabile del trattamento dei dati può utilizzare il certificato acquisito, quali sono i periodi di sorveglianza, la struttura compensativa, ecc. Alcuni degli aspetti procedurali sono già chiariti dal GDPR - ad esempio la durata di validità della certificazione è di 3 anni (art. 42(7)).
Alcuni schemi di certificazione esistenti potrebbero usare il termine "criteri" per indicare i requisiti sostanziali (prodotto/processi/persona)[44]. Il GDPR sembra indicare i requisiti sostanziali come "criteri" e i requisiti procedurali come "requisiti".
La certificazione Privacy by Design non si basa né sulla legislazione né su uno standard tecnico, ma su un quadro di riferimento di sette principi fondamentali per la Privacy by Design.
Riguardo l'aspetto di garanzia di conformità con la legislazione, possono esistere tre approcci principali di seguito elencati:

- **Certificazioni indipendenti dalla legislazione**: Si tratta delle certificazioni basate sulle norme ISO/IEC o su altri documenti normativi, come il quadro normativo Privacy by Design Principles. La certificazione Privacy by Design fornita da Ryerson University e Deloitte Canada non sta' a significare la conformità alle leggi sulla privacy dell'Ontario. La norma ISO/IEC 27018, ad esempio, "stabilisce obiettivi di controllo, controlli e linee guida comunemente accettati per l'attuazione di misure volte a proteggere le informazioni personali (PII) in conformità ai principi di riservatezza contenuti nella

[43] La norma ISO/IEC 17065:2012 fornisce esempi di tali requisiti. Un esempio è il pagamento della tassa da parte dell'organizzazione richiedente all' organismo di certificazione.

[44] Un esempio è la certificazione EuroPrise.

norma ISO/IEC 29100 per l'ambiente pubblico di cloud computing"[45]. Tali certificazioni non utilizzano la legislazione come fonte normativa, bensì si basano su altri standard tecnici. Queste possono tener conto della legislazione vigente, ad esempio nel senso che i requisiti della norma non possono essere in contraddizione con gli obblighi giuridici. Tuttavia, non viene fatto alcun riferimento diretto alla legge.

- **Certificazioni che utilizzano la legislazione come fonte per i loro criteri sostanziali**: L' ePrivacyseal UE sostiene di attestare la "conformità di un prodotto all'elenco dei criteri UE ePrivacyseal, che riflette i requisiti imposti dalla legislazione UE sulla protezione dei dati"[46]. In tal senso, tali certificazioni non promettono direttamente di offrire la conformità alla legislazione sulla protezione dei dati. Essi utilizzano tuttavia la legislazione come quadro normativo. Nell' ANNEX A sono indicati i criteri di ciascuna certificazione esistente esaminata.
- **Certificazioni che garantiscono il rispetto della legislazione**: Un esempio è il caso CNIL. L'obiettivo dei sigilli di privacy del CNIL è quello di fornire al richiedente il riconoscimento da parte dell'autorità francese per la protezione dei dati che il suo prodotto, corso, procedura "soddisfi i requisiti dell'autorità francese per la protezione dei dati" ("indicatore di fiducia")[47]. Un altro esempio del genere è EuroPrise[48]. Va osservato che l'attestazione fornita da un'autorità di protezione dei dati non deve essere interpretata erroneamente come garanzia del rispetto della legislazione, fornita dall'autorità, nel suo ruolo di controllo. Al fine di evitare tali implicazioni, la CNIL chiarisce che i sigilli CNIL per la tutela della privacy non mirano ad esentare i titolari da sanzioni amministrative pecuniarie[49]. L'ICO ha proposto un approccio diverso. Nel 2015 l'ICO ha annunciato la sua intenzione di introdurre un sigillo nazionale di riservatezza. L'obiettivo era quello di fornire un timbro per le organizzazioni che "dimostrino le buone pratiche e gli elevati standard di conformità nella protezione dei dati". L'aspetto interessante dei sigilli ICO precedentemente indicati è che il sigillo non solo dimostrerebbe la conformità ai requisiti della legge britannica sulla protezione dei dati personali, ma dimostrerebbe anche che l'organizzazione certificata supera i requisiti di legge andando "oltre il necessario".

5.8.3.1 Riferimenti normativi ed esempi di certificazione

La certificazione privacy per il GDPR viene regolamentata sulla base delle indicazioni recitate dalla norma, di seguito riassunte:

- **Considerando 100**: Deve essere incoraggiata l'istituzione di meccanismi di certificazione e sigilli nonché marchi di protezione dei dati che consentano di valutare rapidamente il livello di protezione dei dati dei prodotti e servizi.
- **Articolo 42, paragrafo 1**: Incoraggia l'istituzione di meccanismi di certificazione nonché di sigilli e marchi di protezione dei dati allo scopo di dimostrare la conformità al GDPR dei trattamenti.
- **Paragrafo 5**: La certificazione è rilasciata dagli OdC o dalla DPA competente in base ai criteri approvati dalla DPA o dal Board (DPB).
- **Articolo 43, paragrafo 1**: Gli organismi di certificazione sono accreditati da (opzione):
    - la DPA competente;

[45] ISO/IEC 27018:2014 "Information technology - Security techniques - Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors", section 1 ("Scope")

[47] https://www.cnil.fr/fr/node/682

[48] Il sito web EuroPrise, ad esempio, riporta: "Certifichiamo la conformità dei prodotti IT e dei servizi basati su tecnologie dell'informazione alle normative europee sulla protezione dei dati". - https://www.european-privacy-seal.eu/EPS-en/About-EuroPriSe

[49] CNIL dichiara: "Il sigillo sulla privacy informa il pubblico che la procedura o il prodotto proposto corrisponde ai requisiti dell'Autorità per la protezione dei dati personali. A tal fine, svolge il ruolo di indicatore di fiducia. Essa non intende esonerare i suoi titolari dalle formalità amministrative." - https://www.cnil.fr/fr/questions-reponses-sur-les-labels-cnil

- dall'organismo nazionale di accreditamento (regolamento CE n. 765/2008) secondo la EN ISO/IEC 17065:2012 (relativa ai prodotti[50], processi[51] e servizi[52]) e i requisiti aggiuntivi stabiliti dalla DPA competente.

