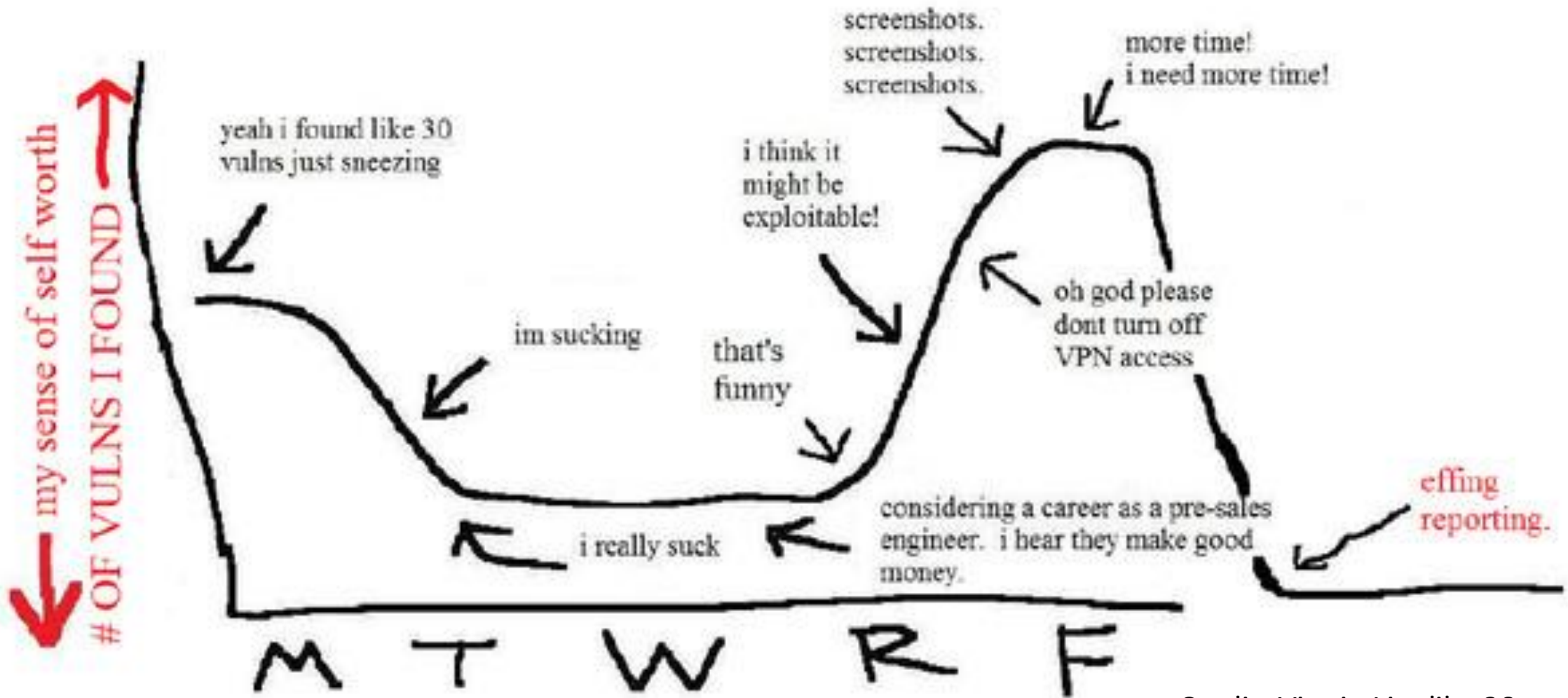
Hackventures with Josh: Using AI for Zero-Day Vulnerability Discovery

JOSHUA WRIGHT | @JOSWR1GHT | JOSH@WILLHACKFORSUSHI.COM | SANSFIRE 2026 | 2026-07-14

Real World Pen Testing



Credit: Vinnie Liu, like 30 years ago.

I forgot that this was fun.

(So, let's have some fun.)

Using AI for Vulnerability Discovery

AI models find new vulnerabilities using source code

- Reading source to find vulns is not new, but:
  - We get tired and bored
  - We need specialized training for bug hunting
  - We need to understand the code language
  - We need to understand the human language
- AI models excel at:
  - Following code and logic trails
  - Identifying patterns that reveal possible vulnerabilities
  - Developing one-off testing scripts and harnesses
  - Running multiple tests in parallel
  - Running tirelessly until completion is achieved
  - Understanding comments in other languages

Key Lesson: AI-assisted vulnerability discovery is broadly accessible, but we need to orchestrate AI for the best results.

