Mechanizing the Methodology

How to find vulnerabilities while you’re doing other things

Daniel Miessler

Why Automate?

-Hacking is fun, so why automate it away?

- It’s not either-or: you can have both

- Let automation feed your manual work

- *Find bugs while you eat, sleep, game*

Turn Everything Into a Question

- I turn every recon and security consideration into a question

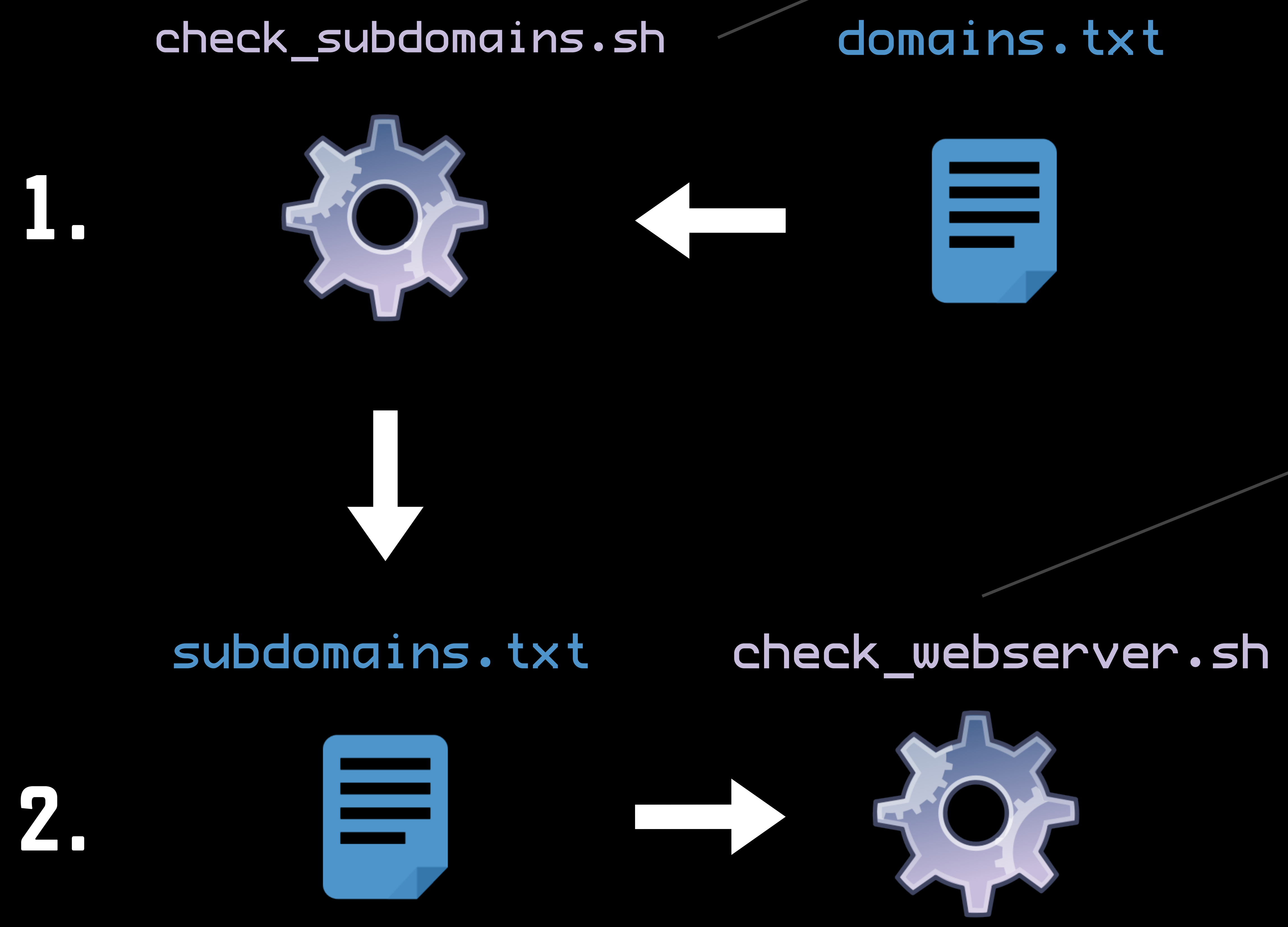
Unix Philosophy

1. Make each program do one thing well.
2. Expect the output of every program to become the input to another, as yet unknown, program.

What are their subdomains?
What ports are open?
Is this ip running a web server?
Has this site changed?
Is this a sensitive site?
What urls are in their js?
Which of these share analytics code?
What domains do they own?
Which certs are about to expire?
What are all the links on this site?
What are this customer's asns?
What ips are in their asns?
Which ips are running web servers?
What stack is this site running?
Which of these sites is running wordpress?
Which of these sites is running drupal?
Who works at this company?
Do they have personal github accounts?
Do those accounts have sensitive content?
Do those accounts have content related to work?
Do they have any s3 buckets that are open?
Are they serving databases?
Are they open or bruteforceable?
…

Input -> Program -> Output = Input -> ...

Unix Philosophy = Simple & Discrete



What are their subdomains?

What ports are open?

Is this ip running a web server?

Has this site changed?

Is this a sensitive site?

What urls are in their js?

Which of these share analytics code?