Questa viene citata anche:

- **nell'Articolo 24 paragrafo 3**: L'adesione ai codici di condotta o al meccanismo di certificazione può essere utilizzata come elemento per dimostrare il rispetto degli obblighi del titolare del trattamento.
- **nell'Articolo 25 paragrafo 3**: Il meccanismo di certificazione può essere utilizzato come elemento per dimostrare la conformità ai requisiti di cui ai paragrafi 1 [attuare in modo efficace i principi di protezione dei dati] e 2 [trattati, per impostazione predefinita, solo i dati personali necessari per ogni specifica finalità del trattamento] del presente articolo.
- **nell'Articolo 28 paragrafo 5**: L'adesione da parte del responsabile del trattamento a un codice al meccanismo di certificazione può essere utilizzata come elemento per dimostrare le garanzie sufficienti.
- **nell'Articolo 32 paragrafo 3**: L'adesione a un codice di condotta o al meccanismo di certificazione può essere utilizzata come elemento per dimostrare di garantire un livello di sicurezza adeguato al rischio.

Ma ci si riferisce anche ai codici di condotta, come ad esempio:

- **nell'Articolo 40**: Le associazioni e gli altri organismi rappresentanti le categorie di titolari del trattamento o responsabili del trattamento possono elaborare i codici di condotta.
- **nell'Articolo 41 (paragrafo 1)**: il controllo della conformità con un codice di condotta ai sensi dell'articolo 40 può essere effettuato da un organismo in possesso del livello adeguato di competenze riguardo al contenuto del codice e del necessario accreditamento a tal fine dell'autorità di controllo competente.
- **nell'Articolo 41 (paragrafo 3)**: L'autorità di controllo competente presenta al comitato il progetto di requisiti per l'accreditamento dell'organismo.
- **nell'Articolo 41 (paragrafo 4)**: un organismo adotta le opportune misure in caso di violazione del codice da parte di un titolare del trattamento o responsabile del trattamento. Esso informa l'autorità di controllo competente di tali misure e dei motivi della loro adozione.

Relativamente alla certificazione delle persone, si può dire che, la posizione più comune è che gli articoli del GDPR relativi alla certificazione non includono le persone. In Spagna è stato pubblicato nel Luglio del 2017 uno schema relativo al profilo del DPO, mentre in Italia, da tempo, si sta lavorando (in attesa di prossima pubblicazione) su una norma "Profili professionali relativi al trattamento e alla protezione dei dati personali".
La certificazione regolamentata coinvolge i seguenti attori:

- gli enti di normazione (quelli ufficiali sono regolamentati dal Regolamento Europeo 1025/2012) che emettono norme, e possono essere:
  - enti nazionali (UNI e UNINFO, BSI, DIN, etc.);
  - enti internazionali (CEN/CENELEC, ISO, IEC);
  - enti privati.

50 Risultato di un processo; servizi (per esempio, trasporto), software (per esempio, un programma per computer, il contenuto di un vocabolario), hardware (per esempio, la parte meccanica di un motore), materiali da processo continuo (per esempio, un lubrificante).

51 Insieme di attività correlate o interagenti che trasformano elementi in ingresso in elementi in uscita; esempi sono i processi di tratamento termico, di fabbricazione, di produzione di cibo, di crescita di un impianto.

52 Risultato di almeno un'attività necessariamente effettuata all'interfaccia tra il fornitore e il cliente, che è generalmente intangibile; esempi sono: riparazione di un'automobile, elaborazione della dichiarazione dei redditi, erogazione di formazione, messa a disposizione di una camera d'albergo.

- Gli organismi di accreditamento sorvegliano le attività di certificazione, accreditando (e verificando) gli organismi di certificazione. Questi si riconoscono mutualmente tra di loro (MLA), e in Europa, rispondono al Regolamento europeo 765/2008. Tali organismi operano secondo standard definiti come ISO/IEC 17011 e recepiscono anche altre norme.
- Gli organismi di certificazione, certificano le organizzazioni o i prodotti, processi e servizi, e selezionano gli auditor. Questi lavorano secondo standard definiti (ISO/IEC 17021, 17065, ecc.) e certificano organizzazioni e persone secondo "specifiche".
- Le organizzazioni e le persone che si vogliono certificare. La certificazione deve avvenire nel rispetto di norme specifiche (es. ISO 9001, UNI 11506, ecc.) dove tali specifiche possono essere pubblicate da chiunque.

E' necessario stabilire una regolamentazione degli organismi di certificazione in merito:

- alla modalità di conduzione degli audit;
- alle verifiche sulla conduzione degli audit;
- al mantenimento dei certificati (sorveglianza, verifiche in occasione di modifiche);
- alla gestione dei reclami;
- alla trasparenza e all'imparzialità;
- alle competenze del personale.

Gli Enti di accreditamento (soprattutto in Italia) pubblicano requisiti aggiuntivi a quelli delle norme ISO per gli OdC. Gli Organismi di certificazione devono pubblicare un regolamento relativo alle attività di certificazione dei prodotti, processi e servizi. Il regolamento dettaglia alcuni processi generali (per esempio la gestione dei reclami dei clienti) e dettaglia come svolgere gli audit in termini di:

- numero e tipo di verifiche di certificazione, sorveglianza e ri-certificazione;
- modalità di condivisione del rapporto;
- tipo di rilievi (classificazione delle non conformità) e loro gestione (per esempio, a fronte di non conformità gravi è necessario un audit straordinario entro poche settimane).

Dal punto di vista del funzionamento della certificazione è possibile prevedere un "sistema di certificazione alternativo" dove l'ente di accreditamento non ha accordi con altri enti, e in Europa, non risponde al Regolamento CE n. 765/2008 (ossia in competizione con l'ente nazionale), nel contempo, l'OdC non è accreditato e eroga alcuni servizi non accreditati e le specifiche vengono scritte da enti non "pubblici" o non riconosciuti. In alcuni casi, le attività non accreditate hanno come obiettivo di promuovere un sistema "normato" (in altre parole, fungono da prototipi), in altri invece, le attività non accreditate hanno come finalità la creazione di un mercato "parallelo" (pertanto non apprezzato da chi promuove il mercato "normato").
La sicurezza delle informazioni si basa sulla buona gestione, ovvero sulla giusta scelta di buoni processi di selezione, attuazione e manutenzione che portano a loro volta ad una buona scelta tecnologica. In tal senso, le normative note sono:

- Sistemi di gestione noti e diffusi:
  - ISO 9001:2015 per la qualità;
  - ISO/IEC 27001:2013 per la sicurezza delle informazioni.
- Sistemi di gestione per la privacy (standard nazionali):
  - BS 10012:2017 (UK);
  - JIS 15001:2006 (JP);
  - ISO/IEC 29151 (estensione della ISO/IEC 27001 solo per i titolari);
  - ISO/IEC 27018 (estensione della ISO/IEC 27001 solo per i fornitori di cloud pubblici);
  - ISO/IEC 27552 (di sistema di gestione, estensione della ISO/IEC 27001, disponibile da Agosto 2019).