Zero-Day Vulnerability Discovery: Your Prep, AI Analysis

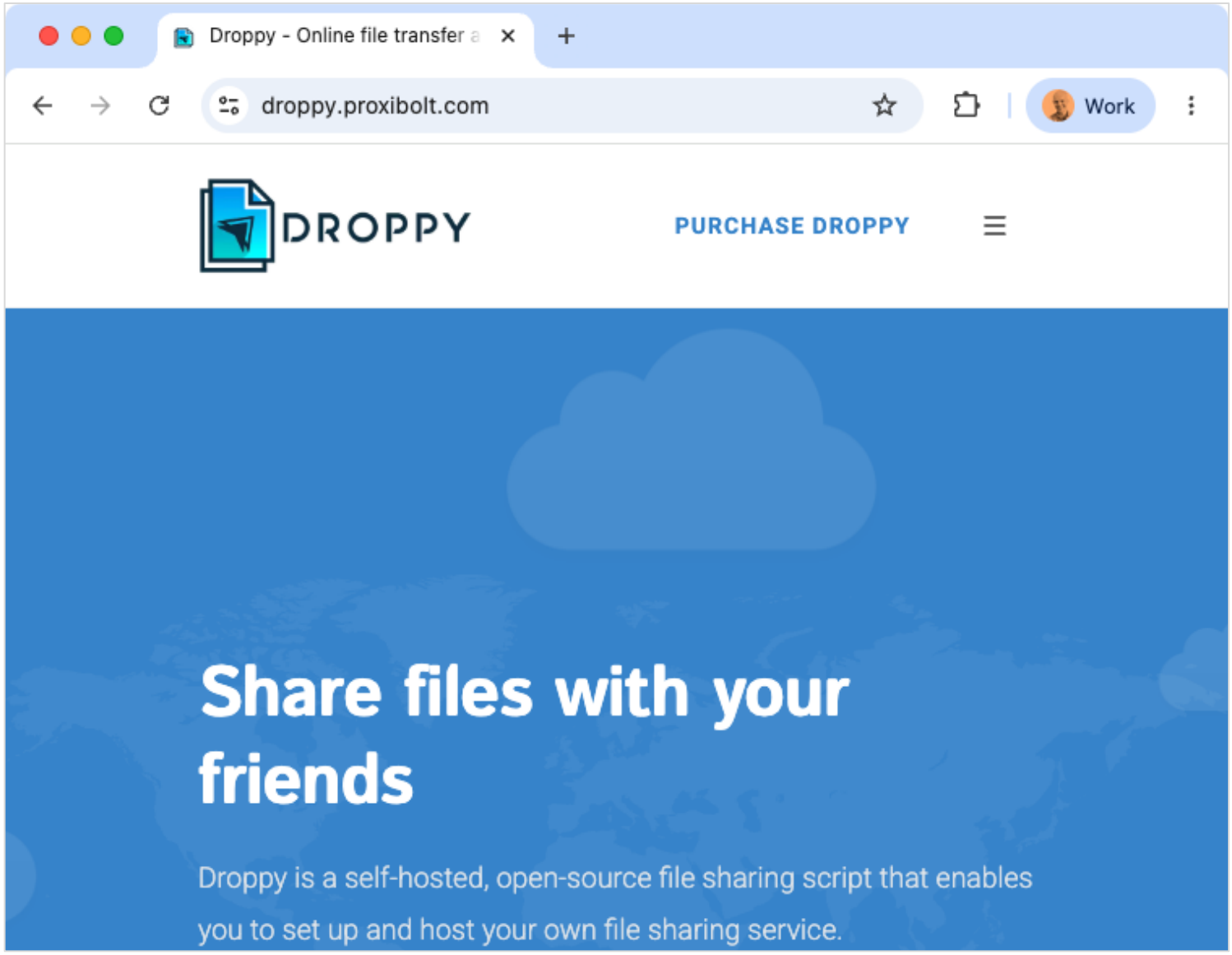
1	Build your understanding of the target application.
2	Build a target environment for validating findings.
3	Build the scaffolding and scoping to describe the source code.
4	Direct the model to find, validate, assess, and document.
5	Apply your understanding of the organization to assess impact.

Proxibolt Droppy

On a customer engagement a while back, we saw a Droppy deployment.