What domains do they own?

Which certs are about to expire?

What are all the links on this site?

What are this target's asns?

What ips are in their asns?

Which ips are running web servers?

What stack is this site running?

Which of these sites is running wordpress?

Which of these sites is running drupal?

Who works at this company?

Do they have personal github accounts?

Do those accounts have sensitive content?

Do those accounts have content related to work?

Do they have any s3 buckets that are open?

Are they serving databases?

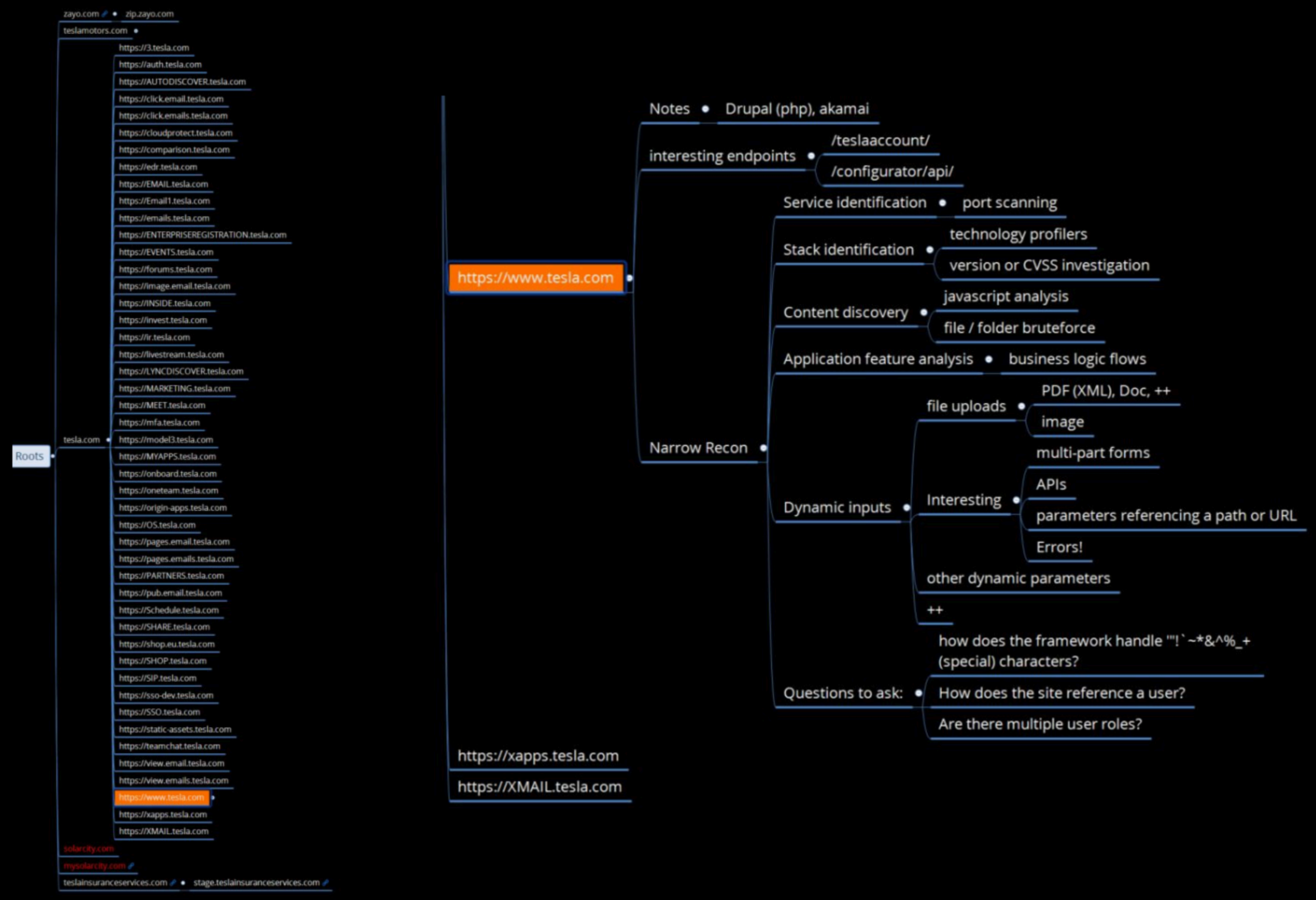
Are they open or bruteforceable?

...