Le certificazioni dei sistemi di gestione sono governate dalla ISO/IEC 17021, ma il GDPR cita la ISO/IEC 17065.
Dal considerando 100 del GDPR, si deduce che un trattamento è riconducibile ad un servizio e come tale si riportano a titolo puramente indicativo, alcuni esempi di certificazione:

- Centri di contatto multicanale (norme EN 15838:2010 e UNI 11200:2010)
  - Regolamento Accredia RT-22;
  - la EN 15838 include il ciclo PDCA (strategia, attività operative, riesami periodici, gestione del miglioramento);
- Erogazione di corsi di formazione;
- Vigilanza (norma UNI 10891);
- Centri di monitoraggio e ricezione allarmi (EN 50518 e UNI 11068);
- Fornitori di servizi eIDAS;
  - I requisiti consigliano l'adozione della ISO/IEC 27001;
- Fornitori di servizi SPID (richiesta la certificazione ISO/IEC 27001).

In modo analogo, si riportano a seguire alcuni esempi di certificazione di prodotto:

- Attualmente "lo" schema di certificazione della sicurezza dei prodotti informatici è costituito dalla ISO/IEC 15408 (Common Criteria);
  - richiede l'attuazione di processi molto simili (e forse più rigorosi) di quelli richiesti dai sistemi di gestione per la qualità.
- Gli schemi di certificazione di prodotto sono tanti, tra cui:
  - Regolamento Reg. CE 303/2008 (apparecchiature con gas fluorurati);
  - Direttiva PED (per i recipienti in pressione);
  - Direttiva MED (dispositivi medici).
- In molti casi è richiesto un sistema di gestione (per la qualità), anche se non necessariamente certificato.

In tale ambito esistono delle iniziative italiane, quali:

- L'11 settembre del 2018 è entrata in vigore la "Prassi di riferimento UNI" dal titolo "Linee guida per la gestione dei dati personali in ambito ICT secondo il Regolamento europeo EU 679/2016 (GDPR)" che ha le seguenti caratteristiche:
  - non è uno standard di requisiti certificabili;
  - è solo per l'ICT (comunque molto importante);
  - è comunque un punto di riferimento.
- Sulla base di bozze della norma "Profili professionali relativi al trattamento a alla protezione dei dati personali" di prossima pubblicazione, che riguarda il DPO, il Manager privacy, lo Specialista privacy e il Valutatore privacy, alcuni registri stanno promuovendo schemi di certificazione professionali.

Per il futuro, non risultano allo studio delle DPA degli schemi condivisi per i servizi. Esiste un interesse sulla ISO/IEC 27552. Alcuni Enti promuovono schemi "proprietari" non promossi da alcuna DPA (non secondo criteri approvati da alcuna DPA). Il comunicato emesso dal Garante Privacy e Accredia del 18 luglio 2017 riporta quanto segue: *in Italia non è ancora stato stabilito dal Legislatore nazionale a chi spetti il ruolo di ente di accreditamento ai fini del regolamento, né sono stati definiti i "requisiti aggiuntivi" per l'accreditamento degli organismi di certificazione (cfr. art. 43, paragrafo 1, lettera b) e i criteri di certificazione (cfr. art. 42 paragrafo 5)*. Esistono schemi promossi solo da una DPA (con valore reale solo in un Paese – Label CNIL, dal 2011; 12 label Gouvernance e 1 label relativa ai servizi) e schemi promossi da EuroPriSe (dal 2008, meno di 50 "seals" per prodotti, servizi e siti web) e ePrivacy Seal (dal 2011, circa 200 "seals" per prodotti). Gli schemi per la certificazione di prodotti, solitamente sono molto complessi da realizzare in ambito IT e molto onerosi da certificare. Forse alcuni schemi nazionali (o europeo), meno onerosi dei Common Criteria, si imporranno sul mercato europeo generale.

Successivamente l'Italia, attraverso il d.lgs. 101/2018 art. 2 septiesdecies, ha identificato nell'organismo nazionale di accreditamento (Accredia) designato in virtù del regolamento (CE) 765/2008, il soggetto che dovrà effettuare l'accreditamento pur rimanendo al Garante il potere di accreditare direttamente in alcune materie specifiche (es. Biometria, genetica, ecc.) o lì dove l'Ente di accreditamento risulti inadempiente.

