1. Workshop Ergebnisse

### **Allgemeine Herausforderungen & Überlegungen**

1. Chat-GBT ist bei „grüner Wiese“/generischem Input nicht sonderlich hilfreich - es fehlen Kontext und klare Sicherheitsinformationen.
   * **Erfüllt:** Du lädst Chat-History, übergibst sie an den Prompt, inkl. User-Level, JSON-Input → Kontext vorhanden
2. DFDs sind oft unzureichend, da sie keine implementierten Sicherheitsmaßnahmen enthalten.
3. Die Qualität der Analyse steht und fällt mit dem DFD: Magelnder Input, Mangelnder Output.
4. Ein reiner One-Shot-Ansatz ist unrealistisch -> Threat Modeling ist ein iterativer, dialogbasierter Prozess.
   * **Erfüllt:** Chat-Verlauf wird gespeichert, History genutzt, multiple Sessions möglich → iterative Nutzung technisch möglich
5. Nutzer sollten ermutigt werden, möglichst viele Informationen einzugeben - ChatGPT muss zur Nachfragelogik befähigt sein.
6. KI kann nicht vollständig autonom bewerten, es braucht Nutzerfeedback & manuelles Nachbessern.

### **Eingabeformate & Datenquellen**

1. Bildformate (z. B. DFD als Grafik) sind schlecht verarbeitbar, textuelle oder strukturierte Formate (z. B. JSON, YAML) sind besser.
   * **Erfüllt:** Du akzeptierst JSON-Dateien, keine Bilder → Fokus auf strukturiertes Format
2. Vorteil textbasierter Formate: günstigere und direktere Verwendung mit der API.
   * **Erfüllt:** JSON-Upload, Ausgabe ebenfalls als JSON → textbasiert
3. Möglichkeit, auch den **Output** wieder als JSON zu bekommen, um ihn weiterzuverarbeiten oder zu kürzen.
   * **Erfüllt:** Prompt fordert JSON-Output, Antwort enthält JSON am Ende

### **Zielgruppen & Benutzerprofile**

1. Für Nicht-Security-Experten ist die Tiefe der Ausgabe oft zu viel, es braucht didaktische Aufbereitung.
   * **Erfüllt:** User-Level wird genutzt
2. Entwickler interessieren sich mehr für Frameworks, Bibliotheken und technische Umsetzungstipps.
   * **Erfüllt:** User-Level wird genutzt
3. Kleine Firmen/Freelancer profitieren besonders, da oft Know-how fehlt oder später nachgeholt wird.
   * **Erfüllt:** User-Level wird genutzt. Dadurch können auch Freelancer ohne viel Knowledge das Tool nutzten
4. Security-Profis würden das System eher als Zweitgutachter oder Challenge-Partner nutzen.
   * **Erfüllt:** User-Level wird genutzt. Tool soll algemein beim ThreatModeling assistieren
5. System sollte persönliche Expertenprofile berücksichtigen: je nach Vorwissen differenzierte Erklärungen anbieten.
   * **Erfüllt:** User-Level wird genutzt

### **Interaktion & Nutzerführung**

1. Nummerierung von DFD-Kanten erlaubt gezielte Rückmeldung zu einzelnen Datenflüssen
2. Das System sollte interaktiv Feedback einholen: **„Hier fehlen mir Infos, bitte nachspezifizieren“**.
3. Threat Modeling sollte eingebettet sein in agiles oder iteratives Vorgehen (Spiralmodell).
   * **technisch möglich:** Sessions + History + JSON-Update ermöglichen Iteration
4. Das Tool sollte auch Systeme/ Ansätze von vor Wochen analysieren können, was hat sich seit letzter Iteration geändert?
   * **Erfüllt:** Chat-Verlauf wird gespeichert, History genutzt, multiple Sessions möglich → wieder am Chat weiterarbeiten
5. System soll helfen, das DFD gemeinsam zu verbessern, nicht nur auswerten, sondern mitentwickeln.
   * **Teilweise:** JSON wird eingelesen und Antwort kann DFD updaten → gemeinsames Verbessern angedacht
6. Das Interface sollte den Nutzer stärker anleiten: „Was fehlt noch?“, „Was wurde schon implementiert?“.
7. Automatisches Parsen der Antwort nötig: nicht rohe ChatGPT-Ausgabe zeigen, sondern strukturieren.