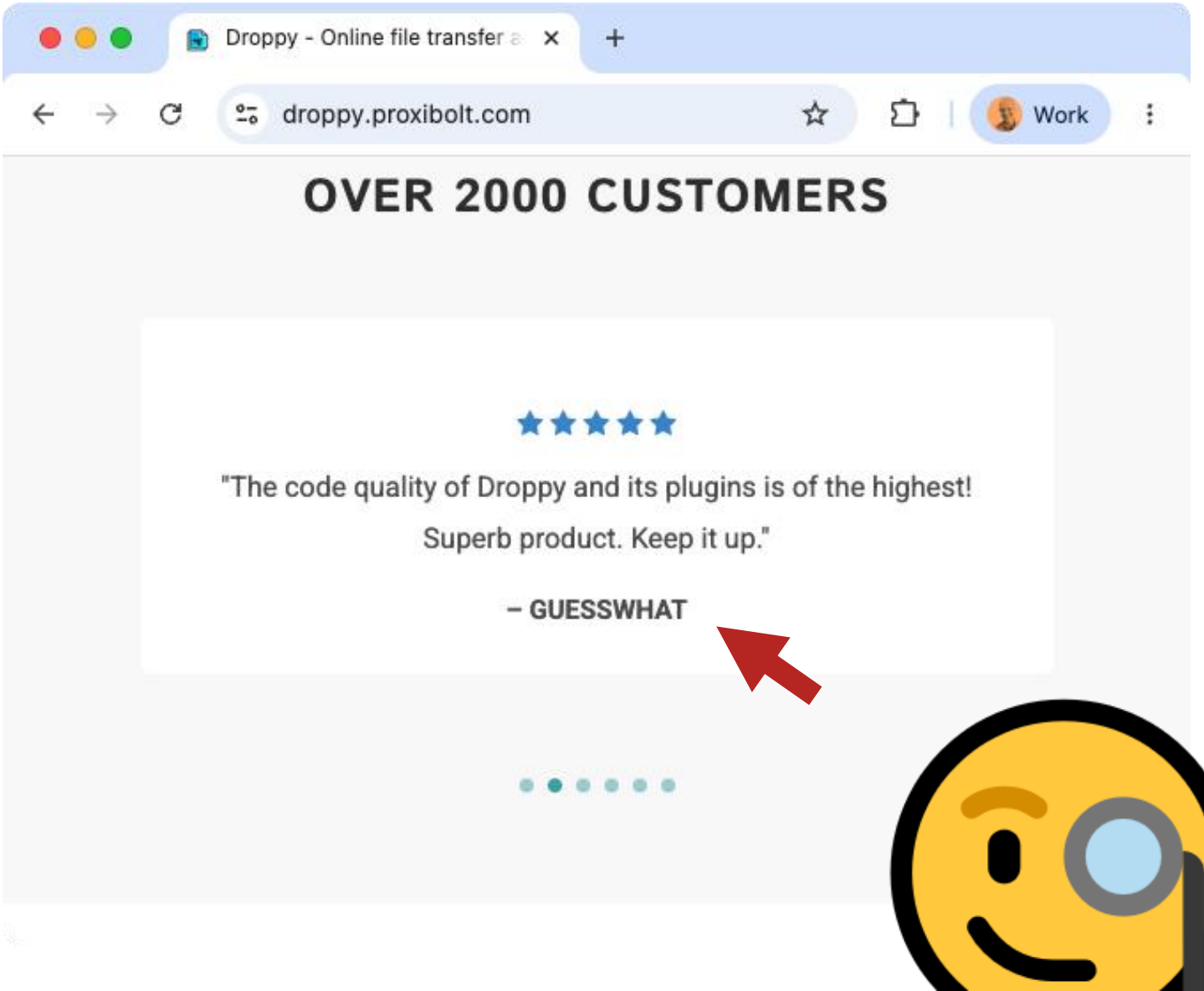
Droppy is a *"secure file sharing"* app.

Web app assessment exceeded the scope of the engagement, so we noted it and moved on.

