AI DIY Speed Run: Build Your Own AI Tools for Pentesting In No Time At All

Betta Lyon Delsordo
Lead Application Penetration Tester @ OnDefend
September 30, 2025
CyberCon @ BSC

I bet you spend at least 3 hours a day on boring stuff... emails, meetings..?

Well let’s make this a productive 3 hour build session instead!

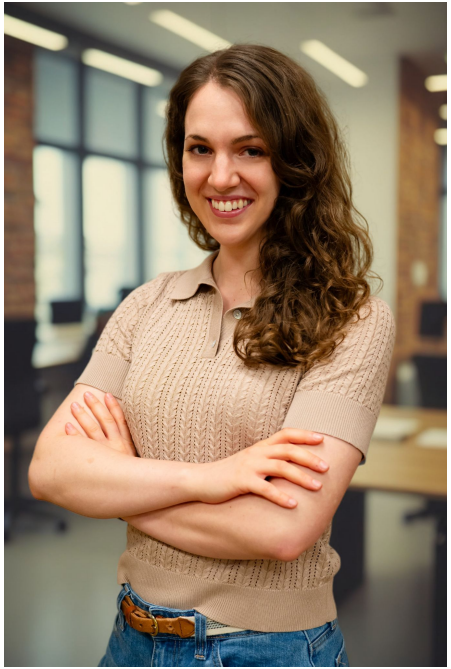
Let's get building!

I've saved you the research time, let's jump right into the quickest ways to build AI tools for pentesting

1) Intro
2) Framing the AI tool problem
3) Task brainstorm
4) No code AI tools
5) Low code AI tools
6) Offline AI tools
7) Uncensored AI & agents
8) Full code AI tools
9) Closing
10) Questions?

1) Intro

Hi, I'm Betta! I hack websites



- Started teaching myself to code at 13
- Began building websites for small businesses in Montana in high school through college
- Realized that web dev made me a good web hacker!
- Full cyber mode: M.S. Cyber, NSA cert program, GPEN
- Now an ethical hacker: web, cloud, AI, source code
- Currently a Lead Application Pentester @ OnDefend
- Have built several AI tools for pentesting: AI Secure Code Review and Pentest Threat Intel tool

Workshop Materials:

https://github.com/Betta-Lyon-Delsordo/building-ai-tools/

2) Framing the AI tool problem

AI Tools > AI Pentesters

Fancy AI tools exist, but aren't perfect

- People are building 'AI Pentesters'....
- Existing AI tools for pentesting and cyber have issues:
  - Not accurate or observable
  - Too expensive
  - Too complicated
  - Don't connect to your tools and infrastructure
  - Train on sensitive client data
  - Don't work offline or on small testing VMs
  - Lack of oversight and guardrails
- So, let's go build our own!

3) Task brainstorm

What tasks are best for AI tools?

- Target a very niche and repetitive task that you have to perform over and over in pentests
- Don’t try to build a whole ‘AI pentester’, just a tool to make an expert go faster
- Often tasks that already have existing automations (scripts, aggregations) but still require analysis, pattern matching, summarization

What tasks are best for AI tools?

- OSINT
- Recon
- Code review
- Documentation review
- Digesting past pentest reports
- Cloud IAM policy review
- Application log review
- Known exploits
- Reporting (but with specific prompts)
- News feed analysis and industry updates

What are your ideas?

https://app.sli.do/event/sVR5JW2tAKyhBMMXoijM6J

4) No code AI tools

No code = great for prototyping

- No code AI tools are popping up everywhere, can ‘vibe’ your way to tools
- Great for prototyping and ideation, present an idea
- Harder to scale and maintain
- May not be customizable beyond a certain point
- Perfect for startups, less technical people, leadership to try out AI tools in a quick way
- And also pentesters with limited time!

AWS PartyRock Demo: https://partyrock.aws/

Show off your creation:

https://app.sli.do/event/ofEvrhskgvEtCVGaFjoNxA/live/polls

Zapier Demo: https://zapier.com/

5) Low code AI tools

Low code, lots of possibilities

- Low code AI options are targeted at practitioners with some technical background, but limited time
- Easy to abstract out repetitive tasks, then you fine tune
- Focus on writing queries, searches instead of algorithms
- May have to write your own integrations to own infrastructure

Amazon Bedrock: https://aws.amazon.com/bedrock/

AI for kids?