Use a Methodology to Build Your Questions

Jason Haddix
@jhaddix

The Bug Hunter's Methodology



- Roots
  - zayo.com
    - zip.zayo.com
  - teslamotors.com
  - tesla.com
    - https://3.tesla.com
    - https://auth.tesla.com
    - https://AUTODISCOVER.tesla.com
    - https://click.email.tesla.com
    - https://click.emails.tesla.com
    - https://cloudprotect.tesla.com
    - https://comparison.tesla.com
    - https://edr.tesla.com
    - https://EMAIL.tesla.com
    - https://Email1.tesla.com
    - https://emails.tesla.com
    - https://ENTERPRISEREGISTRATION.tesla.com
    - https://EVENTS.tesla.com
    - https://forums.tesla.com
    - https://image.email.tesla.com
    - https://INSIDE.tesla.com
    - https://invest.tesla.com
    - https://ir.tesla.com
    - https://livestream.tesla.com
    - https://LYNCDISCOVER.tesla.com
    - https://MARKETING.tesla.com
    - https://MEET.tesla.com
    - https://mfa.tesla.com
    - https://model3.tesla.com
    - https://MYAPPS.tesla.com
    - https://onboard.tesla.com
    - https://oneteam.tesla.com
    - https://origin-apps.tesla.com
    - https://OS.tesla.com
    - https://pages.email.tesla.com
    - https://pages.emails.tesla.com
    - https://PARTNERS.tesla.com
    - https://pub.email.tesla.com
    - https://Schedule.tesla.com
    - https://SHARE.tesla.com
    - https://shop.eu.tesla.com
    - https://SHOP.tesla.com
    - https://SIP.tesla.com
    - https://sso-dev.tesla.com
    - https://SSO.tesla.com
    - https://static-assets.tesla.com
    - https://teamchat.tesla.com
    - https://view.email.tesla.com
    - https://view.emails.tesla.com
    - https://www.tesla.com
    - https://xapps.tesla.com
    - https://XMAIL.tesla.com
  - solarcity.com
  - mysolarcity.com
  - teslainsuranceservices.com
    - stage.teslainsuranceservices.com

- https://www.tesla.com
  - Notes
    - Drupal (php), akamai
  - interesting endpoints
    - /teslaaccount/
    - /configurator/api/
  - Narrow Recon
    - Service identification
      - port scanning
    - Stack identification
      - technology profilers
      - version or CVSS investigation
    - Content discovery
      - javascript analysis
      - file / folder bruteforce
    - Application feature analysis
      - business logic flows
    - Dynamic inputs
      - file uploads
        - PDF (XML), Doc, ++
        - image
      - Interesting
        - multi-part forms
        - APIs
        - parameters referencing a path or URL
        - Errors!
      - other dynamic parameters
      - ++
    - Questions to ask:
      - how does the framework handle "'!`~*&^%_+ (special) characters?
      - How does the site reference a user?
      - Are there multiple user roles?
- https://xapps.tesla.com
- https://XMAIL.tesla.com

What are their subdomains?

What ports are open?

Is this IP running a web server?

Has this site changed?

Is this a sensitive site?

What URLs are in their JS?

Which of these share analytics code?

What domains do they own?

Which certs are about to expire?

What are all the links on this site?

What are this target's ASNs?

What IPs are in their ASNs?

Which IPs are running web servers?

What stack is this site running?

Which of these sites is running WordPress?

Which of these sites is running Drupal?

Who works at this company?

Do they have personal GitHub accounts?

Do those accounts have sensitive content?

Do those accounts have content related to work?

Do they have any S3 buckets that are open?

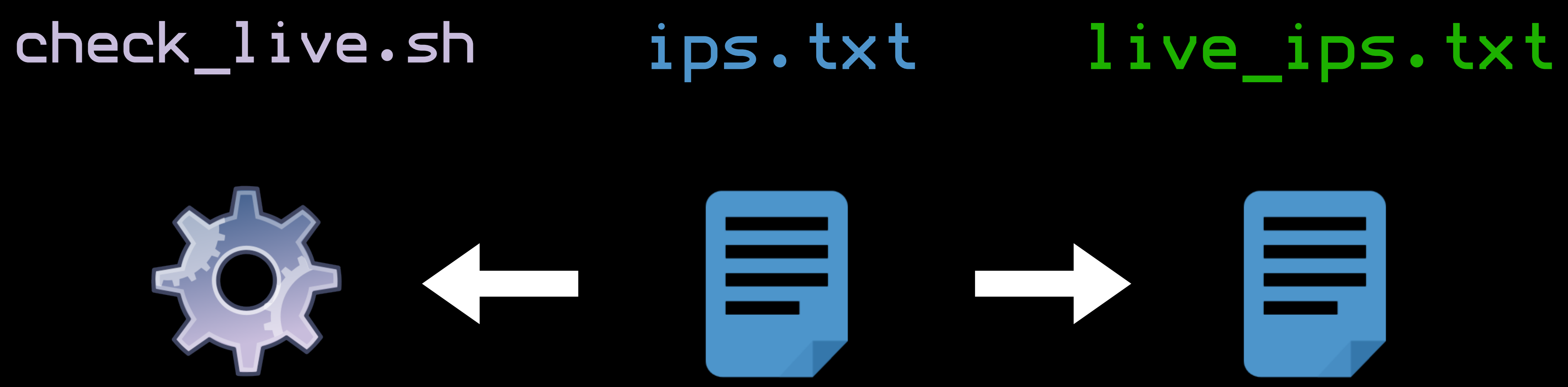
Are they serving databases?

Are they open or bruteforceable?

...

Methodology | Find Live Hosts

Q: For a given IP range, what hosts are alive?



