

Extending Open Source Tools using AI

or

Generating a DefectDojo Parser using only AI



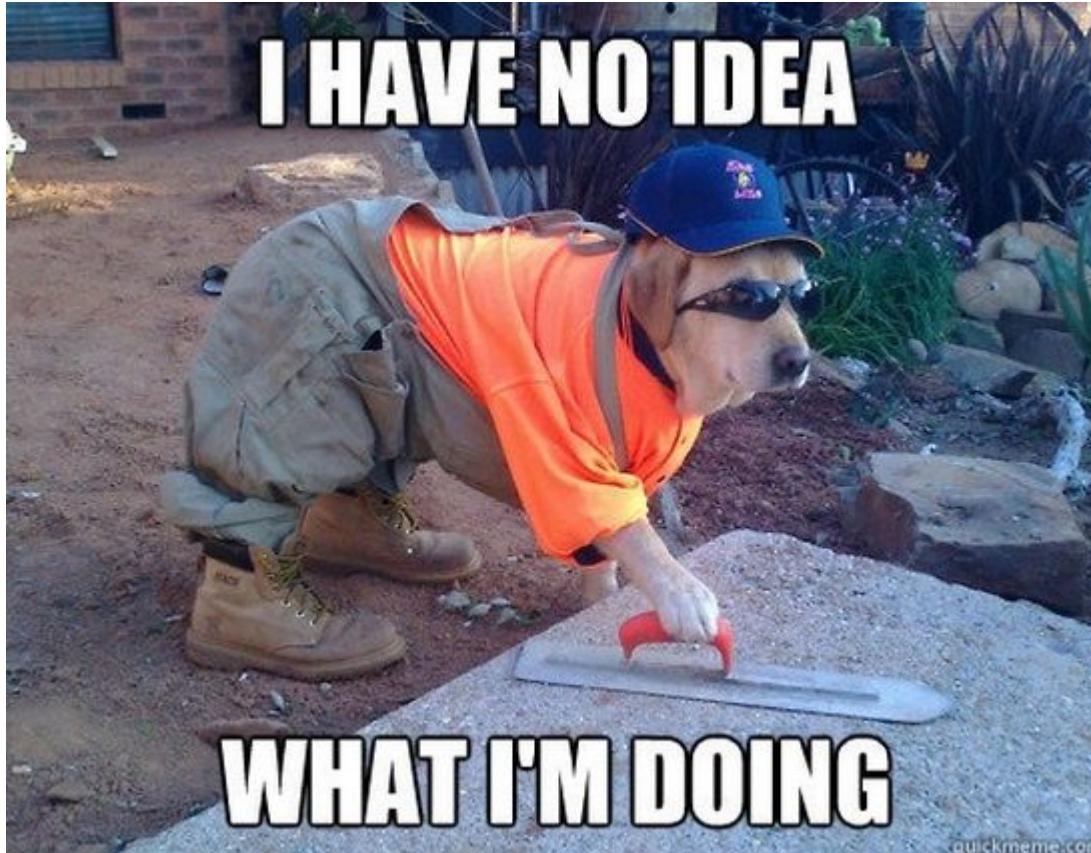
Tracy Walker

Howdy



- 31 yrs
- 27 yrs
- 19 yrs
- 10 yrs
- 3 yrs
- 2 yrs
- 9 months
- 3 months

DEFECTDOJO



“What’s some normal stuff I’ve used an LLM to do?”

- Itinerary when visiting a new City
- Packing lists for “special” trips
- Summarize foreign language meetings
 - Recording > Whisper > .txt > LLM > Summary
- PC troubleshooting
- Replacing bad PC motherboard, updating BIOS
- Electrical Wiring Education
- Calculating OneWheel charge time to specific %
- Responding intelligently to familial political arguments
- Restoring antique rusty dumbbells
- 2024 Taxes

“What I’ve really been using A.I. to do?”

- Bash scripts
- Python scripts
- Parser Analysis - Supported scanners, file types
- Pandemic / Disaster Plan
- Extracting Slack .json into a FAQ sorted by most popular Q?s
- Documentation template
- Documentation Updates <<- HUGE
- Parser Deep Analysis - Data parsed / dropped
- Parser Generation

Extending an Open Source Tool Using AI

A Journey (Our Agenda)

1. What's the OSS Tool?
2. How can it be “extended”?
3. What was I trying to do?
4. Start: The Quest to Build a Parser
5. Side Quest 1: Parser Docs
6. Side Quest 2: Hostile Liquid
7. Boss Battle: Fuego Rapido
8. Bonus Quest (Time/Talk): Dojo Docs

01

What Is DefectDojo?

What is DEFECTDOJO?

- Unified Vulnerability Management Platform
- Created by AppSec Professionals in 2014
- OWASP Flagship Project 10 years
- ~40 Million downloads
- 400+ Contributors
- GitHub Top 25 Open Security Project
- Adaptable Data Model / django Architecture
- 200+ Integrations = PARSERS!!
- OSS scalable to 20k+ findings
- Pro scalable to 20M+ findings, SaaS, UI, etc.

