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**.

**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.

Nach jedem Output: Feedback von Nutzer erhalten.

Descision/logic Tree:

Json

Json file interpretiert werden (wie viele Kanten haben wir?)

1. -> Konten mit Nutzer überprüft werden
2. Fragen hin und her (was fehlt noch? Wo hat die KI Grenzen?)
3. STRIDE analyse methode
4. Ganz zum schluss das alles als JSON output

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input | System/KI-Prozess | Output |
| 1. Nutzer-Input | JSON-File (aktuelles DFD) + Text-Input (z.B. Beschreibung, Fragen) | JSON parsen und analysieren: Anzahl Knoten & Kanten erkennen | Interne Datenstruktur für DFD (Knoten, Kanten) |
| 2. DFD-Analyse | Interne DFD-Datenstruktur | Logik-/Entscheidungsbaum aufbauen: Knoten & Kanten prüfen, Inkonsistenzen erkennen, fehlende Verbindungen identifizieren | Hinweise auf fehlende/vervollständigte Knoten/Kanten; Analysebericht (Text) |
| 3. Nutzer-Dialog | Nutzer-Feedback (via Chat-Fenster) | Interaktiver Dialog: gezielte Fragen stellen, KI-Grenzen kommunizieren, Nutzeranweisungen geben | Verbessertes Verständnis der DFD; Anleitungen & Vorschläge (Text) |
| 4. STRIDE-Analyse | Aktualisierte DFD-Datenstruktur | STRIDE-Methode anwenden: Bedrohungen identifizieren, klassifizieren und priorisieren | STRIDE-Analysebericht (Text) |
| 5. Zusammenfassung | Alle gesammelten Daten & Nutzerfeedback | Abschluss: Validierung der Analyse, JSON mit finalem DFD und Anmerkungen generieren | Finales JSON-File mit aktualisiertem DFD + Analysehinweise; Textzusammenfassung |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Input | System/KI-Prozesse (Zwischenschritte) | Output | Nutzerinteraktion/Anleitung |
| 1. Initialisierung | - JSON-File (aktueller DFD mit 4–5 Knoten)- Textinput zur Beschreibung des Systems (optional) | - Parsen der JSON-Datei- Visualisierung des DFD- Prüfung der Struktur: Anzahl Knoten/Kanten, Typen, Verbindungen | - Visualisierte Übersicht des DFD- Textuelles Feedback zur Struktur | - Kurze Einführung: „Ihr aktuelles Modell hat X Knoten und Y Kanten.“- Aufforderung: „Prüfen Sie die Darstellung. Fehlt etwas?“ |
| 2. LLM-gestützte Strukturprüfung | - Bestätigter oder angepasster DFD | - Entscheidungspfad: a) Sind alle Knoten verbunden? b) Stimmen Datenflüsse? c) Hinweise auf typische Fehler (z. B. unverbundene Elemente, fehlende Datenflüsse)- Identifikation möglicher Lücken oder Unstimmigkeiten | - Textuelle Analyse der DFD-Qualität- Markierung fehlerhafter oder verdächtiger Knoten/Kanten | - Fragen wie: „Was macht dieser Knoten? Gehört er zur Datenverarbeitung?“- Schrittweise Anregungen zur Verbesserung |
| 3. Dialogische Vervollständigung | - Nutzerantworten im Chat | - LLM stellt gezielte Fragen zur Domäne & Funktionalität- Aufbau eines iterativen Dialogs zur Vervollständigung des DFD (z. B. „Fehlt ein Authentifizierungsprozess?“)- Prüfung auf Konsistenz | - Verfeinerter DFD- Neue Vorschläge als Text & JSON-Patches | - Feedbackaufforderung: „War diese Ergänzung sinnvoll?“- Möglichkeit zur manuellen Korrektur |
| 4. STRIDE-Bedrohungsanalyse | - Finalisierter oder bestätigter DFD | - Anwendung der STRIDE-Methode: a) Klassifikation jeder Komponente b) Bedrohungen nach Kategorie identifizieren c) Priorisierung (z. B. CVSS-Schätzung)- Erkennung potenzieller Angriffsvektoren | - Liste erkannter Bedrohungen pro DFD-Komponente- Priorisierte Maßnahmenempfehlungen | - Erklärung der Kategorien („S = Spoofing, T = Tampering…“)- Interaktive Beispiele zur Verdeutlichung |
| 5. Evaluation & Feedback | - Vom System generierte Bedrohungsanalyse | - Einbindung eines einfachen Feedbackmoduls: „Welche Vorschläge waren hilfreich?“„Was war unverständlich?“- Bewertung der KI-Ausgabe (z. B. 1–5 Sterne) | - Dokumentierte Nutzerbewertungen- Verbesserte Hinweise für zukünftige Analysen | - Nach jeder Analyse: Feedbackabfrage- Button: „Waren diese Hinweise nachvollziehbar?“ |
| 6. Finalisierung | - Validiertes DFD + Analyse | - Generierung eines strukturierten JSON-Outputs (kompletter DFD + Bedrohungen)- Ausgabe zusammengefasster Analyseergebnisse (Textform)- Hinweise zu Modellgrenzen und Unsicherheiten | - Finales JSON-File (exportierbar)- Textzusammenfassung inkl. Unsicherheiten- Optional: PDF-Bericht | - Abschlussanzeige: „Modell ist abgeschlossen. Möchten Sie exportieren oder Feedback geben?“ |