5.8.4 Best practices per il trattamento dei dati personali

- **Ridurre al minimo i dati personali utilizzati -** *Ridurre l'impatto dei rischi limitando la gestione dei dati personali a ciò che è strettamente necessario per raggiungere lo scopo definito.*
  - Comprovare che i dati personali siano sufficienti, pertinenti e non eccessivi rispetto all'intento; diversamente, non raccoglierli.
  - Comprovare che i dati personali non rivelino (direttamente o indirettamente) l'origine razziale o etnica, le opinioni politiche, filosofiche o religiose, l'appartenenza sindacale, le informazioni sulla salute o le informazioni sulla vita sessuale di un individuo, tranne che per circostanze eccezionali.
  - Comprovare che i dati personali non si riferiscano a reati, sentenze penali o misure di sicurezza.
  - Evitare la raccolta di dati personali aggiuntivi.
  - Limitare la trasmissione di documenti elettronici contenenti dati personali allo stretto necessario.
  - Eliminare in caso di necessità se non più utili, i dati personali o le richieste di un soggetto dal sistema in esercizio o dai backup.
- **Gestire i periodi di conservazione dei dati personali -** *Ridurre l'impatto dei rischi assicurando che i dati personali non vengano mantenuti per più di quanto necessario.*
  - Definire periodi di conservazione dei dati personali limitati nel tempo e appropriati allo scopo dell'elaborazione.
  - Verificare che l'elaborazione possa rilevare la scadenza del periodo di conservazione.
  - Verificare che l'elaborazione consenta la cancellazione dei dati personali a scadenza del periodo di conservazione e che il metodo scelto per l'eliminazione sia appropriato ai rischi legati alla libertà civile e alla privacy dei soggetti interessati.
  - Eliminare immediatamente i dati personali quando il periodo di conservazione scade.
- **Informare i soggetti e ottenere il consenso -** *Consentire ai soggetti interessati di effettuare una scelta libera, specifica e informata.*
  - Determinare se l'elaborazione si basa su una base giuridica diversa dal consenso.
  - Determinare i mezzi pratici da attuare per ottenere il consenso degli interessati.
  - Assicurare che il consenso sia ottenuto prima che inizi l'elaborazione.
  - Assicurare che il consenso sia ottenuto liberamente.
  - Assicurare che il consenso sia ottenuto in modo notificato e trasparente in termini di finalità del trattamento.
  - Assicurare che il consenso sia ottenuto per uno scopo specifico.
- **Partizionare i dati personali -** *Ridurre la possibilità che i dati personali possano essere correlati e che possa verificarsi una violazione di questi***.**
  - Identificare i dati personali utili solo al singolo processo (l'identificazione deve essere svolta per ciascun processo dell'organizzazione).
  - Separare in modo logico i dati utili di ciascun processo.
  - Comprovare regolarmente che i dati personali siano partizionati in modo efficace e che i destinatari e le interconnessioni non siano correlabili ai dati stessi.
- **Cifrare i dati personali -** *Rendere incomprensibili i dati personali a chiunque non sia autorizzato all' accesso.*
  - Determinare tutto ciò che deve essere crittografato (inclusi dischi rigidi, file, dati provenienti da un database o canali di comunicazione) in base alla forma in cui sono memorizzati i dati personali, i rischi individuati e le prestazioni richieste.

  - Scegliere il tipo di crittografia (simmetrica o asimmetrica) in base al contesto e ai rischi individuati.
  - Adottare soluzioni di crittografia basate su algoritmi pubblici notoriamente robusti.
  - Definire opportune misure per garantire la disponibilità, l'integrità e la riservatezza delle informazioni necessarie al recupero delle informazioni perse (incluse le password di amministratore e dati di ripristino).
- **Anonimizzare i dati personali -** *Eliminare le caratteristiche che identificano i dati personali.*
  - Determinare ciò che deve essere anonimo in base al contesto, alla forma in cui vengono memorizzati i dati personali (compresi i campi del database o estratti dai testi) e rischi individuati.
  - Anonimizzare permanentemente i dati che richiedono tale criterio di protezione in base alla loro forma (inclusi database e record testuali) e i rischi individuati.
  - Se questi dati non possono essere anonimizzati in modo permanente, scegliere strumenti che rispondano innanzitutto alle esigenze funzionali.

5.8.5 Linee guida per lo sviluppo di applicazioni sicure conformi al GDPR

L' introduzione della legge europea sulla privacy dei dati online è destinata ad avere un grande impatto sul modo in cui le organizzazioni dovranno trattare e gestire i dati personali dei propri utenti. Per le organizzazioni che trattano regolarmente i dati dei propri utenti o i dati personali elaborati dai servizi ai cittadini europei, sorgono questioni relative alle implicazioni tecniche riguardo le loro applicazioni e operazioni web online.
La direttiva principale di questa regolamentazione conferisce alle persone fisiche il potere di controllare i propri dati. Ciò significa che gli enti che richiedono informazioni personali online devono informare l'utente su cosa esattamente accadrà ai loro dati, dal momento in cui questi vengono forniti.
Gli aspetti più importanti del regolamento sono i seguenti:

- **Un accesso più facile ai dati personali**: gli individui dovranno avere maggiori informazioni sul trattamento dei loro dati e tali informazioni dovranno essere disponibili in modo chiaro e comprensibile.
- **Il diritto alla portabilità dei dati**: dovrà essere più facile per i soggetti trasferire i propri dati personali tra i vari fornitori di servizi.
- **Un "diritto all'oblio" più chiaro**: se non si desidera più che i propri dati vengano trattati e se non vi sono motivi legittimi per la loro conservazione, questi dovranno essere cancellati.
- **Il diritto di sapere quando i propri dati vengono violati**: ad esempio, le aziende e le organizzazioni devono informare quanto prima possibile l'autorità nazionale di controllo in merito ad eventuali gravi violazioni dei dati personali, affinché gli utenti interessati possano prendere le opportune misure.

Pertanto, in che modo si realizza un'applicazione conforme alla direttiva di cui sopra, tale da fornire un controllo completo dei dati personali degli utenti? Quelle che seguono possono essere considerate le best practices, sulla base delle linee guida sulla privacy indicate dall'OWASP Top Ten:

- **Determinare se l'applicazione ha realmente bisogno di tutti i dati personali richiesti**: l'implementazione ottimale della privacy consente di salvare il minor numero possibile di dati personali, come data di nascita, nome, paese di residenza, ecc. Ciò non è sempre possibile; alcune entità potrebbero aver bisogno di maggiori informazioni. Tuttavia, gli sviluppatori e il management devono definire esattamente quali dati sono assolutamente necessari e quali no.
- **Crittografare tutti i dati personali e informarne gli utenti**: Se un'applicazione ha bisogno di salvare informazioni personali, tali dati devono essere cifrati con algoritmi di crittografia robusti e appropriati, includendo anche l'hashing. Inoltre, dovrebbe essere esplicitamente indicato agli utenti che tutti i loro dati personali, compresi i numeri di telefono, il paese di residenza e l'indirizzo, verranno criptati e verrà calcolato il valore di hash per prevenire qualsiasi forma di estrazione e

manipolazione dei dati in oggetto che porterebbe ad una potenziale esposizione in caso di violazione di quest'ultimi.