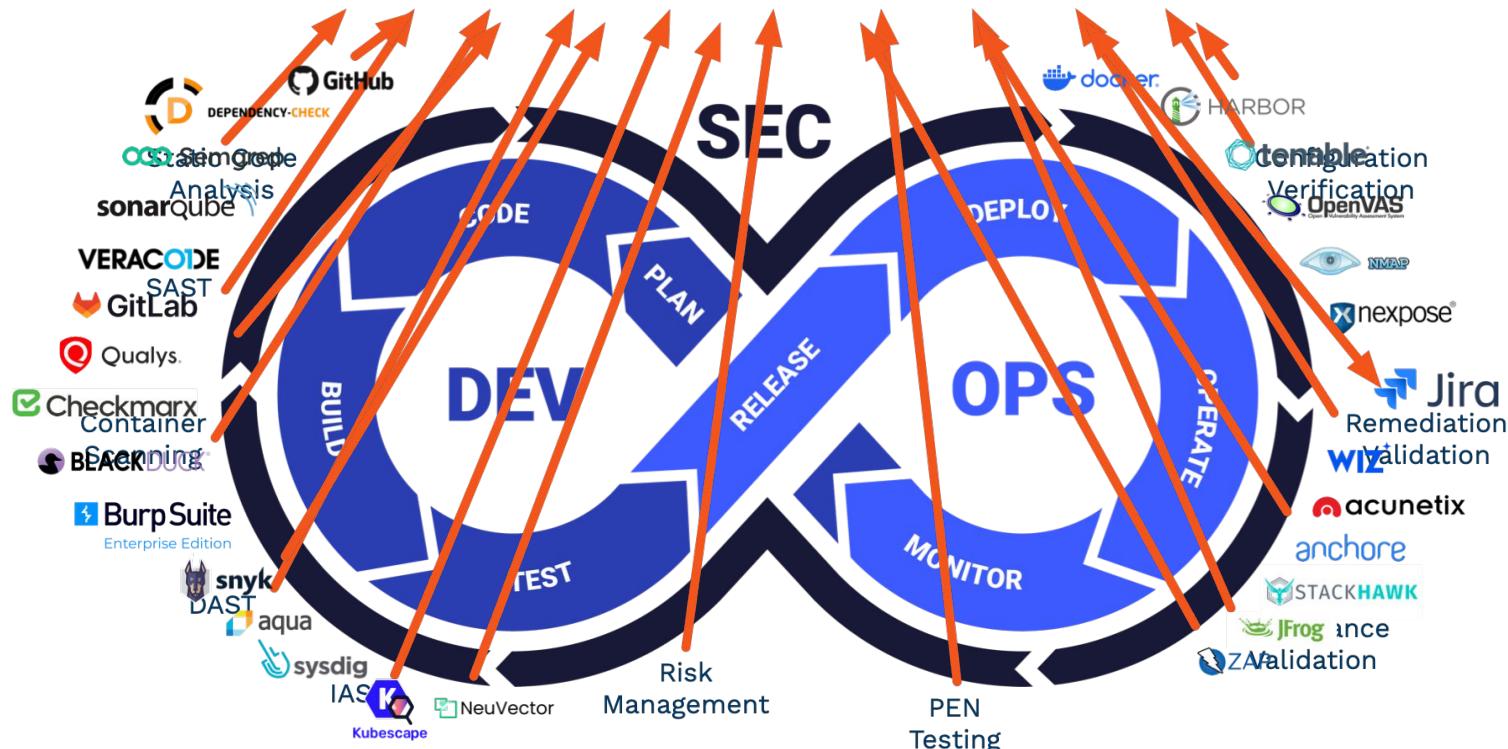
[www.DefectDojo.com \(OSS & Pro\)](http://www.DefectDojo.com (OSS & Pro))



The screenshot shows the DefectDojo website homepage. At the top, there's a navigation bar with links for Platform, Integrations, Pricing, Resources, Company, Log In, and Contact Us. The main header features the text "Unify Connect Secure" above a sub-header "DevSecOps, ASPM, Vuln Management All on one platform". A prominent orange "Learn more" button is located below the sub-header. To the right, there's a large graphic of a smartphone displaying various security-related icons like a lock, a gear, and a chart. Below this graphic, three circular icons represent different features: "Connect the Dots", "Unify, Distill, & Act", and "Unlock DevSecOps Scalability". Each feature has a list of benefits. At the bottom of the main section, it says "30 Million+ Downloads, Trusted by Top Organizations, 180+ Supported Security Tools". The footer section is titled "Elevate your security with DevSecOps Orchestration" and contains a diagram showing a central blue square connected to four white boxes containing icons of a person, a gear, a globe, and a chart. Below the diagram, there's a paragraph about DevSecOps automation and a "See Platform" button.