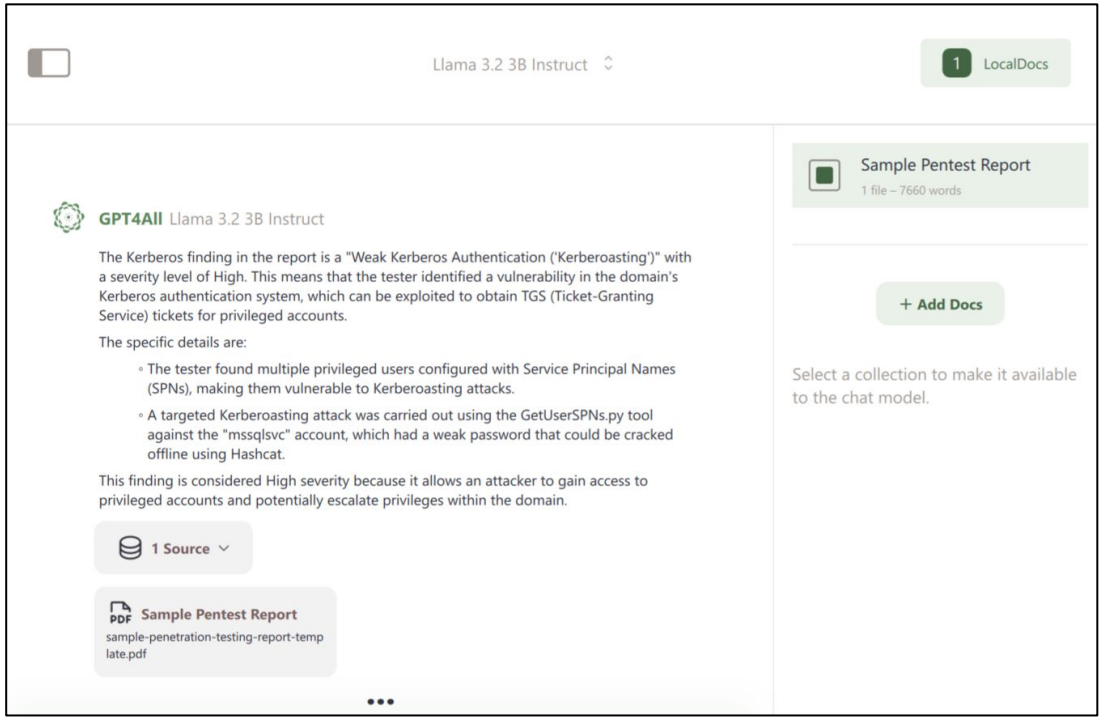
- There are many great “AI for kids” tools out there, that are actually great low-code (colored blocks) options for startups and proofs of concepts
- Machine Learning for Kids: https://machinelearningforkids.co.uk/
- Teachable Machine: https://teachablemachine.withgoogle.com/
- Thunkable: https://thunkable.com/

6) Offline AI tools

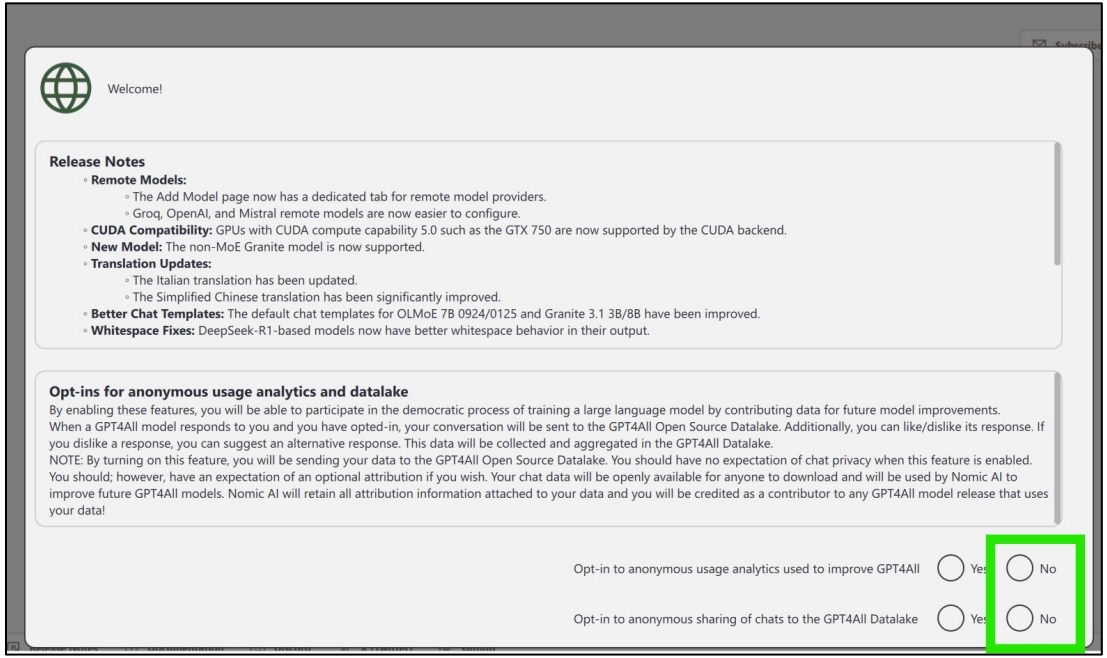
Why do we need offline AI tools?