- **Considerare l'utilizzo dell'OAUTH per la portabilità dei dati**: I protocolli per l'accesso singolo come OAUTH[53] consentono agli utenti di creare account semplicemente fornendo un altro account, ma al contempo assicurano la non memorizzazione di dati personali diversi dall'ID di autenticazione dell'altro servizio. In altre parole, trattasi di uno standard di autenticazione online open source che consente ad un utente di dare l'accesso alle sue informazioni archiviate nei sistemi di un determinato service provider ad un altro servizio senza tuttavia condividere le sue credenziali.
- **Garantire comunicazioni sicure tramite HTTPS**: Molti siti web non utilizzano l'HTTPS perché non lo ritengono necessario. Ad esempio, se l'applicazione non richiede alcun tipo di autenticazione, l'HTTPS potrebbe sembrare non necessario. Ma è facile ignorare alcuni aspetti. Ad esempio, alcune applicazioni raccolgono dati personali tramite i loro form "contattaci". Se tali informazioni vengono inviate in chiaro, queste potrebbero essere visualizzate su Internet. Inoltre, è necessario assicurarsi che il certificato SSL sia stato distribuito correttamente e non sia esposto a vulnerabilità a cui i protocolli SSL sono soggetti.
- **Informare gli utenti e crittografare i dati personali provenienti dai form 'Contattaci'**: Le applicazioni non raccolgono informazioni solo tramite autenticazione o sottoscrizione, bensì anche attraverso moduli di contatto. La maggior parte di queste informazioni sono personali, tra cui indirizzo e-mail, numero di telefono e paese di residenza. Gli utenti devono essere informati su come e per quanto tempo questi dati verranno conservati. E' altamente raccomandato l'uso di crittografia forte per la memorizzazione di tali informazioni.
- **Assicurarsi che le sessioni e i cookie abbiano una scadenza e che vengano poi distrutti dopo il logout**: Gli utenti devono avere una visibilità adeguata sull'uso dei cookie da parte dell'applicazione. Essi devono essere informati del fatto che l'applicazione sta utilizzando i cookie, l'applicazione deve fornire la possibilità agli utenti di accettare o negare i cookie, e i cookie devono essere adeguatamente distrutti dopo un periodo predefinito di inattività o il logout.
- **Non tenere traccia delle attività degli utenti ai fini della business intelligence**: Molte applicazioni e-commerce su web tracciano le attività degli utenti per determinare le loro preferenze attraverso le ricerche effettuate o i prodotti da loro acquistati. Spesso aziende come Amazon e Netflix utilizzano questo tipo di informazioni per sponsorizzare le loro piattaforme. Poiché i gusti e le scelte personali degli utenti vengono monitorati e conservati ai fini commerciali, gli utenti devono poter accettare o rifiutare tale opzione. Se gli utenti decidono di accettare il tracciamento, questi devono essere informati su come e per quanto tempo i loro dati verranno salvati nel sistema. Ovviamente, tutto ciò che riguarda le informazioni personali deve essere cifrato.
- **Informare gli utenti riguardo l'attività di logging svolta, che riguarda la memorizzazione della posizione o degli indirizzi IP**: Molte applicazioni utilizzano indirizzi IP o posizioni di geo localizzazione come parametro per controllare l'autenticazione e le autorizzazioni, e registrano queste informazioni nel caso in cui qualcuno tenti di aggirare i controlli di autenticazione. Gli utenti devono essere informati di ciò, così come della durata della storicizzazione di tali informazioni nei log del sistema. Non memorizzare mai nei registri di log informazioni significativamente sensibili come le password.
- **Conservare i dati di log in un luogo sicuro, preferibilmente in formato cifrato**: Conservare in un luogo sicuro tutti i dati di log che contengono le informazioni degli utenti e informare quest'ultimi su come tali informazioni vengono trattate: come vengono memorizzati e per quanto tempo vengono conservati. I log stessi devono essere cifrati.
- **L'uso di domande relative alla sicurezza non devono esporre i dati personali degli utenti a possibili violazioni**: In molte applicazioni, le domande di sicurezza vengono usate per confermare l'identità di un utente. Queste domande non dovrebbero includere dati personali come il nome da ragazza della mamma o il colore preferito dell'utente. Se possibile, sostituire tali domande con l'autenticazione a

[53] Protocollo aperto, sviluppato da Blaine Cook e Chris Messina a partire dal novembre 2006: https://oauth.net/

due fattori. Se ciò non è possibile, permettere agli utenti di creare e personalizzare le proprie domande mettendoli in guardia contro la scelta di domande che contengono dati personali. Tutte le informazioni fornite devono essere cifrate.

- **Creare termini e condizioni chiari e assicurarsi che gli utenti li leggano**: Non nascondere i termini e le condizioni. In base alle nuove leggi dell'UE sulla privacy, i termini e le condizioni devono essere presenti sulla pagina iniziale di qualsiasi applicazione ed essere altamente visibili durante la navigazione dell'applicazione da parte dell'utente. È necessario implementare un meccanismo nell'applicazione tale per cui gli utenti debbano accettare i termini e le condizioni prima di poter accedere all'applicazione stessa, e in particolar modo, quando tali termini sono stati oggetto di modifica. I termini e le condizioni devono inoltre essere redatti in un linguaggio facilmente comprensibile dall'utente finale.
- **Informare gli utenti di qualsiasi condivisione dei loro dati con terze parti**: Se l'organizzazione ha la necessità di condivide i dati personali degli utenti con terze parti, siano essi componenti software esterni, affiliati o organizzazioni governative, ciò deve essere esplicitato nei termini e condizioni.
- **Creare dei criteri chiari per le violazioni dei dati personali**: Uno degli aspetti più importanti della legge dell'UE è il diritto degli utenti di essere informati in caso di violazione dei propri dati. Le organizzazioni devono implementare politiche chiare che stabiliscano ruoli e procedure da seguire in modo tale che, ad esempio, questi vengano tempestivamente informati a fronte di ogni violazione.
- **Rimuovere i dati personali degli utenti quando questi annullano la propria sottoscrizione al servizio**: Molte applicazioni non forniscono con chiarezza informazioni riguardo il trattamento dei dati personali relativi ad un utente che ha annullato la propria sottoscrizione al servizio o che ha cancellato il proprio account. Con il diritto di essere dimenticati, le organizzazioni devono rispettare il diritto degli utenti di cancellare il proprio account e tutti i relativi dati associati. Deve essere chiaro e visibile a tutti gli utenti che possono abbandonare il servizio e che sistematicamente tutti i loro dati verranno cancellati. Le organizzazioni che trattano gli account cancellati semplicemente come inattivi, potrebbero essere in contrasto con la normativa.
- **Patch delle vulnerabilità del Web**: Come menzionato nell' elenco OWASP Top 10[54], uno dei principali rischi per la privacy dei dati riguarda le vulnerabilità delle applicazioni web: "La vulnerabilità è un problema chiave in qualsiasi sistema che protegga o operi su dati sensibili dell'utente. La progettazione e l'implementazione inadeguata di una applicazione, la rilevazione di un problema o l'applicazione tempestiva di una correzione (patch) possono comportare una violazione della privacy. Assicurarsi che l'organizzazione disponga di un programma capace di valutare i rischi informatici ed eseguire efficacemente test di penetrazione e patch.