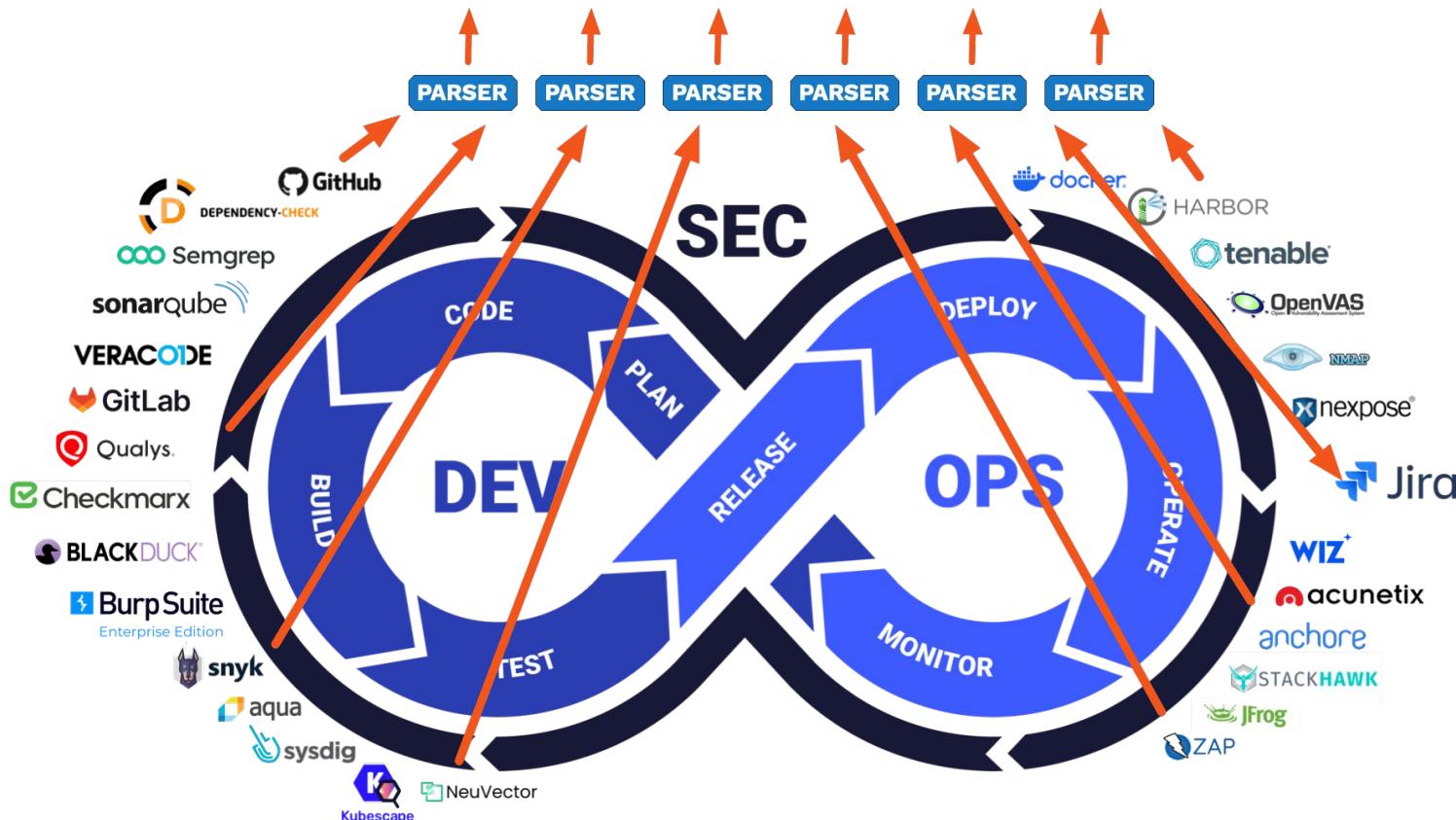
DEFECTDOJO A Unified Vulnerability Management Platform

DEFECTDOJO



200+ Supported Security Tools

DEFECTDOJO



200+ Supported Security Tools

What was I trying to do?

03

1. Learn how to build a DefectDojo parser
2. Save engineering time & effort for creating parser
3. Learn to use an AI LLM for something “real”
4. Contribute to DefectDojo & Security Community
5. Experience DefectDojo OSS from community Pov
6. Collaborate / Share / Learn New Ideas

04

The Quest To Build A Parser

As a Community Member

Tools & Code

1. 100% open source sources & outputs
Code: <https://github.com/DefectDojo/django-DefectDojo>
Documentation: <https://docs.defectdojo.com/>
2. Dev Environment - MBPro / Docker / Github DefectDojo
3. Claude.ai Pro Personal Subscription (\$20.76/Month)
4. ChatGPT / DeepSeek / Google Gemini
5. CLI commands - docker / git / vi / python

Rules?

1. No proprietary *anything* uploaded to LLM.
2. Approach as community member - No special help.
3. No human generated code - prompts only.
4. Yes copy-n-paste, debugging, editing.
5. Repeatable / Automatable as much as possible
6. Create something of value - not just time saved

Parser Specifics

- 200+ Security Scanners / 300+ File Formats
- 100% Open Source - Used by DefectDojo OSS and PRO versions
- 100% Developed by community & DefectDojo engineering
- Varying data mapping detail / documentation / parsing functions
- No Mapping Details

parser.py	Actual parser written in Python	
unittests/scans/{many, no, one_vuln}.json	Sample files for Unit Tests	<input checked="" type="checkbox"/>
unittests/tools/test_parser.py	Unit Tests	<input checked="" type="checkbox"/>
dojo/settings/settings.dist.py	Hash based deduplication settings	<input checked="" type="checkbox"/>
connecting_your_tools/parsers/file/doc.md	Documentation for Parser	<input checked="" type="checkbox"/>

django-DefectDojo / unittests / scans /		
api_bugcrowd	Api Bugcrowd: Fix handling of invalid endpoint (#8289)	2 years ago
api_cobalt	Refactor API parsers (#7002)	2 years ago
api_edgescan	Refactor API parsers (#7002)	2 years ago
api_sonarqube	Refactor API parsers (#7002)	2 years ago
api_vulners	Refactor API parsers (#7002)	2 years ago
appcheck_web_application_scanner	appcheck-severity-determination-fix Use v4, v3, v2 cvss vectors for s...	5 months ago
appspider	move unittests outside dojo folder (#5527)	4 years ago
aqua	⚡ add epss for aqua parser #10849 (#10855)	5 months ago
arachni	Upgrade to FontAwesomeFree 6.2.1 (#7304)	3 years ago
astff	Update ASFF parser to create endpoints (#9346)	last year
auditjs	fix(auditjs-parser): error when npm package has scope (#6596)	3 years ago
aws_inspector2	Fixing call to date_parser in AWS Inspector2 parser (#11767)	4 days ago
aws_prowler	add scan_date fix also for reimport, fix validation (#5574)	4 years ago
aws_prowler_v3plus	prowler_v4.py Prowler v4.5.0 changed the 'event_time' key with 'time_...	3 months ago
awssecurityhub	AWS Security Hub: Accommodate for reports with missing AccountID (#11...	4 months ago
azure_security_center_recommendations	move unittests outside dojo folder (#5527)	4 years ago
bandit	Changing name of WhiteSource to Mend (#9359)	last year
bearer_cli	⚡ added parser for Bearer CLI (#9672)	11 months ago
blackduck	move unittests outside dojo folder (#5527)	4 years ago
blackduck_binary_analysis	Parser - Black Duck Binary Analysis (#9163)	2 years ago
blackduck_component_risk	move unittests outside dojo folder (#5527)	4 years ago