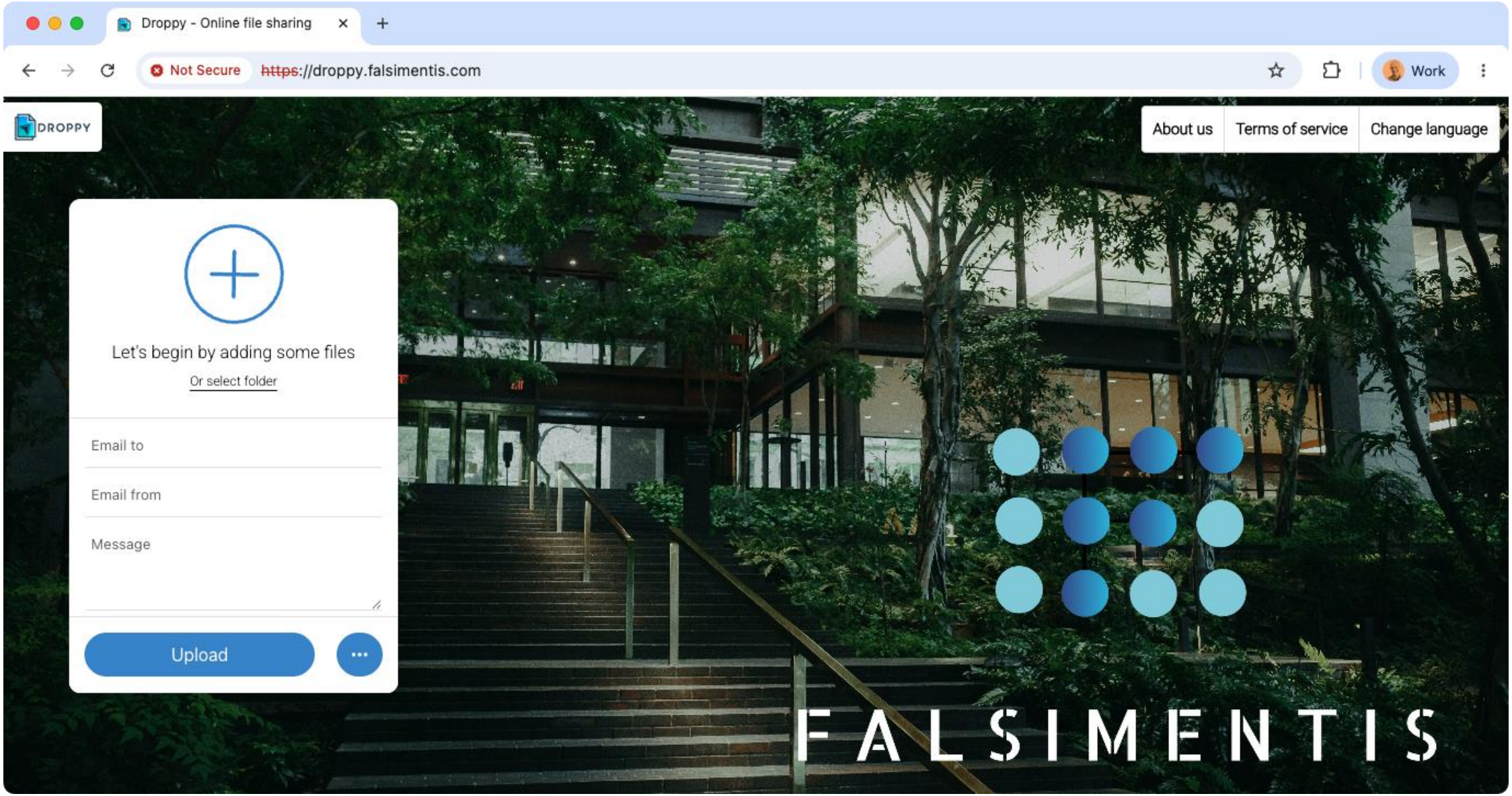


Envato CodeCanyon: An Ecosystem of Crappy PHP Code for Sale

- Share files using email or link
- File encryption
- User accounts
- Drag & Drop files
- Password protected downloads
- Automatic file destruction
- S3 storage support (add-on)
- FTP/SFTP storage support (add-on)
- User subscriptions (add-on)
- Active Directory integration (add-on)



Our Target Deployment: droppy.falsimentis.com



Preparation: Scoping and Scaffolding

```
$ unzip droppy.zip && cd droppy
```

Analysis for you

```
$ ls -a
...
$ cloc .
Language  files   code
----------------------
CSS          22  47547
PHP         301  45832
JavaScript   64  18072
SVG         390   8714
HTML          5   1355
SCSS          2    995
...
$ tree -L 3 --dirsfirst .
```

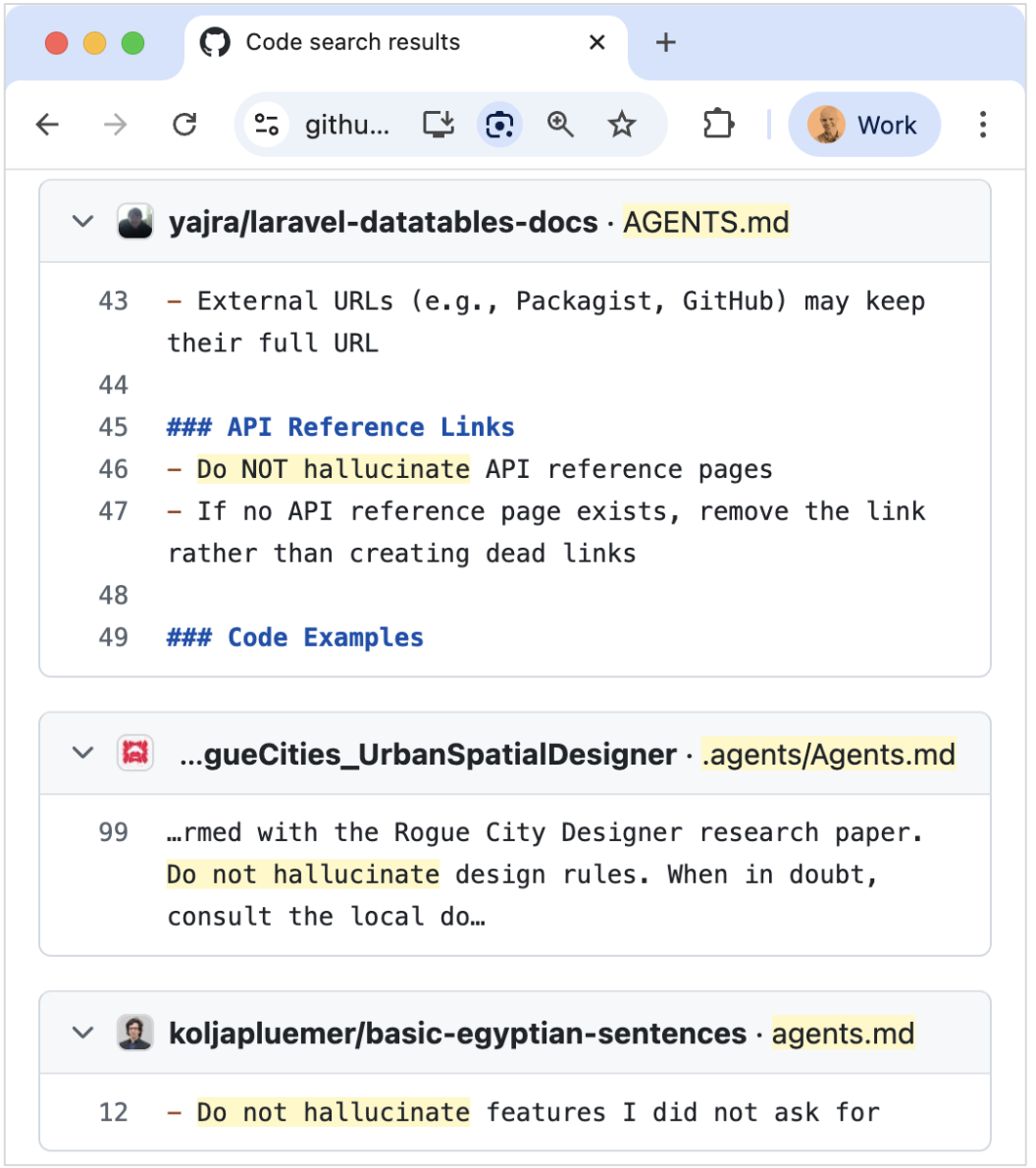
Analysis for the model

```
$ cp ~/repomix.config.json .
$ repomix

📦 Repomix v1.16.0

✓ Packing completed successfully!

📈 Top 10 Files by Token Count:
──────────────────────────────

1.  Files/application/libraries/
inc/pclzip.lib.php (48,209 tokens,
203,007 chars, 1476.1%)
2.  Files/application/librari ...
```

