Plan de Tesis - IA para Detección y Validación de Vulnerabilidades

IA para extracción, priorización y validación de vulnerabilidades: un sistema RAG entrenado con write-ups públicos y validación de explotabilidad en laboratorio controlado

Resumen:

Desarrollaré un sistema híbrido (Retrieval-Augmented Generation + reglas heurísticas) que, usando write-ups públicos (HackTheBox retired, VulHub, DVWA, blogs técnicos) y fuentes CVE/NVD, identifique y priorice brechas en activos autorizados. El sistema realizará reconocimiento pasivo y safe scans en entornos de producción únicamente con consentimiento, y cuando el riesgo lo requiera, replicará el objetivo en un laboratorio aislado para validar explotabilidad mediante PoC controladas (snapshots, instrumentación, logs firmados). El resultado será un informe técnico y uno ético/operacional, además de un plan de remediación priorizado y playbooks de verificación.

Preguntas de investigación:

1. ¿Puede un sistema RAG entrenado con write-ups públicos y datos CVE inferir con alta precisión la explotabilidad potencial sin ejecutar exploits en producción? 2. ¿Qué combinación de evidencia (CVSS, similitud con write-ups, resultados de escáner pasivo) optimiza la detección de vulnerabilidades explotables? 3. ¿Qué metodología de validación en laboratorio produce la menor tasa de falsos positivos/negativos al confirmar explotabilidad? 4. ¿Qué formato de informe y playbook facilita la remediación efectiva y verificable por equipos de operaciones?

Objetivos:

- Objetivo general: Construir y evaluar un sistema IA que detecte, priorice y valide vulnerabilidades en activos autorizados, realizando validación de explotabilidad solo en laboratorio aislado, y genere planes de remediación verificables. - Objetivos específicos: 1. Recolectar, normalizar y anotar un dataset de write-ups públicos con etiquetas MITRE/CVE/exploitability_label. 2. Implementar un pipeline RAG (Embeddings + VectorDB + LLM) para recuperación y síntesis de casos similares. 3. Diseñar e implementar un módulo de decisión (PE-score) que combine CVSS, similitud, scanner flags y exposición. 4. Crear un orquestador de laboratorio (docker/vagrant/ansible) para validación PoC segura y reproducible. 5. Evaluar el sistema mediante métricas automáticas y evaluación humana.

Alcance y limitaciones:

Incluye: write-ups públicos, CVE/NVD, transcripciones públicas; análisis pasivo y safe scanning en producción con consentimiento; validación de explotación en laboratorio aislado y controlado. Excluye: ataque/explotación en sistemas sin autorización; publicación de PoC ejecutable que facilite ataques externos.

Metodología (resumida):

Fase 0: Aprobación ética y formato de consentimiento (JWT/PDF firmado). Fase 1: Scraping y normalización de write-ups públicos → JSONL; anonimización y extracción de pasos. Fase 2: Indexación (embeddings) y RAG (FAISS/Chroma + LLM); templates de prompt para extracción de MITRE/JSON outputs. Fase 3: Módulo /analyze-target (FastAPI) con recolector pasivo y safe-scan opcional; cálculo de PE-score y decisión. Fase 4: Orquestador LAB (docker/vagrant + Ansible) para deploy → snapshot → instrumentación → PoC controlada → revert. Fase 5: Generación de informes (técnico + ético) y plan de remediación con playbooks de verificación. Fase 6: Evaluación: métricas (precision/recall/F1, ROC-AUC, recall@k) y evaluación humana con pentesters.

Pipeline de decisión y scoring (simplificado):

pe_score = w1*cvss_norm + w2*writeup_similarity + w3*scanner_flag + w4*config_exposure
Pesos sugeridos: w1=0.35, w2=0.30, w3=0.25, w4=0.10 Umbrales: low=0.3; medium=0.6; high=0.85

Laboratorio controlado (normas):

- Aislamiento total (red privada). - Snapshots antes/después y reversión automática. - Instrumentación: auditd/sysmon, pcap, logs firmados. - PoC sólo en réplicas; no publicar PoC ejecutables.

Entregables y cronograma (24 semanas):

Sem 1-2: Revisión, alcance y aprobación ética. Sem 3-6: Scraping y dataset inicial (>=500 write-ups). Sem 7-9: Anotación y RAG prototipo. Sem 10-12: API /analyze-target (validación de consentimientos). Sem 13-15: Orquestador LAB + playbooks Ansible. Sem 16-18: Validación PoC en lab (>=20 tests). Sem 19-20: Evaluación cuantitativa y panel humano. Sem 21-22: Redacción capítulos Experimentos y Resultados. Sem 23-24: Revisión final, anexos y defensa.

Métricas de éxito:

- Precisión ≥ 0.80 en clasificación de exploitability confirmed/unconfirmed (N≥200). - ROC-AUC ≥ 0.85 para PE-score. - Recall@5 ≥ 0.9 en retrieval. - ≥ 70% hallazgos con plan de remediación verificable.

Riesgos y mitigaciones:

- Poca data: usar RAG y augmentación. - Riesgos legales: consentimiento JWT y revisión legal. - Abuso de PoC: PoC almacenado solo en repo interno con control de acceso. - Falsos positivos: calibración y revisión humana.

Entregables prácticos propuestos (appendix):

1. Esquema JSONL (ejemplo). 2. Plantilla de consentimiento (PDF/JWT example). 3. Función PE-score (Python snippet). 4. Esqueleto FastAPI (/analyze-target). 5. docker-compose + Ansible skeleton para lab. 6. Notebook RAG demo. 7. Plantillas Jinja2 para informes técnico y ético.

Checklist para presentación:

- Título y resumen aprobados. - Aprobación ética/legal. - Repo inicial y dataset prueba. - Prototipo RAG funcional. - API básica y LAB reproducible. - Panel de validación humana definido. - Cronograma con hitos y entregables.