<https://github.com/DefectDojo/django-DefectDojo/tree/master/unittests/scans>

1st (failed) Approach to LLM generated a parser

- ❖ LLM's can “read” markdown, html, text, json, python
- ❖ Build documentation, finding class, parser documentation
- ❖ How many .CSV Parser code & example files exist
- ❖ Use example parsers as primary reference
- ❖ Generate parser, build, test, refactor, repeat
- ❖ Generate unit test
- ❖ Generate documentation

1st step: Use Claude.ai to Find Existing .csv Parsers

	A	B	C	D	E	F	G
1	Scanner Name	Type	Opensource?	JSON	XML	CSV	XLSX
2	AWS Prowler Scanner	INFRA	Open Source	X		X	
3	Azure Security Center Recommendation:	INFRA	Commercial			X	
4	Blackduck Binary Analysis	SCA	Commercial			X	
5	Blackduck Hub	INFRA	Commercial			X	
6	Bugcrowd	OTHER	Commercial			X	
7	Cobalt.io Scan	DAST	Commercial			X	
8	Contrast Scanner	SAST	Commercial			X	
9	CredScan Report	SAST	Commercial			X	
10	Deepfence Threatmapper		Open Source			X	X
11	Generic Findings Import	OTHER	Open Source	X		X	
12	HackerOne Cases	OTHER	Commercial	X		X	
13	IntSights Report	OTHER	Commercial	X		X	
14	Kiuwan Scanner (SAST)	SAST	Commercial			X	
15	OpenVAS Parser	INFRA	Open Source		X	X	
16	PMD Scan	SAST	Open Source			X	
17	Qualys Hacker Guardian Scan	INFRA	Commercial			X	
18	Qualys Scan	DAST	Commercial		X	X	
19	SKF Scan	OTHER	Open Source			X	
20	Solar Appscreener Scan	DAST	Commercial			X	
21	Sysdig Vulnerability Reports	INFRA	Commercial	X		X	
22	Tenable	DAST	Commercial			X	
23	Testssl Scan	INFRA	Open Source			X	
24	Trustwave	DAST	Commercial			X	
25	Twistlock	INFRA	Commercial	X		X	
26	Veracode SourceClear	SCA	Commercial	X		X	
27	Wiz Scanner	INFRA	Commercial			X	
28	42Crunch Scan	DAST	Commercial	X			
29	Acunetiv Scanner	DAST	Commercial	X	X		

1st Attempt

- Loaded .csv files, parser documentation, parser examples
- Required multiple changes before successful build
- Missing several data fields - fixed to import all data fields
- Unit tests failed to execute
- LLM began repeating errors / hallucinating

Lessons Learned

- Prompt order is critical
- LLM does not consistently “link” data relationships between prompts
- Troubleshooting / Error Correction too high
- Chats cannot be too long or complex
- Something called “chunking”



ChatGPT

LLM Specifics: * Claude

Gemini

LLM Definitions

- Context Window - Amount of text LLM can use as input
- Tokenization - input & output broken down to smaller units
- Prompt Engineering (A.I. Jockey)
- Hallucinations - factually incorrect or **not present in input data.**
- Temperature - Controls “randomness” of output (low = 0.2 / high = 0.8)
- Attention - Weight of different Tokens within Context Window

Additional LLM Lingo

- Chunking - Breaking input into smaller pieces (also applies to problems)
- Branching - Editing a prompt and generating new branch of output
- “No Shot” / “Few Shot” - Examples given
- “Vibe” coding
- “Slopsquatting”

Vibe Coding

- Feels.. Negative / Sarcastic?
- Refers to “Code Completion”
- Ignores QA / Testing
- Copy-n-paste errors? Yes!!
- Random in / Random out



Andrej Karpathy ✅
@karpathy



...

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

12:17 AM · Feb 3, 2025 · 4.1M Views

https://en.wikipedia.org/wiki/Vibe_coding

How Slopsquatting Attacks Work



- 01 AI Hallucinations: AI code generators suggest non-existent packages.
- 02 Supply Chain Attack: Malicious code becomes part of the project.
- 03 Malicious Registration: Attackers register these packages with malicious code.
- 04 Blind Installation: Developers install these malicious packages.

New Approaches Practices

- Break parser problem down into small chunks - multiple chats
- Full Shot (*examples for input / parser code / output*)
- Branch (Edit) to achieve high quality responses with each prompt
- Use LLM to generate output to be used for new chats.
 - (Data Mapping index file to save LLM from having to do the extra lift every time!)
- When hitting error, debug to resolution, then edit the prompt where the error occurred to cause the chat to branch and “skip” troubleshooting.