### **Prompt-Design & Automation**

1. Prompt-Vorlagen könnten helfen, müssen aber sorgfältig genutzt werden (Gefahr: Anchoring-Effekt).
   * **Erfüllt:** Du nutzt ein umfassendes Prompt-Template, achtest aber nicht explizit auf Anchoring-Gefahr
2. Vorlagen können helfen, Themen zu erschließen, in denen der Nutzer kein Experte ist (z. B. Krypto).
3. Gefahr der Überautomatisierung: Nutzer verfallen in „Autopilot“, vertrauen dem System zu sehr.
4. Hintergrund-Prompts sind sinnvoll, sollten aber die Eigenverantwortung des Nutzers nicht verdrängen.
5. Vorschlag: Prompts und Ergebnisse intern validieren oder mit bekannten Mustern/Datenbanken kombinieren.

### **Transparenz & Vertrauenswürdigkeit**

1. System soll seine **Unsicherheiten** klar kommunizieren („Ich vermute, weil ich es nicht genau sehe...“).
2. Vertrauen steigt mit menschlicher, transparenter Kommunikation: wo liegen Unsicherheiten?
3. Persönliche Erklärungstiefe: Experten brauchen keine Basisinfos, Personen ohne Fachwissen hingegen schon.
4. Wichtiger Hinweis: LLMs bleiben fehleranfällig -> Nutzer müssen darauf hingewiesen werden.

### **Zielbild für das Tool**

1. Ein interaktives System, das Nutzer durch die Modellierung begleitet und bei der Verbesserung unterstützt.
2. Fokus auf Bedrohungen erkennen, nicht unbedingt gleich Lösungen liefern.
3. Ideal für Architekten als Checkliste/Review-Tool zur Überprüfung, ob relevante Threats bedacht wurden.
   * **Erfüllt:** Fokus auf Begleitung & Assistenz ist da

Nicht Ziel: Komplettlösung für Security-Design, sondern **Assistenzsystem im Prozess**.

**1.Schritt**

Zielgruppe: Anfänger, Junior und Jinor Leute

Interface soll: DFD verbessern/vertieften, auswertung, transparenz verbessern (welche infos fehlen noch) (Output soll Analyse, detailierte Infos über DFD (4-5 Kanten))

Was brauch man als Input und was als Output und was muss in der Mitte passieren?????????

**Interface:**

Input: (JSON)-file und text input

Chat-Fenster (mit iterativen Versuch)

Output: text + JSON-file (strukturiert und hinweise auf Grenzen von Chat-GBT)

Logic-Desicion Tree aufbauen (DFD Kanten hinzufügen)

Beginn: Nutzer sollen sich in Knowledge einranken.

JSON-Datei einlesen → analysieren → Feedback generieren → neue JSON-Version → wieder analysieren usw.

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Eingabe | Prozess | Ausgabe |
| 1. Start | Erste JSON-DFD | Grundanalyse + STRIDE + Kantenerkennung (DFD- Prüfung der Struktur: Anzahl Knoten/Kanten, Typen, Verbindungen) | Analyse + neue JSON v1 |
| 2. Dialog 1 | JSON v1 + User-Input (Bestätigter oder angepasster DFD | Prüfung offener Fragen, Rückfragen stellen (Interaktiver Dialog: gezielte Fragen stellen, KI-Grenzen kommunizieren, Nutzeranweisungen geben, Markierung fehlerhafter oder verdächtiger Knoten/Kanten) | JSON v2 + präzisere Analyse |
| 3. Dialog 2 | JSON v2 + User-Input | Weitere Detaillierung, Validierung | JSON v3 ... |
| ... | ... | ... | ... |
| Finale | JSON final + STRIDE (Alle gesammelten Daten & Nutzerfeedback) | Endauswertung | Endbericht + JSON final |
|  |  |  |  |

* **Diskussion Workshop ergebniss als stichpunkte 1.-…**
* **diese in Methodik aufgreifen: benutzen ergebnis von 1. workshop in unserem interface. Bsp. Punkt 1 ahebn wir so und so ungestzt….**