- There are many ways to scan ports
- I use **masscan** for speed, and **nmap** for follow-up and NSE
- This snippet will return any host that is listening on any of Nmap's top 100 ports

// the Nmap equivalent of —top-ports 100

```
masscan --rate 100000
-p7,9,13,21-23,25-26,37,53,79-81,88,10
6,110-111,113,119,135,139,143-144,179,
199,389,427,443-445,465,513-515,543-54
4,548,554,587,631,646,873,990,993,995,
1025-1029,1110,1433,1720,1723,1755,190
0,2000-2001,2049,2121,2717,3000,3128,3
306,3389,3986,4899,5000,5009,5051,5060
,5101,5190,5357,5432,5631,5666,5800,59
00,6000-6001,6646,7070,8000,8008-8009,
8080-8081,8443,8888,9100,9999-10000,32
768,49152-49157 -iL ips.txt | awk
'{ print $6 }' | sort -u >
live_ips.txt
```

// the file's output is naked IP addresses ready to become input

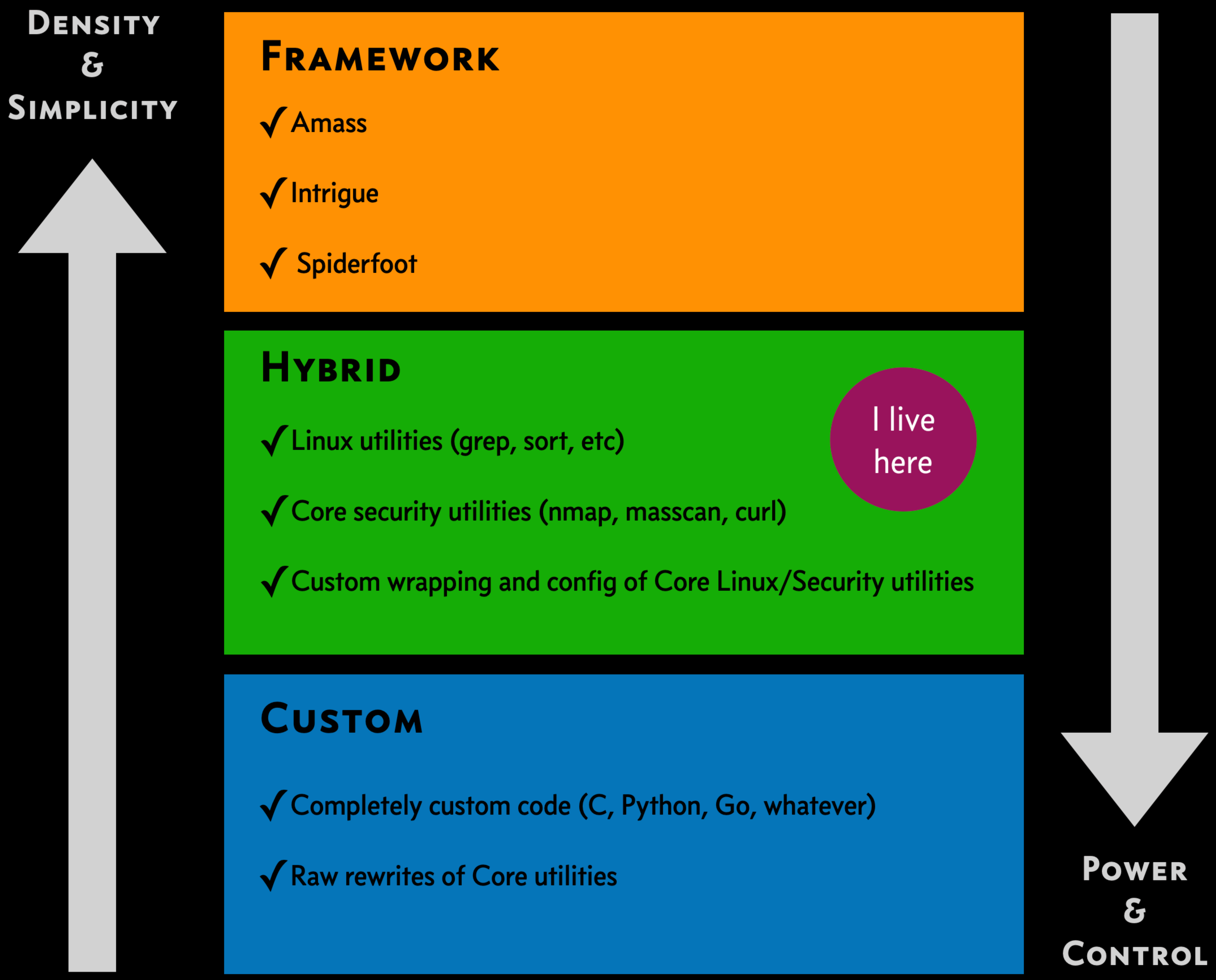
```
vi live_ips.txt
```

```
1.2.3.4
1.2.3.5
1.2.3.6
1.2.3.7
```

A Brief Word on Module Philosophy

What about frameworks?