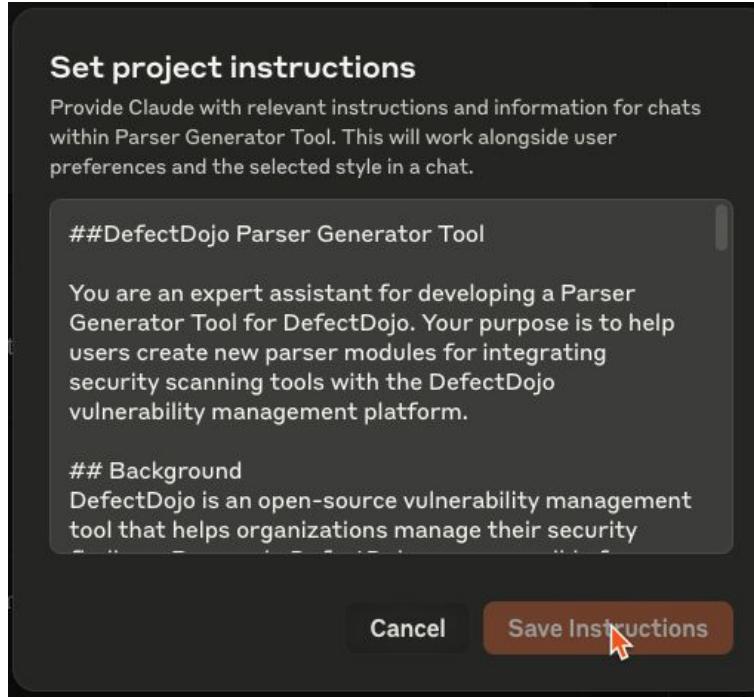
2nd Attempt (successful!)

- Claude Projects!! (Store artifacts reused by new conversations)
- Reused previous parser prompts but with much more detail and planning/approach.
- Used 7 detailed mapping documents for many example parsers
- Generated mapping for new parser 1st as design reference, *then* generate code

Lessons Learned

- Use AI to map data elements to feed back into AI as reference
- Compartmentalize documentation
- Generated dev environment metadata so LLM understands the env
- Back-up above errors within prompt tree to start new branches
- Claude Message Limits Reset Every 5 Hours

Claude Project



The screenshot shows a list of items under the heading "Project knowledge". The top item is a summary: "This project is related to DefectDojo vulnerability manageme... Edit". Below it, a progress bar indicates "46% of knowledge capacity used". The list of items includes:

- DefectDojo-branching-model (MD, 1 day ago)
- How to develop a DefectDojo Parser (MD, 1 day ago)
- parser-documentation-template (MD, 1 day ago)
- trustwave_findings (JSON, 1 day ago)
- trustwave-mapping (MD, 1 day ago)
- trustwave_parser (PY, 1 day ago)
- wiz-mapping (HTML, 1 month ago)
- wiz_parser (PY, 1 month ago)
- wiz_findings (JSON, 1 month ago)
- qualys_hacker_guardian_parser (PY, 1 month ago)

Claude Project

ΔΩ Parser Generator Tool / Setting Up Parser Development for DefectDojo ▾

TW Can you help me set the project instructions, given the content of the files currently loaded in this project's knowledge capacity?

I'd be happy to help you set the project instructions for your "Parser Generator Tool" for DefectDojo. Based on the files you've shared, I can see this is about creating a tool to

ΔΩ Parser Generator Tool / Setting Up Parser Development for DefectDojo ▾

TW Can you help me set the project instructions for the claude.ai project "Parser Generator Tool"? If you review the content of the files currently loaded in this project's knowledge capacity, I believe you will know what I should set as your project instructions. To be clear, I am not looking for project instructions, but rather am asking you to help me set the project instructions for this project within claude.ai.

Managing threat security findings. This is done in DefectDojo, the responsibility for ingesting reports from various security scanning tools and converting them into a standardized format within the DefectDojo database.

Remember the Existing .csv Parsers?

	A	B	C	D	E	F	G
1	Scanner Name	Type	Opensource?	JSON	XML	CSV	XLSX
2	AWS Prowler Scanner	INFRA	Open Source	X		X	
3	Azure Security Center Recommendation:	INFRA	Commercial			X	
4	Blackduck Binary Analysis	SCA	Commercial			X	
5	Blackduck Hub	INFRA	Commercial			X	
6	Bugcrowd	OTHER	Commercial			X	
7	Cobalt.io Scan	DAST	Commercial			X	
8	Contrast Scanner	SAST	Commercial			X	
9	CredScan Report	SAST	Commercial			X	
10	Deepfence Threatmapper		Open Source			X	X
11	Generic Findings Import	OTHER	Open Source	X		X	
12	HackerOne Cases	OTHER	Commercial	X		X	
13	IntSights Report	OTHER	Commercial	X		X	
14	Kiuwan Scanner (SAST)	SAST	Commercial			X	
15	OpenVAS Parser	INFRA	Open Source		X	X	
16	PMD Scan	SAST	Open Source			X	
17	Qualys Hacker Guardian Scan	INFRA	Commercial			X	
18	Qualys Scan	DAST	Commercial		X	X	
19	SKF Scan	OTHER	Open Source			X	
20	Solar Appscreener Scan	DAST	Commercial			X	
21	Sysdig Vulnerability Reports	INFRA	Commercial	X		X	
22	Tenable	DAST	Commercial			X	
23	Testssl Scan	INFRA	Open Source			X	
24	Trustwave	DAST	Commercial			X	
25	Twistlock	INFRA	Commercial	X		X	
26	Veracode SourceClear	SCA	Commercial	X		X	
27	Wiz Scanner	INFRA	Commercial			X	
28	42Crunch Scan	DAST	Commercial	X			
29	Acunetiv Scanner	DAST	Commercial	X	X		

Mapping Parsers

Purpose: Generate detailed mapping of existing parsers

Identify all .csv parsers

Import .csv into DefectDojo

.csv Input -> Parser

Parser -> Finding Object

Finding Object -> .json Output

Copy & Rename 3 files

Upload into Claude Project

```
tracywalker@Tracys-M1-Max-MacBook-Pro:~/Development/DEV_defectdojo/
Parser Import and API Retrieval Tool
Current Configuration:
Product Type: Parser-Dev
Product: Parser-Products
Engagement: one-vuln-parser

Available Parsers:
1) aws_prowler
2) azure_security_center_recommendations
3) blackduck
4) blackduck_binary_analysis
5) bugcrowd
6) cobalt
7) cred_scan
8) h1
9) insights
10) kiwan
11) openvas
12) pmd
13) qualys_hacker_guardian
14) solar_appscreener
15) sysdig_reports
16) tenable
17) testssl
18) trustwave
19) twistlock
0) Exit
Enter choice [0-      19]: █
```

Output for Input!

