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*.