- There are some amazing frameworks out there
- There are two extremes: writing your own low-level code, and using high-level frameworks
- There are tradeoffs with each approach
- Frameworks save you tons of work and combine multiple steps
- But frameworks also abstract steps away from you so you can't easily see how they're being accomplished



Density & Simplicity

Framework

- ✓ Amass
- ✓ Intrigue
- ✓ Spiderfoot

Hybrid

- ✓ Linux utilities (grep, sort, etc)
- ✓ Core security utilities (nmap, masscan, curl)
- ✓ Custom wrapping and config of Core Linux/Security utilities

I live here

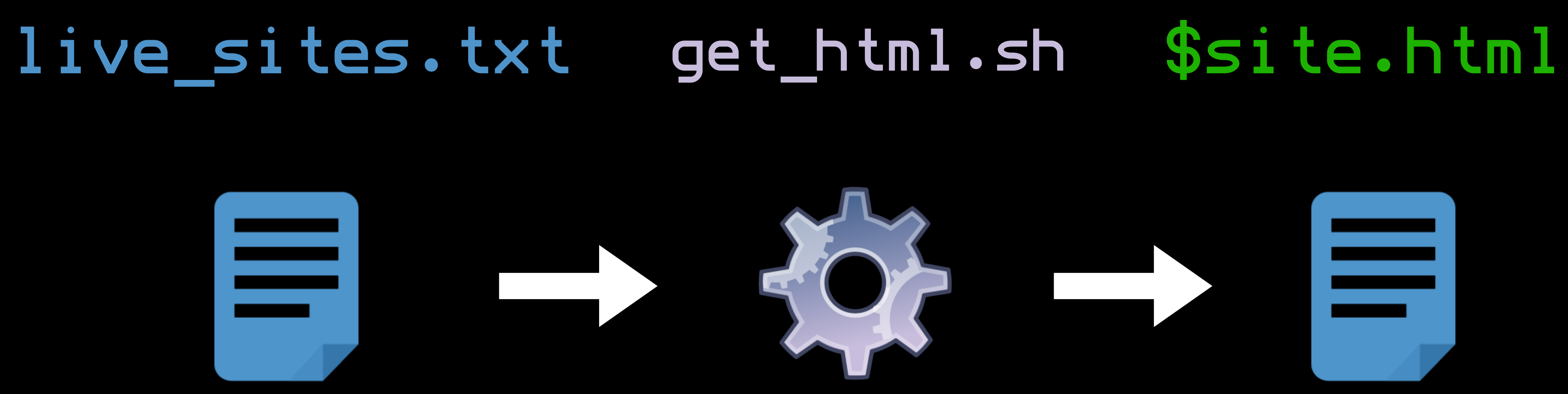
Custom

- ✓ Completely custom code (C, Python, Go, whatever)
- ✓ Raw rewrites of Core utilities

Power & Control

Methodology | Get a Page's HTML

Q: What is the full HTML for a given page?



- A page's full HTML is a fundamental seed for many other modules

- From there you can find potentially sensitive fields, parse links, pull out JavaScript files, etc.

- The trick is to do this as authentically as possible, hence Chromium vs. Curl

// Chromium is far better than Curl at pulling real-world content

```
chromium-browser —headless --user-
agent='Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537. 36
(KHTML, like Gecko) Chrome/
67.0.3396.99 Safari/537.36' --dump-dom
$site > $site.html
```

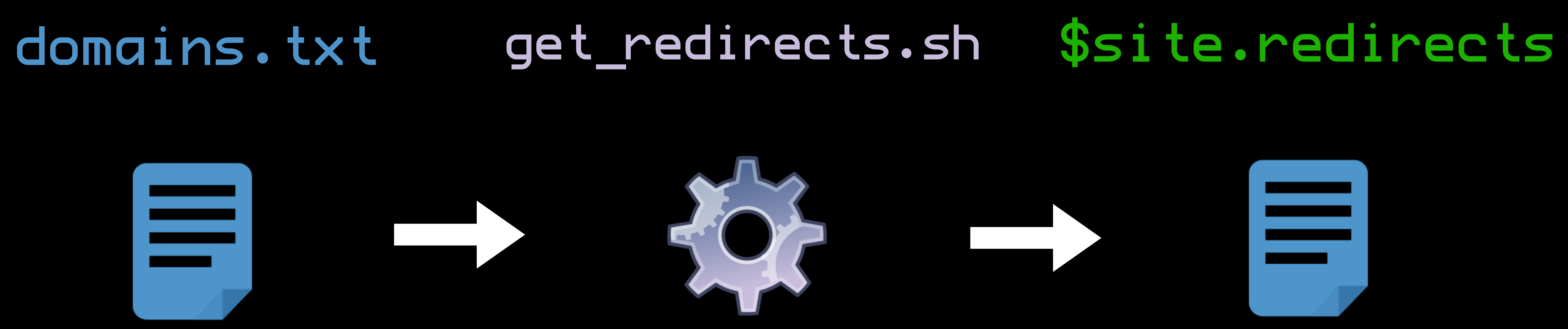
// You then have full HTML to parse and inspect

```
vi $site.html
```

```
<!DOCTYPE html>
<html lang="en-US"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-
width, initial-scale=1"> <style media="all">@font-face{font-family:'concourse-t3';src:url(//
danielmiessler.com/wp-content/themes/danielmiessler/fonts/concourse_t3_regular-webfont.woff)
format('woff');font-style:normal;font-weight:400;font-stretch:normal;font-display:fallback}
@font-face{font-family:'equity-text-b-caps';src:url(//danielmiessler.com/wp-content/themes/
danielmiessler/fonts/equity_caps_a_tab_regular-webfont.woff) format('woff');font-
style:normal;font-weight:400;font-stretch:normal;font-display:fallback}@font-face{font-
family:'equity-text-b';src:url(//danielmiessler.com/wp-content/themes/danielmiessler/fonts/
equity_text_b_regular-webfont.woff) format('woff');font-style:normal;font-weight:400;font-
stretch:normal;font-display:fallback}@font-face{font-family:'equity-text-b';src:url(//
danielmiessler.com/wp-content/themes/danielmiessler/fonts/equity_text_b_bold-webfont.woff)
format(‘woff');font-style:normal;font-weight:700;font- …
```

Methodology | Getting the Domains That Redirect to a Domain

Q: What domains redirect to my domain?