Purpose: Generate detailed mapping of existing parsers

Identify all .csv parsers

Import .csv into DefectDojo

.csv Input -> Parser

Parser -> Finding Object

Finding Object -> .json Output

Copy & Rename 3 files

Upload into Claude Project

-rw-r--r--	1	tracywalker	staff	2699	Jan 14 02:37	insights_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:26	openvas_findings.json
-rw-r--r--	1	tracywalker	staff	1594	Jan 14 02:37	openvas_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	733	Jan 14 02:37	openvas_parser.py
-rw-r--r--	1	tracywalker	staff	959	Jan 13 23:17	parsers.txt
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:26	pmd_findings.json
-rw-r--r--	1	tracywalker	staff	383	Jan 14 02:37	pmd_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	2044	Jan 14 02:37	pmd_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:26	qualys_hacker_guardian_findings.json
-rw-r--r--	1	tracywalker	staff	2546	Jan 14 02:37	qualys_hacker_guardian_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	2681	Jan 14 02:37	qualys_hacker_guardian_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:26	solar_appscreener_findings.json
-rw-r--r--	1	tracywalker	staff	2595	Jan 14 02:37	solar_appscreener_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	2004	Jan 14 02:37	solar_appscreener_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:26	sysdig_reports_findings.json
-rw-r--r--	1	tracywalker	staff	782	Jan 14 02:37	sysdig_reports_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	11879	Jan 14 02:37	sysdig_reports_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:28	tenable_findings.json
-rw-r--r--@	1	tracywalker	staff	2803	Jan 14 03:28	tenable_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:28	testssl_findings.json
-rw-r--r--	1	tracywalker	staff	214	Jan 14 02:37	testssl_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	3716	Jan 14 02:37	testssl_parser.py
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:29	trustwave_findings.json
-rw-r--r--@	1	tracywalker	staff	774	Jan 14 03:29	trustwave_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	2720	Jan 14 03:29	twistlock_findings.json
-rw-r--r--	1	tracywalker	staff	657	Jan 14 02:37	twistlock_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	6626	Jan 14 02:37	twistlock_parser.py
-rw-r--r--	1	tracywalker	staff	2641	Jan 14 02:37	wiz_one_vuln.csv
-rw-r--r--	1	tracywalker	staff	8075	Jan 14 02:37	wiz_parser.py

Press enter to continue

CSV Parser File Copier

1. Find and copy CSV parser files
2. List current parser files
3. Clean output directory
0. Exit

Enter choice [0-3]: 0

05

Side Quest: Parser Mapping & Documentation

Parser Documentation

Extending Open Source Tools using AI

32

Before

The screenshot shows a web browser displaying the DefectDojo documentation. The URL in the address bar is [https://defectdojo.com/en/](#). The page title is "Trustwave". The main content area displays the following text:

CSV output of Trustwave vulnerability scan.

Sample Scan Data

Sample Trustwave scans can be found [here](#).

Navigation links at the bottom left and right:

- Left: "← Prev" (Trufflehog3)
- Right: "Next →" (Trustwave Fusion API Scan)

The left sidebar contains a navigation menu with the following sections and items:

- About DefectDojo**
 - About Our Documentation
 - New User Checklist
 - Pro Features List
 - Contact DefectDojo Support
 - Beta UI Features
 - Request a DefectDojo Pro Trial
- Connect Your Tools**
 - Import Method Comparison
 - External Tools: Universal Importer & DefectDojo-CLI
 - How To Import Data** >
 - API Connectors** >
 - Universal Parser
- Supported Report Types** >
 - Generic Findings Import
 - Files** >
 - Acunetix Scanner
 - Anchore Enterprise Policy Check
 - Anchore Grype
 - Anchore-Engine
 - AnchoreCTL Policies Report
 - AnchoreCTL Vuln Report
 - AppCheck Web Application Scanner

Before:

A screenshot of a GitHub pull request page for the file `trustwave.md`. The pull request has 3 people watching and is titled "OS Docs Overhaul - theming change, many new articles (#11...)" by user `8bede6f` 5 months ago. The CSV file contains the following data:

title	toc_hide
Trustwave	true

CSV output of Trustwave vulnerability scan.

Sample Scan Data

Sample Trustwave scans can be found [here](#).

Mapping Parsers

Generate detailed mapping index for example parsers

Step 1 - Create .json Output

- A. .csv Input -> Parser
- B. Parser -> Finding Object
- C. Finding Output -> .json

Step 2 - Load 3 files + prompt



The screenshot shows a dark-themed application window titled "Parser Generator Tool / trustwave finding mapping". At the top, there are three cards representing different file types: "trustwave.csv", "trustwave.py", and "trustwave.json". Below these cards is a text input area with the following content:

TW You are an expert in comparing data fields and generating documentation that can be used in the future to identify how DefectDojo Parsers input .csv files and parse into a DefectDojo Finding Object.
I want to create a data field mapping document between the trustwave.csv file imported into defectdojo and the trustwave.json data that represents what was parsed into the finding object class.
To be clear, I used a DefectDojo "reimport" API call to parse the data in the trustwave.csv file, and then use an API call to export the same finding object data from defectdojo into the trustwave.json file. This gives you the input, the parser code, and the output to do the mapping.
Please identify every field in the trustwave.csv file, listing every field, and provide a total count.
Identify whether or not each data field is parsed into one of the finding fields in the trustwave.json file. If the field is not parsed, note no parser function. Note any special conditions and defaults. Double check that every data field is identified, and that the total number of mapped fields and unmapped fields equate to the total number of fields.
Finally, review the trustwave.py file to understand which functions are performing the parsing, and add the line number in the parser file where the field is being parsed. Call that column "Parser Line #"
Please create the mapping document in .html format for me to review. If it passes my review, I will ask you to recreate the exact same document in a markdown (.md) format.
For this step, please create just the mapping document in .html so I can review your analysis.