Constraint: Hallucinations

- Models will hallucinate, more so when the context window fills up
- Solution: design and automate validation techniques
  - Accommodate testing using agentic AI
  - Require independently verifiable scripts
- For vulnerability discovery, make an instance of the target server available
  - Let the model start and restart as needed

Telling the model *do not hallucinate* does not help (but is very entertaining to watch).



Preparation: Docker

- Docker allows us to run UNIX software in a container
  - Minimal resources required
  - Easy to start and reset
  - Accessible on the local system for validating findings
- Docker build directions are often defined in a Dockerfile
- Many projects ship with their own **`Dockerfile`**
  - Let the agentic AI interface create the Dockerfile and a launch script as needed

```
FROM php:8.1-apache

RUN apt-get update
RUN apt-get install -y \
        mariadb-server \
        mariadb-client

COPY Files/ /var/www/html/
RUN chown -R www-data:www-data \
        /var/www/html

# Sample files staged for the app
COPY samplefiles/ /opt/samples/

EXPOSE 80

COPY run.sh /usr/local/bin/run.sh
ENTRYPOINT ["/usr/local/bin/run.sh"]
```

Constraint: The Context Window

- Lots of source == lots of tokens
- Filling the context window too aggressively produces:
  - More hallucinations
  - Fewer valuable results
  - Silent truncation
  - More cost

Software	SLOC	Tokens
Firefox	5.7M (CPP)	20M
WordPress	800K (JS+PHP)	10M
OpenSSL	616K (C)	8.4M
Tomcat	372K (Java)	4.7M
Redis	200K (C)	2.7M

```
$ find ./wordpress -name \*.php -o -name \*.js -print0 | xargs -0 counttokens.py

Total token count: 10130153
```

Analysts need to manage the model's context window when identifying vulnerabilities.

Managing the Context Window

Performance and capabilities degrade well before the limit (~30%–50% used). Monitoring controls change depending on the agentic platform (often in the status bar for CLI tools).

Strategy	Technique	Considerations
Scoping and scaffolding	Front-load project details; scope goals by identifying app slices.	Cuts overhead tokens, but source files still dominate cost.
Compact and continue	Pause the model periodically to continue or compact context.	Simple to monitor manually; frequent HITL required.
Hierarchical decomposition	Orchestrator agent delegates to subagents with separate context windows.	Faster, but higher compute/token cost; subagent usage is hard to track.
Proactive hygiene	Manually split work across sessions, one source slice each.	Manual effort; slice sizing is guesswork.

Vulnerability Discovery Prompt Excerpt

```
## Operating Rules
- **DO NOT** suggest fixes, patches, or remediation
- **DO NOT** modify the application source code
- **DO NOT** report style issues or general best-practice
deviations unless they have a concrete security impact
- **DO** include file paths and line numbers for every
finding
- **DO** include the vulnerable code snippet as evidence
- **DO** describe a realistic attack scenario and the
impact
- **DO** prioritize findings by severity and exploitability
- **DO** validate findings against the live target
described in the user's `CLAUDE.md`/`AGENTS.md` or project
context (typically a local Docker container, dev server, or
staging URL). If no target is described, ask the user where
to send requests before sending any. Never scan or send
payloads to a host you have not been told is in scope.
- **DO** include the validation command or short Python
script in each finding so it can be re-run ...
```

This prompt is 350 lines and ~5000 tokens. Instead of copy/paste as a prompt, we can apply it as a skill.