- One of the biggest tasks is getting the total, top-level scope for a given company

- This requires you to pivot from known TLDs to other TLDs in various ways

- Following redirects to a trusted domain is one technique to add to your toolchain

// I heavily rely on `ipinfo.io` and `host.io`

```
curl -s “https://host.io/api/
domains/redirects/$site?
&limit=1000" | jq -r '.domains' |
jq '.[]' | tr -d \" >
$site.redirects
```

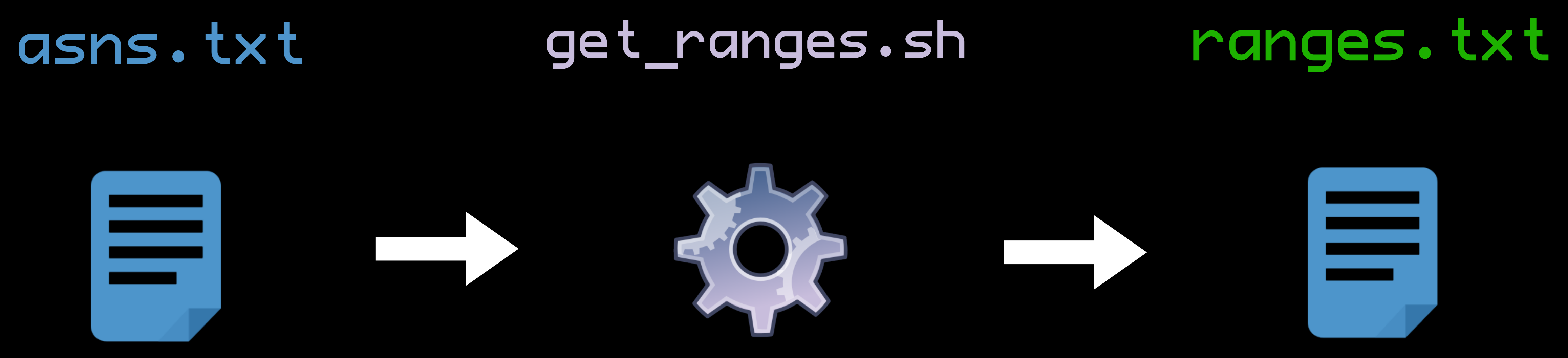
// Now you have redirects to further scrutinize to see if they belong to that company.

```
head -5 tesla.com.redirects

teslasmail.com
telsamotors-losangeles.com
teslamotorsantartica.com
telsa-newyork.com
mevg.info
```

Methodology | Getting Ranges for an ASN

Q: What IP ranges are associated with these ASNs?



- Many tools can get you IP ranges from ASNs
- I prefer to query ipinfo.io directly as part of my automation
- The trick is to fully understand your sources and develop trust in them
- You never want to wonder how they get their data, or if they might be stale or broken

// Pulling ranges for one of the ASNs for Tesla

```
curl -s "https://ipinfo.io/AS394161/json/" | jq '.prefixes[].netblock' | tr -d \" > ranges.txt
```

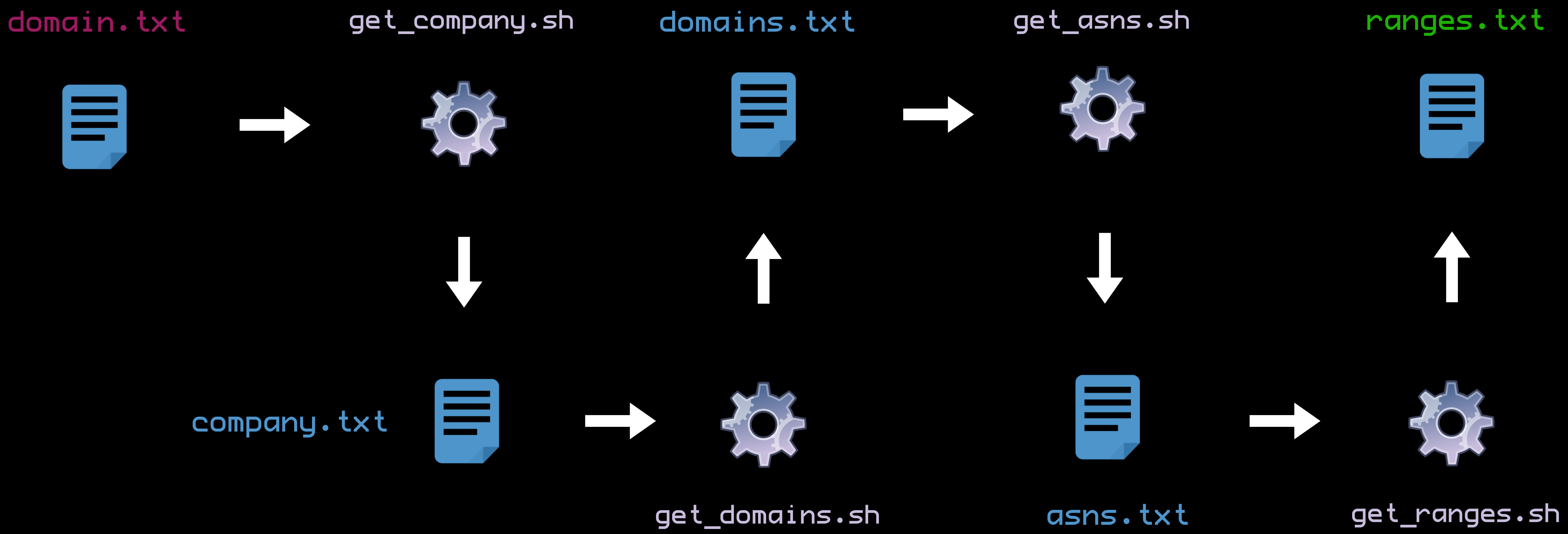
// Now you have ranges to pass into testing modules

```
head -5 ranges.txt
```

```
192.95.64.0/24
199.120.48.0/24
199.120.49.0/24
199.120.50.0/24
199.66.10.0/24
```