Mapping Parsers

Trustwave CSV to DefectDojo Finding Field Mapping

CSV Field Mappings

Total Fields in CSV: 13

CSV Field	Finding Field	Parser Line #	Notes
Status	Not Mapped	N/A	Present in CSV but not used in parser
IP	endpoints host	34-37	Used as fallback if Domain is empty
Target Name	Not Mapped	N/A	Present in CSV but not used in parser
Domain	endpoints host	34-37	Primary choice for host, falls back to IP if empty
Vulnerability Name	title	42	Direct mapping to finding title
Description	description	43	Direct mapping to finding description
Remediation	mitigation	45	Direct mapping to finding mitigation
Protocol	endpoints protocol	40-41	Added to endpoint if not empty
Port	endpoints port	38-39	Converted to integer and added to endpoint if not empty
Severity	severity	48	Mapped through severity_mapping dictionary: I=Info, L=Low, M=Medium, H=High, C=Critical. Defaults to Low if not matched.
CVE	unsaved_vulnerability_ids	49	Added to vulnerability IDs list
Service	Not Mapped	N/A	Present in CSV but not used in parser
Evidence	references	44	Direct mapping to finding references

Summary

- Total CSV Fields: 13
- Mapped Fields: 10
- Unmapped Fields: 3 (Status, Target Name, Service)

Additional Finding Field Settings

Finding Field	Value	Parser Line #	Notes
test	test parameter	29	Set from test parameter passed to get_findings
nb_occurrences	1 (initial)	30	Initialized to 1, incremented for duplicates

Processing Notes

- Deduplication is performed using SHA256 hash of severity + title + description
- For duplicate findings, nb_occurrences is incremented rather than creating a new finding
- The parser uses csv.DictReader with comma delimiter and quote character
- All empty string values ("") are treated as None values

Index
About DefectDojo
About Our Documentation
New User Checklist
Contact DefectDojo Support
 Beta UI Features
Request a DefectDojo Pro Trial
 Connect Your Tools
Import Method
Comparison
External Tools: Universal Importer & DefectDojo-CLI
How To Import Data
Import Scan Form
Add new Findings to Trial
 Connect Your Tools
Import Method
Comparison
External Tools: Universal Importer & DefectDojo-CLI
How To Import Data
Import Scan Form
Add new Findings to Trial
 Connect Your Tools
Import Method
Comparison
External Tools: Universal Importer & DefectDojo-CLI
How To Import Data
Import Scan Form
Add new Findings to a Test via Reimport
Smart Upload for infrastructure scans
Create an automated import pipeline via API
API Connectors
About Connectors

Trustwave

File Types

Trustwave vulnerability scan reports can be exported/imported in CSV format.

Total Fields in CSV: 13

Fields in order of appearance:

1. Status (Not mapped)
2. IP - Used for endpoint host if Domain is empty
3. Target Name (Not mapped)
4. Domain - Primary choice for endpoint host
5. Vulnerability Name - Maps to finding title
6. Description - Maps to finding description
7. Remediation - Maps to finding mitigation
8. Protocol - Added to endpoint if present
9. Port - Added to endpoint port if present, converted to integer
10. Severity - Mapped through severity levels:
 - o I = Info
 - o L = Low
 - o M = Medium
 - o H = High
 - o C = Critical
11. CVE - Added to vulnerability IDs list
12. Service (Not mapped)
13. Evidence - Maps to finding references

Field Mapping Details

For each finding created, the parser:

- Creates endpoints by combining Domain/IP, Port, and Protocol fields
- Sets default nb_occurrences to 1, incremented for duplicates
- Uses SHA256 hash of severity + title + description for deduplication
- Defaults severity to Low if mapping not matched

Sample Scan Data

Sample Trustwave scans can be found in the [unit tests folder](#).

Link To Tool

Trustwave provides vulnerability scanning services through their SecureConnect platform.

Pull Request!

The screenshot shows a GitHub pull request page for a repository named "django-DefectDojo". The pull request is titled "docs: Updating Trustwave parser documentation with detailed field mappings" and has the identifier "#11880". The status of the pull request is "Open", and it is described as "skywalke34 wants to merge 1 commit into DefectDojo:dev from skywalke34:docs/trustwave-parser".

The main content of the pull request includes:

- Description:** Updates the Trustwave parser documentation with detailed field mappings and additional information. Includes:
 - Total fields and their mapping details
 - CSV field order and descriptions
 - Parser behavior details
 - Links to sample data and tool information
- Test Steps:**
 - Documentation change only
 - Verified locally using Hugo server
 - Verified by human and AI LLM review
- Checklist:**
 - Documentation changes only

On the right side of the pull request page, there are several sections showing status and metadata:

- Reviewers:** No reviews. Still in progress? Convert to draft.
- Assignees:** No one assigned.
- Labels:** docs
- Projects:** None yet.
- Milestone:** No milestone.
- Development:** Successfully merging this pull request may close these issues. None yet.
- Notifications:** Customize. Unsubscribe.

06a

Side Quest: CodeQL

*“It’s easy to smoosh stuff together...
Separating smooshed stuff? Very difficult.
Let’s just eat it.” -me*

Display giant screenshot separately from presentation
Note the “Help” data element has multiple data pieces embedded within the same “help” field. Universal Parser cannot split data out of a single element.