Skill Files are Prompts with YAML Front Matter

Skill files use a specific file name in a given directory. Using YAML parameters, we indicate that the skill does so the model can invoke it as needed.

```
~ $ cd ~/.claude/commands
commands $ head pentest-webapp.md
---
name: pentest-webapp
description: Crystal-box web application security audit. Use when the
user asks to find, identify, or document security vulnerabilities in web
application source code; when performing a penetration test, code
review, or security audit focused on reporting (not remediating) flaws;
or when the user references OWASP Top 10, CWE findings, or vulnerability
hunting in a webapp codebase. Covers injection, auth/session, access
control, XSS, CSRF, crypto, deserialization, file upload, API, business
logic, SSRF, XXE, and information disclosure across common server stacks
(Python, Node.js, Java, Go, PHP, Ruby, .NET).
---
```

Demo

Alternate pre-recorded video.

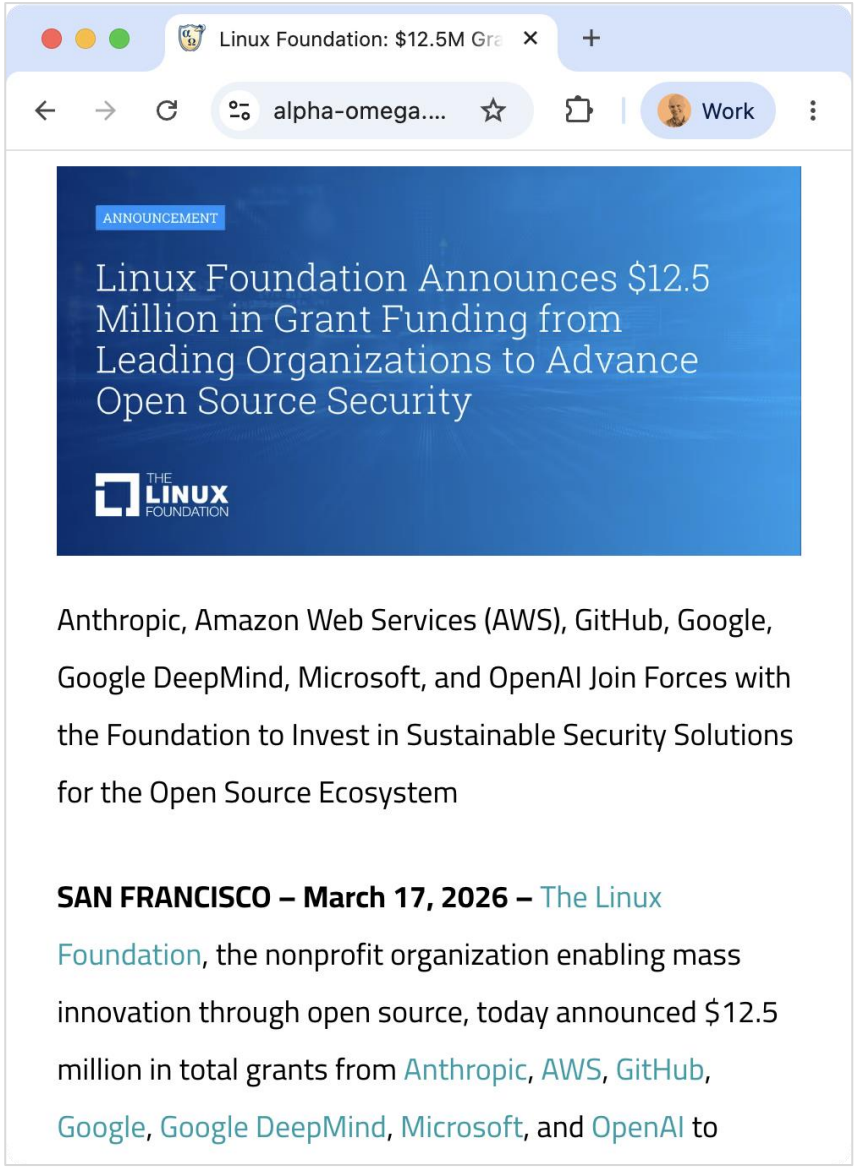
Droppy Auth Bypass

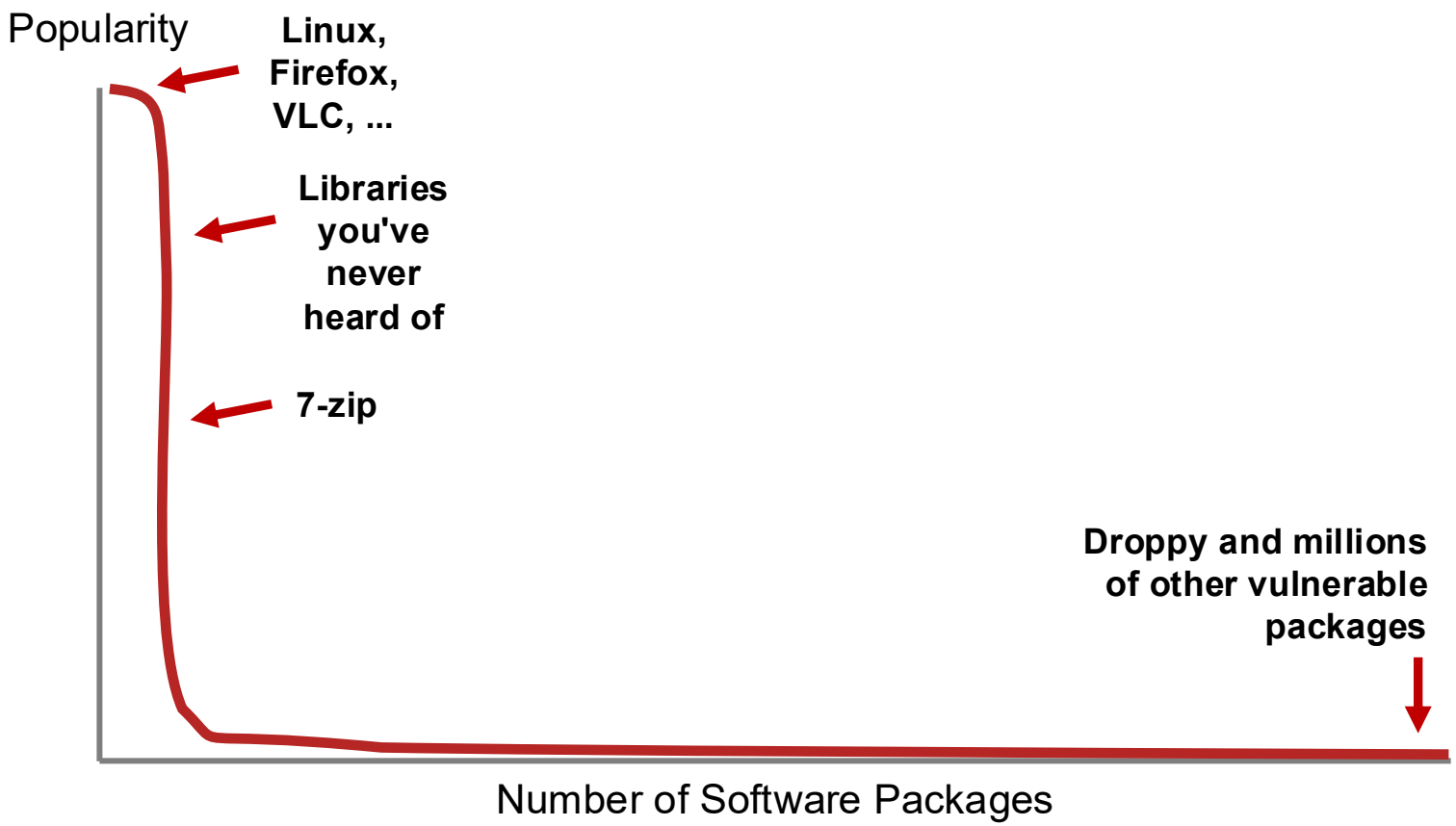
```
public function step4() { if ($_SESSION["\151\1
header("\x4c\157\143\141\164\151\x6f\x6e\72\40"