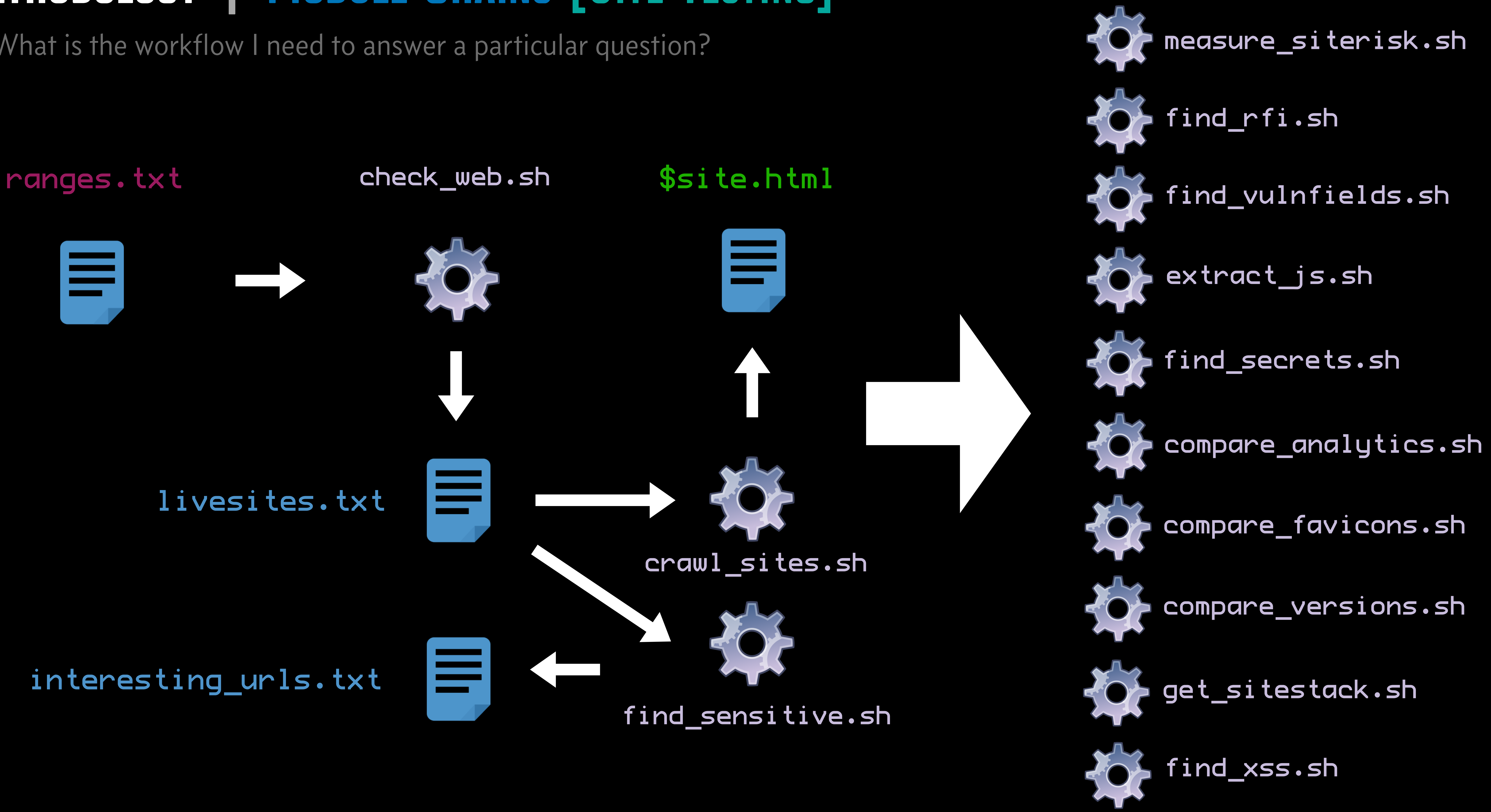
Methodology | Module Chains [Ranges]

Q: What is the workflow I need to answer a particular question?