5.8.6 Tecniche di modellazione e individuazione delle minacce

5.8.6.1 LINDDUN

La privacy è diventata una questione chiave nell'e-society. È della massima importanza che la privacy sia integrata quanto prima nel ciclo di vita del software di sviluppo. LINDDUN[55] è una metodologia di analisi delle minacce alla privacy e supporta gli analisti nell'individuare i requisiti di riservatezza.
LINDDUN è un nome mnemonico sviluppato da Mina Deng [15] per il suo dottorato di ricerca alla Katholieke Universiteit di Leuven, Belgio. Questa è una metodologia speculare alla modellazione delle minacce STRIDE (STRIDE-per-element) e tratta le violazioni delle seguenti proprietà sulla privacy:

- Non collegabilità (Unlinkability);
- Non identificabilità (Anonymity/Pseudonymity);

54 https://www.owasp.org/index.php/OWASP_Top_10_Privacy_Risks_Project

55 https://www.linddun.org/

- Ripudio (Plausible Deniability);
- Non rilevabilità (Undetectability);
- Non divulgazione di informazioni (Confidentiality);
- Consapevolezza sul contenuto (Content Awareness);
- Aderenza alla politica sul consenso (Policy and consent compliance).

LINDDUN viene presentato come un approccio completo alla modellazione delle minacce con un metodo di individuazione di processi, minacce e requisiti. Può essere ragionevole utilizzare LINDDDUN come framework per l'identificazione delle minacce sulla privacy, in sostituzione o in aggiunta alla STRIDE.
In primo luogo, viene creato un diagramma di flusso dei dati (DFD), una rappresentazione grafica strutturata del sistema che utilizza quattro tipi principali di elementi: entità, archivi dati, flussi di dati e processi. Ciascun tipo di elemento DFD viene associato a una serie di categorie di minacce alla privacy (sono state identificate sette categorie di alto livello di minacce alla privacy: **L**-Linkability, **I**-Identifiability, **N**-Non Repudiation, **D**-Detectability, **D**-Disclosure of information, **U**-Content Unawareness e **N**-Policy and consent Non-compliance). Per identificare le minacce che insistono sul sistema analizzato, per ciascun elemento è necessario esaminare le minacce corrispondenti alle categorie di cui sopra.
La tabella seguente, mostra la correlazione tra le minacce di privacy previste da LINDDUN e le tipologie di elementi DFD sopra descritte:

Elemento DFD	L	I	N	D	D	U	N
Archivio dati	X	X	X	X	X		X
Flusso dati	X	X	X	X	X		X
Processo	X	X	X	X	X		X
Entità	X	X				X	

Tabella 24 - Minacce LINDDUN per elemento DFD

La metodologia LINDDUN supporta l'analista fornendo una serie di alberi di minaccia che descrivono i percorsi d'attacco più comuni per ogni possibile combinazione tra le tipologie di minaccia e gli elementi DFD. Basandosi su questi alberi, l'analista potrà documentare le minacce identificate, utilizzando scenari di casi di abuso per descrivere in dettaglio i possibili attacchi. Le minacce vengono quindi considerate prioritarie in base al loro rischio. Tuttavia non fornisce esplicitamente un supporto per l'analisi del rischio. Le minacce che derivano da tale processo, possono quindi essere tradotte in requisiti di riservatezza. Infine, LINDDUN fornisce un elenco di soluzioni per la privacy al fine di mitigare le minacce individuate.

La tabella seguente, riporta gli obiettivi di privacy (proprietà della privacy) basati sule varie tipologie di minaccia previste in LINDDUN, dove (**E**-Entità, **DF**-Flusso Dati, **DS**-DataStore, **P**-Processo).

Minacce LINDDUN	Obiettivo elementare a tutela della privacy
Linkability of (E,E)	Unlinkability of (E,E)
Linkability of (DF,DF)	Unlinkability of (DF,DF)
Linkability of (DS,DS)	Unlinkability of (DS,DS)
Linkability of (P,P)	Unlinkability of (P,P)
Identifiability of (E,E)	Anonymity/pseudonymity of (E,E)

Identifiability of (E,DF)	Anonymity/pseudonymity of (E,DF)
Identifiability of (E,DS)	Anonymity/pseudonymity of (E,DS)
Identifiability of (E,P)	Anonymity/pseudonymity of (E,P)
Non-repudiation of (E,DF)	Plausibledeniability of (E,DF)
Non-repudiation of (E,DS)	Plausibledeniability of (E,DS)
Non-repudiation of (E,P)	Plausibledeniability of (E,P)
Detectability of DF	Undetectability of DF
Detectability of DS	Undetectability of DS
Detectability of P	Undetectability of P
Information Disclosure of DF	Confidentiality of DF
Information Disclosure of DS	Confidentiality of DS
Information Disclosure of P	Confidentiality of P
Content Unawareness of E	Content awareness of E
Policy and consent Noncompliance of the system	Policy and consent compliance of the system

Tabella 25 - obiettivi di privacy basati sule varie tipologie di minaccia previste in LINDDDUN

5.8.6.1.1 Tecniche di mitigazione

Nella metodologia LINDDUN, le proprietà e la corrispettive minacce alla privacy vengono classificate come hard e soft privacy. La tabella a seguire evidenzia tale classificazione:

Proprietà di privacy	Minaccia alla privacy
Hard privacy	
Unlinkability	Linkability
Anonymity & Pseudonymity	Indentifiability
Plausible deniability	Non repudiation
Undetectability & unobservability	Detectability
Confidentiality	Disclosure of information
Soft privacy	
Content awareness	Content Unawareness
Policy and consent compliance	Policy and consent non-compliance

Tabella 26 - LINDDUN Hard & Soft privacy

LINDDUN fornisce per ogni tipo di potenziale minaccia identificata una o più classificazioni delle tecniche di mitigazione da mettere in campo attraverso una mappatura tra obiettivi e tecniche di miglioramento della privacy (PETs):