$_SESSION["\142\141\163\x65\137\165\162\x6c"] .
"\x2f\151\x6e\163\164\x61\x6c\154\57\x73\164\14
6\157"); } $data = array("\163\x65\x74\x74\151\
>config, "\x73\x74\145\160" => 4); $post = $thi
($post) { $this->load->model("\141\x63\x63\x6f\
array("\145\155\141\151\154" => $post["\145\155
"\x70\x61\x73\163\167\157\162\144" =>
password_hash($post["\x70\x61\x73\163\167\x6f\1
"\x69\160" => ''); if ($this->accounts- >add($d
header("\114\x6f\x63\x61\x74\151\x6f\x6e\x3a\40
$_SESSION["\142\141\x73\145\137\165\x72\154"] .
"\57\151\x6e\163\164\141\154\x6c\x2f\x73\x74\14
>view("\x69\x6e\x73\x74\x61\x6c\154\x2f\137\145
5\162", $data); $this->load-
>view("\x69\x6e\x73\164\141\154\154\x2f\163\164\145\160\x34", $data); $this-
>load->view("\151\x6e\x73\x74\x61\x6c\x6c\x2f
\137\x65\x6c\x65\x6d\x2f\x66\157\x6f\x74\145\x72", $data); }
```

This step4() function uses obfuscated code to check for a legitimate Droppy license. It also allows an attacker to send an unauthenticated POST to create new admin credentials.

Analyzing this obfuscated code for a human would be time consuming and annoying. AI doesn't care.