The image shows two overlapping screenshots. On the left is a GitHub code scanning alert for file `dojoutil.py`, specifically line 1385. The alert title is "Uncontrolled data used in path expression". It highlights a line of code: `eng_path.open("wb") as destination:`. A red dashed box encloses the entire alert content, which includes severity (High), affected branches (master), tags (correctness, security), and weaknesses (CWE-22, CWE-28, CWE-36, CWE-43, CWE-99). Below the alert is a "Recommendation" section with guidelines for validating user input. An "Example" section provides a detailed explanation of how uncontrolled data can lead to file access vulnerabilities.

The right side of the image shows a DefectDojo interface for the same alert. A red dashed box highlights the "help" field, which contains multiple pieces of information: "Numerous data elements cannot be parsed out of 'help'", "Mitigation", "Impact", "References", "Active", "Verified", "False Positive", "Duplicate", "EPSS Score", "EPSS Percentile", "Unique ID From Tool", "Val ID From Tool", "Tags", "Endpoints", and "Vulnerability IDs". The "Cannot Parse Integer to string" message is also highlighted in this field.

06

Side Quest: Hostile Liquid

(Attempting a parser for Fluid Attacks)

Fluidattacks

Extending Open Source Tools using AI

41

DEFECTDOJO

Test Product Fluid

Overview Components Metrics Engagements 1 Findings Hosts / Endpoints 0/0 Benchmarks Settings

Fluid Attacks Scan processed a total of 15 findings.

Engagements / Fluid Attacks Testing / Fluid Attacks Scan / Test

Fluid Attacks Scan Updated now, Created now

Engagement	Environment	Dates	Updated	Progress	Version	Reimports
Fluid Attacks Testing	Development	Jan. 16, 2025 - Jan. 16, 2025	Jan. 16, 2025	100%	1	

Import History (1) ?

Groups (0)

Findings (15) Critical: 4, High: 11, Medium: 0, Low: 0, Info: 0, Total: 15 Findings

Showing entries 1 to 15 of 15

Column visibility Copy Excel CSV PDF Print Page Size Search:

Severity	Name	CWE Id	Vulnerability	EPSS Score	EPSS Percentile	Date	Age	SLA	Reporter	Status	Group	Planned Remediation
Critical	[68afe6-3c9f-4b37-97d4-5bba99801b8a] (41fc4edf-8d4c-4ceb...)	-		N.A.	N.A.	Sept. 12, 2023	492	-485	Admin User (admin)	Inactive, Verified		
Critical	[68afe6-3c9f-4b37-97d4-5bba99801b8a] (95920384-Ecac-44eb...)	-		N.A.	N.A.	Sept. 12, 2023	492	-485	Admin User (admin)	Inactive, Verified		
Critical	[68afe6-3c9f-4b37-97d4-5bba99801b8a] (1c80ea82-Db06-4852...)	-		N.A.	N.A.	Sept. 12, 2023	492	-485	Admin User (admin)	Inactive, Verified		
Critical	[68afe6-3c9f-4b37-97d4-5bba99801b8a] (Cc24dd71-C681-4fa2...)	-		N.A.	N.A.	Sept. 12, 2023	492	-485	Admin User (admin)	Inactive, Verified		
High	[C7b22920-60df-40e9-B887-99b9e84870bd] (3d457ef0-D8c7-48ec...)	-		N.A.	N.A.	May 8, 2023	619	-589	Admin User (admin)	Inactive, Verified		
High	[Ef9a5d4f-7d14-4d87-B5fb-3cad4c41dcde] (8c10b7fe-f543-4fc5...)	-		N.A.	N.A.	April 7, 2022	1015	-985	Admin User (admin)	Inactive, Verified		

Fluidattacks

Extending Open Source Tools using AI

42

DEFECTDOJO

Test Product Fluid

Overview Components Metrics Engagements 1 Findings Hosts / Endpoints 0/0 Benchmarks Settings

Fluid Attacks Testing / Fluid Attacks Scan / [68afefef6-3c9f-4b37-97d4-5bba99801b8a] (Cc24dd7f-C881-4fa2-b0b0-C2c1b3a4f923) Cuando Se Accede Al Aplicativo Web Se Ob... In - - At https://example.example.co/static/js/7248.d38c018c.chunk.js Last Reviewed today by Admin User (admin), Last Status Update today, Created Sept. 12, 2023, Last Mentioned in (Re)Import: today as created

ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	Vulnerability ID	Found by	Vuln ID from tool
50	Critical (9.3)	-485	Inactive, Verified	Static	Sept. 12, 2023	492 days	Admin User (admin)	-	[68afefef6-3c9f-4b37-97d4-5bba99801b8a] (Cc24dd7f-C881-4fa2-b0b0-C2c1b3a4f923)	Fluid Attacks Scan	cc24dd7f-c881-4fa2-b0b0-c2c1b3a4f923

Location: https://example.example.co/static/js/7248.d38c018c.chunk.js

Component Name: -

Component Version: -

Similar Findings (14) ?

Import History (1) ?

Description:

Finding ID: 68afefef6-3c9f-4b37-97d4-5bba99801b8a

Vulnerability ID: cc24dd7f-c881-4fa2-b0b0-c2c1b3a4f923

Description:

Cuando se accede al aplicativo web se observan credenciales de AWS CloudWatch así como también secretos que permiten generar tokens de cognito.

Specific Details:

REACT_APP_AMPLIFY_USER_POLL_AUTH_SECRET_KEY

Threat:

07

Boss Battle:

Fuego Rapido

(A working parser for RapidFire)

Generating Parsers

Generate detailed mapping
index for example parsers

Step 1 - Create .json Output

- A. .csv Input -> Parser
- B. Parser -> Finding Object
- C. Finding Output -> .json

Step 2 - Load 3 files + prompt

Let's view this live!

Tracy Walker

tracy@defectdojo.com

Questions Comments Thank You!