	Tecniche di mitigazione	U	A	P	D	C	W	O
Anonymity system	• Mix-networks (1981) • DC-networks (1985) • ISDN-mixes • Onion Routing (1996) • Crowds (1998) • Single proxy (90s) (Penet pseudonymous remailer (1993-1996), Anonymizer, SafeWeb) • Anonymous Remailer (Cipherpunk Type 0, Type 1, Mixmaster Type 2 (1994), Mixminion Type 3 (2003)) • Low-latency communication (Freedom Network (1999-2001), Java Anon Proxy (JAP) (2000), Tor (2004))	X	X			X		
	• DC-net & MIX-net + dummy traffic • ISDN-mixes	X	X		X	X		
	• Broadcast systems + dummy traffic	X	X		X			
Privacy preserving authentication	• Private authentication • Anonymous credentials (single show, multi show)	X	X					
	• Deniable authentication	X	X	X				
	• Off-the-record messaging	X	X	X		X		
Privacy preserving cryptographic protocols	Multi-party computation (Secure function evaluation)	X				X		
	Anonymous buyer-seller watermarking protocol	X	X			X		
Information retrieval	Private information retrieval + dummy traffic	X	X		X			
	Oblivious transfer	X	X			X		
	Privacy preserving data mining	X	X			X		
	• Searchable encryption • Private search		X			X		
Data anonymization	• K-anonymity model • l-Diversity	X	X					
Information hiding	Steganography	X	X		X			
	Covert communication	X	X		X			
	Spread spectrum	X	X		X			
Pseudonymity systems	Privacy enhancing identity management system	X	X					
	User-controlled identity management system	X	X					

	Privacy preserving biometrics	**X**	**X**			**X**		
Encryption techniques	Symmetric key & public key encryption					**X**		
	Deniable encryption			**X**		**X**		
	Homomorphic encryption					**X**		
	Verifiable encryption					**X**		
Access control techniques	Context-based access control					**X**		
	Privacy-aware access control					**X**		
Policy and feedback tools	Policy communication (P3P)							**X**
	Policy enforcement (XACML, EPAL)							**X**
	Feedback tools for user privacy awareness						**X**	
	Data removal tools (spyware removal, browser cleaning tools, activity traces eraser, harddisk data eraser)						**X**	

Tabella 27 - Mappatura tra obiettivi e tecniche di miglioramento della privacy

* **U**–Unlinkability, **A**–Anonymity/Pseudonymity, **P**–Plausible deniability, **D**–Undetectability/unobservability, **C**–Confidentiality, **W**–Content Awareness, **O**–Policy and consent compliance of the system

5.8.6.2 PRoPAN

Beckers et al. [4] ha creato un approccio in quattro fasi per l'identificazione semiautomatica delle minacce alla privacy. Il metodo ProPAn (Problem-based Privacy Analysis) contribuisce nella produzione dei requisiti di protezione della privacy ed è un approccio basato su problemi per l'identificazione semiautomatica delle minacce alla privacy durante l'analisi dei requisiti dei sistemi software. L'obiettivo di questa metodologia è quello di assistere gli ingegneri del software nella requisitazione al fine di ottenere le seguenti informazioni:

- conoscenza del settore rilevante per la privacy;
- dati personali trattati;
- requisiti di riservatezza.

La metodologia consiste in quattro fasi:

- Disegno del diagramma di contesto e dei diagrammi dei problemi,
- Aggiunta dei requisiti di privacy al modello,
- Generazione di grafici delle minacce alla privacy,
- Analisi dei grafici delle minacce alla privacy.

Riferimento bibliografico: Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis. A Problem-based Approach for Computer Aided Privacy Threat Identification. In Privacy Technologies and Policy, volume 8319 of LNCS, pages 1–16. Springer, 2014.

5.8.6.3 PriS

PriS [5] è un metodo di ingegnerizzazione dei requisiti di sicurezza, che integra i requisiti di riservatezza già nelle fasi iniziali del processo di sviluppo del sistema. PriS considera i requisiti relativi alla privacy come obiettivi organizzativi che devono essere soddisfatti e adotta l'uso di modelli di processi di privacy come un modo per:

1. descrivere l'effetto dei requisiti relativi alla privacy sui processi aziendali;
2. facilitare l'identificazione dell'architettura di sistema che meglio supporta i processi aziendali in relazione agli aspetti di privacy.

In questo modo, PriS fornisce un approccio olistico che va dagli obiettivi di alto livello ai sistemi informatici "rispettosi della privacy". Il metodo si articola in quattro fasi:

- individuare gli obiettivi di tutela della privacy,
- analizzare l' impatto degli obiettivi di tutela della privacy sui processi organizzativi,
- modellare i processi interessati utilizzando modelli di tutela della privacy
- identificare le tecniche che meglio supportano o implementano i processi summenzionati.

Riferimento bibliografico: Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Addressing privacy requirements in system design: the PriS method. Requirements Engineering, 13(3):241–255, 2008.

5.8.6.4 FPFSD

Spiekermann e Cranor [6] hanno sviluppato un framework denominato (FPFSD). Questo è un framework per la progettazione di sistemi rispettosi della privacy (framework for privacy-friendly system design) in cui si attua una distinzione tra due diversi approcci alla privacy:

- Privacy-by-policy, introduce l'approccio di notifica e consenso basato sui principi di Fair Information Practice Principles (FIPP) e implica che l'utente sia informato su quali informazioni vengono utilizzate e perché. Inoltre, l'utente può decidere di non fornire dati.
- Privacy-by-architecture, incoraggia l'archiviazione dei dati presso il cliente invece di far archiviare le informazioni riservate dalle aziende stesse.

Riferimento bibliografico: Sarah Spiekermann and Lorrie F. Cranor. Engineering privacy. IEEE Transactions on Software Engineering, 35(1):67–82, 2009.

5.8.6.5 MPRA

Gürses [7] propone una tecnica multilaterale per l'analisi dei requisiti in materia di tutela della privacy (multilateral privacy requirements analysis technique). Si articola in tre fasi principali: analisi degli stakeholder, analisi funzionale e analisi della privacy. In primo luogo vengono determinati gli attori e le parti interessate. Per ciascuno di questi, vengono determinati gli obiettivi funzionali che vengono poi collegati alle ipotesi di dominio. Viene inoltre creato un modello informativo. Nella fase finale, le problematiche sulla privacy sono determinate in base a ciascun stakeholder, in relazione al modello informativo e agli obiettivi funzionali individuati nella fase precedente. Tali problematiche possono anche tradursi in minacce alla privacy e documentate come casi di abuso. Infine, le problematiche (o minacce) vengono trasformate in obiettivi di tutela privacy come un'approssimazione giustificabile.
Riferimento bibliografico: Fahriye Seda Gürses. Multilateral privacy requirements analysis in online social network services. PhD thesis, Department of Computer Science, KU Leuven, 2010.