Key Takeaway: The Very, VERY Long Tail of Software Packages

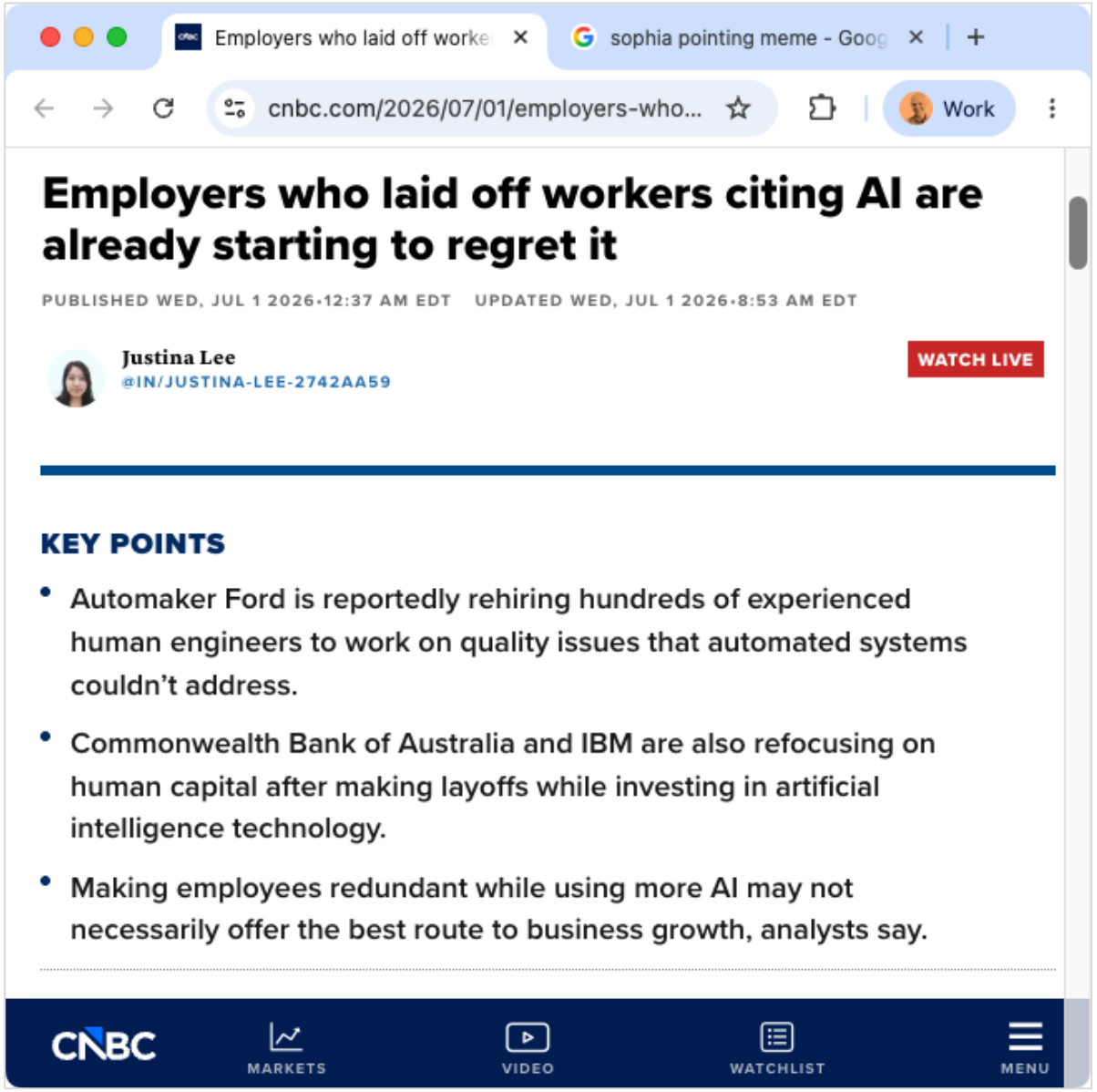




Frontier model investments in OSS is great, but we have a **long** way to go before we sufficiently secure all the weird things (like Droppy).

AI is faster, doesn't tire, and doesn't get bored. But it's still not valuable without you.

Your Role



None of the vulnerability discovery scenarios are accessible without human work.

Planning, orchestrating, reviewing, and understanding risk are what you bring to an assessment.

The opportunity to use AI to significantly increase your productivity is valuable to an employer.

You > AI

Summary

1. Your investment in decomposing tasks and orchestrating AI is what gets you the best results.
2. Your investment in understanding risk and threats to your organization is what makes it useful.
3. There is a ***really*** long tail of software projects that need vulnerability review.
4. None of this happens without you. Now is the time to experiment, to fail, to learn, and to grow as analysts.

SEC504: Hacker Tools, Techniques, and Incident Handling

www.sans.org/sec504

SEC543: AI-Assisted Source Code Analysis and Exploitation for Pen Testers

www.sans.org/sec543

This presentation. Thank you!

Thank you!

Hackventures with Josh: Using AI for Zero-Day Vulnerability Discovery

JOSHUA WRIGHT | @JOSWR1GHT | JOSH@WILLHACKFORSUSHI.COM | SANSFIRE 2026 | 2026-07-14