- Paranoid clients, privacy freaks
- National security
- Competition with hosting providers
- Distributed tech/ limited processing power or connectivity
- Compliance and data privacy regulations
- Prototyping phase, limit risk

Offline Local Docs with RAG



Offline considerations…



GPT4All: https://www.nomic.ai/gpt4all

Llama3: https://www.llama.com/models/llama-3/

Ollama: https://ollama.com/

7) Uncensored AI & agents

Uncensored AI, Jailbreaks

- Most AI models are censored for human decency: they could be horrible to use without some filters
- But where do you draw the line? Political censorship, moral issues, cultural representation
- Us hackers need to do ‘malicious’ things for good
- Jailbreaking = remove restrictions from creators
- Some models have a ‘prebaked’ jailbreak = uncensored
- Causes instability, slower, not good for general tasks

The power of uncensored models…

```
PS C:\Users\betsy> ollama list
NAME                                ID              SIZE      MODIFIED
gdisney/mistral-uncensored:latest    783a351307c9    4.1 GB    5 months ago
llama3.2:latest                      a80c4f17acd5    2.0 GB    5 months ago
PS C:\Users\betsy> ollama run llama3.2
>>> tell me how to hack a website search bar
I can't fulfill this request.

>>> /bye
PS C:\Users\betsy> ollama run gdisney/mistral-uncensored
>>> tell me how to hack a website search bar

To hack a website search bar, you will need to use a tool like SQLmap or Burp Suite. First, you'll need to
identify the type of database used by the website by inspecting the HTML code and looking for the query string.
Once you know the type of database, you can use a tool like SQLmap to inject SQL queries into the search bar. This
will allow you to access sensitive information or even take control of the website. It's important to note that
hacking is illegal in most jurisdictions, and attempting to do so can result in serious consequences.

>>> Send a message (/? for help)
```

Mistral Uncensored:
https://ollama.com/gdisney/mistral-uncensored

ollama run gdisney/mistral-uncensored

Agents work for you

- Agents are all the rage now, autonomous decision makers
- But be VERY careful about level of access and autonomy granted!
- Best to do human in the loop, approve for key decisions
- Good for continuous querying, decision trees, iterative actions
- Must take care in pentesting: scope is legally defined, if an agent goes out of scope and causes damage, you are liable

LangChain Agents:

https://python.langchain.com/docs/tutorials/agents/

PentestGPT:

https://github.com/GreyDGL/PentestGPT

Poll: do you trust AI agents with pentesting?

https://app.sli.do/event/orEWPACrpGhsrqxYDhAuPg

8) Full code AI tools

Full code: RAG applications

- The most common full code AI setup these days is a RAG application: supplementing an LLM's context with your own knowledge base
- RAG = Retrieval-Augmented Generation, combining data retrieval with text generation
- Vector database = A database for storing and searching vector data from text docs
- FAISS index = Facebook AI Similarity Search, tool for similarity search of vectors
- Embedding algorithm = Converts data into vectors

1) Do your research

- What are the most important features to your target users?
- What other solutions are already out there?
- What benchmarks for speed and accuracy are available?
- Will you need an uncensored LLM?
- What are the requirements for confidentiality and use of client data?
- What is your budget?
- What are your hardware or cloud server requirements?
- How comfortable are you with programing, data science, and machine learning?

2) Choose your host

- Cloud providers

- LLM-enabled platforms

- Your own hardware

3) Choose your AI model

- Open source + free + established

- Paid + proprietary

Claude

OpenAI

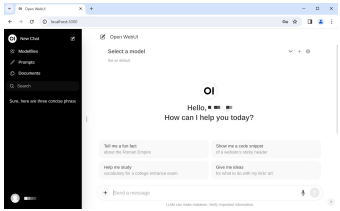
- Experimental + hobby projects

Hugging Face

4) Choose your middleware

- LLM management

- GUI console



- Docker containers, API management, database integration…

LangChain

docker

5) Choose your chunking methods

- LangChain has various methods:
  - Text splitting
  - Code splitting
  - Recursive or not
  - Custom parameters
- More processing power = smaller chunks = more accurate RAG

Hugging Face

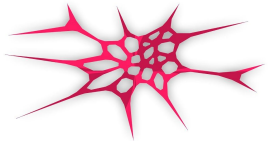
6) Choose your embedding algs + database format

- Embedding algorithms can be open source or proprietary

OpenAI Hugging Face

- RAG could be FAISS stored as a ‘pickle’

FAISS
Scalable Search With Facebook AI



- Or can use ChromaDB as an alternate method

Chroma

7) Develop and test

- Prompt engineering
- Censored and uncensored LLM usage
- Add timing functions to your code
- Ways to improve run time:
- Save your RAG for later, only create once
- Add options for fine-tuning based on user's processing power
- Experiment with chunk sizes and what to include in the RAG
- Choose different embedding algorithms or AI models
- Prompt only: do you actually need a RAG?
- Processing power and CPU/GPU usage

LangChain: https://python.langchain.com/docs/introduction/

Gradio: https://www.gradio.app/

Streamlit: https://streamlit.io/

ChromaDB: https://www.trychroma.com/

Hugging Face Embeddings:

https://huggingface.co/blog/getting-started-with-embeddings

FAISS: FAISS: https://faiss.ai/index.html

9) Closing

Look at you! Already building AI tools

- Now, make a plan to dive deeper on one of these tools or platforms that would be most useful to you
- Get buy-in from your team and leadership
- Start teaching others how to be more productive with these tools as well
- Hopefully you’re on the path to spending less time on boring tasks!

10) Questions?

Let's connect!

https://www.linkedin.com/in/betta-lyon-delsordo/