5.8.6.6 Privacy in the Cloud

Mouratidis et al. [8] pone in particolare l'attenzione sulle applicazioni cloud proponendo un framework che supporta l'elicitazione dei requisiti di sicurezza e privacy. Il framework è composto da un linguaggio (basato su Secure Tropos[56]) e da un processo (basato su PriS) a supporto dell'analisi della sicurezza e della privacy. Il processo si articola in tre fasi. Il primo passo (facoltativo) è la catalogazione delle minacce alla sicurezza e alla privacy, che mira a creare un punto di riferimento basato sull'esperienza passata da utilizzare poi successivamente. In secondo luogo, l'attività di analisi della sicurezza e della privacy consiste in due sotto-attività: la definizione del contesto organizzativo, in cui vengono identificati gli obiettivi organizzativi, gli attori e le dipendenze, i piani e le risorse e gli obiettivi di sicurezza e privacy; e la definizione delle possibili problematiche in materia di sicurezza e privacy, che consiste nell'identificare i requisiti, le misure e i meccanismi di sicurezza e privacy. Il terzo e ultimo passo è la selezione del provider dei servizi

[56] http://www.troposproject.eu/node/301

cloud in base al grado di soddisfazione dei meccanismi di sicurezza e privacy ottenuti dai potenziali fornitori cloud. Sebbene si tratti di un framework che fornisce procedure dettagliate per analizzare i requisiti di sicurezza e privacy, non è disponibile alcuna conoscenza reale della sicurezza e della privacy.
Riferimento bibliografico: Haralambos Mouratidis, Shareeful Islam, Christos Kalloniatis, and Stefanos Gritzalis. A framework to support selection of cloud providers based on security and privacy requirements. Journal of Systems and Software, pages 2276–2293, 2013.

5.8.6.7 Adaptive privacy

Omoronyia et al. [9] propone un framework di riferimento per supportare la divulgazione selettiva delle informazioni personali da parte di applicazioni software in un contesto in continuo cambiamento. Il framework si concentra sui requisiti di sensibilizzazione alla privacy (PAR - privacy awareness requirements) e descrive:

1. come identificare gli attributi da monitorare per rilevare le minacce alla privacy,
2. come scoprire le minacce alla privacy prima della divulgazione delle informazioni personali,
3. come determinare la gravità, così come i benefici relativi a una minaccia scoperta.

Il quadro normativo richiede tuttavia che i requisiti di riservatezza di tutti gli agenti siano già definiti e non fornisce indicazioni su come ottenerli.
Riferimento bibliografico: Inah Omoronyia, Luca Cavallaro, Mazeiar Salehie, Liliana Pasquale, and Bashar Nuseibeh. Engineering adaptive privacy: on the role of privacy awareness requirements. In Proceedings of the 2013 International Conference on Software Engineering, ICSE ’13, pages 632–641. IEEE Press, 2013.

5.8.6.8 STRAP

STRAP [10] è un framework di analisi della privacy che è stato sviluppato sulla base dei risultati dell'analisi di sei framework esistenti (modelli di rischio [16], Patrick e Kenny [17], framework i* [18], Langheinrich [19], Bellotti e Sellen [20], e valutazione euristica [21]). Si tratta di un metodo a cinque fasi, che consiste in una analisi orientata agli obiettivi, analisi della vulnerabilità, perfezionamento e progettazione degli obiettivi, valutazione del progetto e iterazione. La fase di analisi della vulnerabilità è la fase principale in cui vengono combinati i framework esistenti. L'analisi si basa su una serie di domande analitiche per determinare la cattura e l'uso delle informazioni. In secondo luogo, viene utilizzata l'euristica per identificare potenziali problemi basati su difetti e requisiti comuni. Queste euristiche possono essere classificate secondo le FIPP degli Stati Uniti: notice/awareness, choice/consent, integrity/security, enforcement/redress. Sebbene queste categorie, basate sulla legislazione e sugli orientamenti in materia di protezione dei dati, siano molto importanti dal punto di vista della tutela della privacy, non tengono conto delle proprietà fondamentali quali l'anonimato, la non collegabilità e la non rilevabilità.
Riferimento bibliografico: Carlos Jensen. Designing for privacy in interactive systems. PhD thesis, Georgia Institute of Technology, 2005.

5.8.6.9 Microsoft privacy guidelines

Le linee guida sulla privacy fornite da Microsoft descrivono alcuni concetti basilari di privacy [11], come diversi tipi di consenso o concetti di minimizzazione dei dati. Inoltre, per gli scenari selezionati vengono presentate alcune linee guida riguardanti i seguenti principi: avviso, scelta, trasferimento successivo, accesso, sicurezza e integrità dei dati. Tuttavia, contiene solo un elenco piatto degli orientamenti richiesti e raccomandati e non intende descrivere un approccio più strutturato. Queste linee guida possono essere utilizzate come ispirazione per determinare le possibili minacce, ad esempio per ampliare il catalogo degli alberi delle minacce.

Riferimento bibliografico: Privacy guidelines for developing software products and services[57], version 3.1. Technical report, Microsoft Coorporation, Sept 2008.

5.8.6.10 PRET

Miyazaki et al. [12] ha definito una tecnica di elicitazione dei requisiti di riservatezza informatizzata, denominata (PRET - privacy requirements elicitation technique). Questa tecnica restituisce i requisiti necessari affinché il sistema sia conforme alla normativa, sulla base di un questionario compilato dall'ingegnere di sistema.
Riferimento bibliografico: Seiya Miyazaki, Nancy Mead, and Justin Zhan. Computer-aided privacy requirements elicitation technique. Asia-Pacific Conference on Services Computing (APSCC'08), pages 367–372, 2008.

[57] https://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Privacy%20in%20Software%20Development.ppsx&usg=AOvVaw0McrzvmQjgjTgDzttoZEkC