Workshop Ergebnisse

Allgemeine Herausforderungen & Überlegungen

- Chat-GBT ist bei „grüner Wiese“/generischem Input nicht sonderlich hilfreich - es fehlen Kontext und klare Sicherheitsinformationen.
- DFDs sind oft unzureichend, da sie keine implementierten Sicherheitsmaßnahmen enthalten.
- Die Qualität der Analyse steht und fällt mit dem DFD: Magelnder Input, Mangelnder Output.
- Ein reiner One-Shot-Ansatz ist unrealistisch -> Threat Modeling ist ein iterativer, dialogbasierter Prozess.
- Nutzer sollten ermutigt werden, möglichst viele Informationen einzugeben - ChatGPT muss zur Nachfragelogik befähigt sein.
- KI kann nicht vollständig autonom bewerten, es braucht Nutzerfeedback & manuelles Nachbessern.

Eingabeformate & Datenquellen

- Bildformate (z. B. DFD als Grafik) sind schlecht verarbeitbar, textuelle oder strukturierte Formate (z. B. JSON, YAML) sind besser.
- Vorteil textbasierter Formate: günstigere und direktere Verwendung mit der API.
- Möglichkeit, auch den **Output** wieder als JSON zu bekommen, um ihn weiterzuverarbeiten oder zu kürzen.
- Nummerierung von DFD-Kanten erlaubt gezielte Rückmeldung zu einzelnen Datenflüssen.

Zielgruppen & Benutzerprofile

- Für Nicht-Security-Experten ist die Tiefe der Ausgabe oft zu viel, es braucht didaktische Aufbereitung.
- Entwickler interessieren sich mehr für Frameworks, Bibliotheken und technische Umsetzungstipps.
- Kleine Firmen/Freelancer profitieren besonders, da oft Know-how fehlt oder später nachgeholt wird.
- Security-Profis würden das System eher als Zweitgutachter oder Challenge-Partner nutzen.
- System sollte persönliche Expertenprofile berücksichtigen: je nach Vorwissen differenzierte Erklärungen anbieten.

Interaktion & Nutzerführung

- Das System sollte interaktiv Feedback einholen: **„Hier fehlen mir Infos, bitte nachspezifizieren“**.
- Threat Modeling sollte eingebettet sein in agiles oder iteratives Vorgehen (Spiralmodell).
- Das Tool sollte auch Systeme/ Ansätze von vor Wochen analysieren können, was hat sich seit letzter Iteration geändert?

- System soll helfen, das DFD gemeinsam zu verbessern, nicht nur auswerten, sondern mitentwickeln.
- Das Interface sollte den Nutzer stärker anleiten: „Was fehlt noch?“, „Was wurde schon implementiert?“.
- Automatisches Parsen der Antwort nötig: nicht rohe ChatGPT-Ausgabe zeigen, sondern strukturieren.

Prompt-Design & Automation

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

Transparenz & Vertrauenswürdigkeit

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

Zielbild für das Tool

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

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