METHODOLOGY | MODULE CHAINS [SITE TESTING]

Q: What is the workflow I need to answer a particular question?



ranges.txt

check_web.sh

$site.html

livesites.txt

crawl_sites.sh

interesting_urls.txt

find_sensitive.sh

measure_siterisk.sh

find_rfi.sh

find_vulnfields.sh

extract_js.sh

find_secrets.sh

compare_analytics.sh

compare_favicons.sh

compare_versions.sh

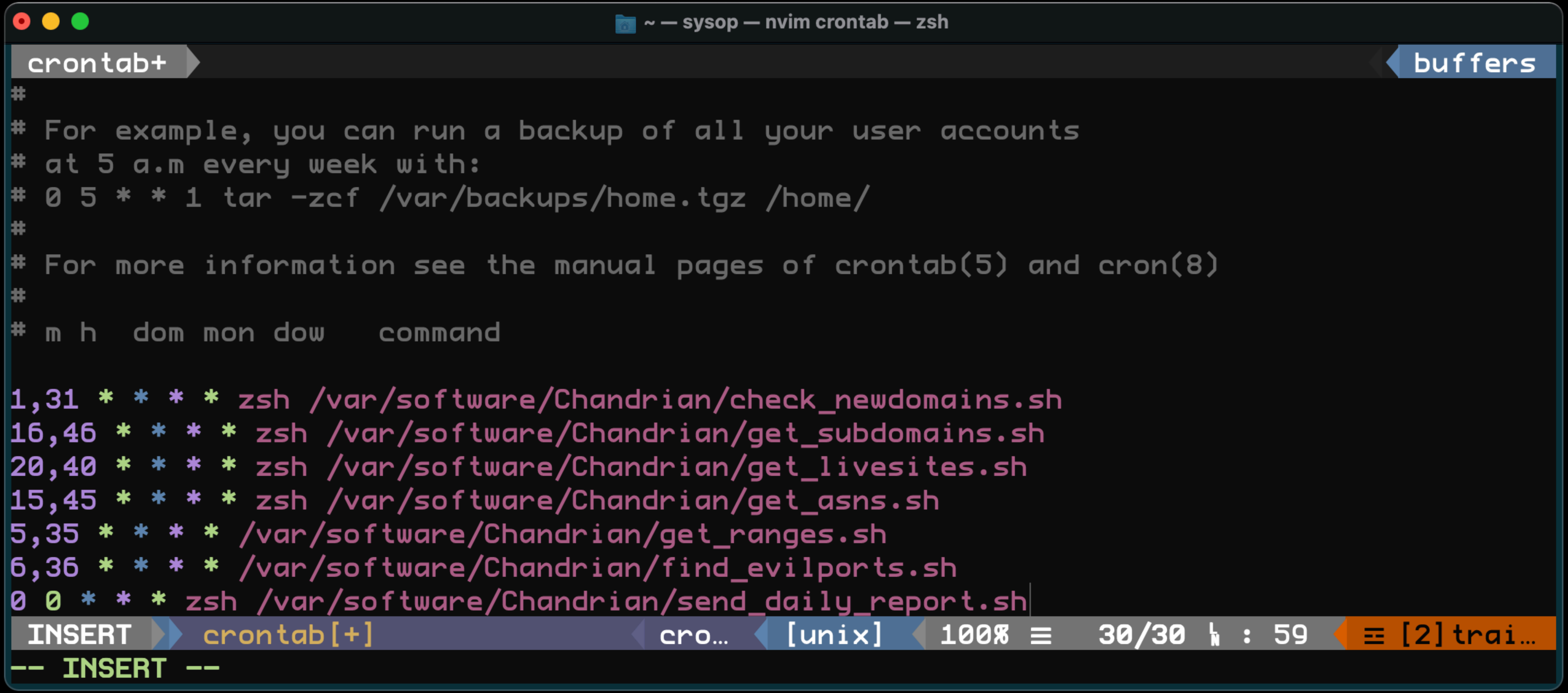
get_sitestack.sh

find_xss.sh

Automation | Linux [Cron]

Forget scanning or checking, let's *monitor* instead

- If only Linux had a way to schedule things to happen continuously…oh, it does
- Cron can make sure as many modules as possible are run continuously, at any schedule you need
- With proper checking in the modules and some care in the scheduling, you can make sure the output for one is ready before it's needed for the next



```
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

1,31 * * * * zsh /var/software/Chandrian/check_newdomains.sh
16,46 * * * * zsh /var/software/Chandrian/get_subdomains.sh
20,40 * * * * zsh /var/software/Chandrian/get_livesites.sh
15,45 * * * * zsh /var/software/Chandrian/get_asns.sh
5,35 * * * * /var/software/Chandrian/get_ranges.sh
6,36 * * * * /var/software/Chandrian/find_evilports.sh
0 0 * * * zsh /var/software/Chandrian/send_daily_report.sh
 INSERT  crontab[+]                cro…  [unix]  100% ≡  30/30 ㏑ : 59  ☰ [2] trai…
-- INSERT --
```

Notification | Amazon SES, Slack, Whatever

You can do anything from the command line, including sending emails, sending to Slack, etc.

- One major benefit to continuous monitoring is continuous alerting
- Using email, Slack, or other types of API-based notification, you can know as soon as your workflow finds something
- I really like Amazon SES for sending emails, and Slack for something richer
- The code for sending notifications via the command line is dead-simple—basically a one-liner

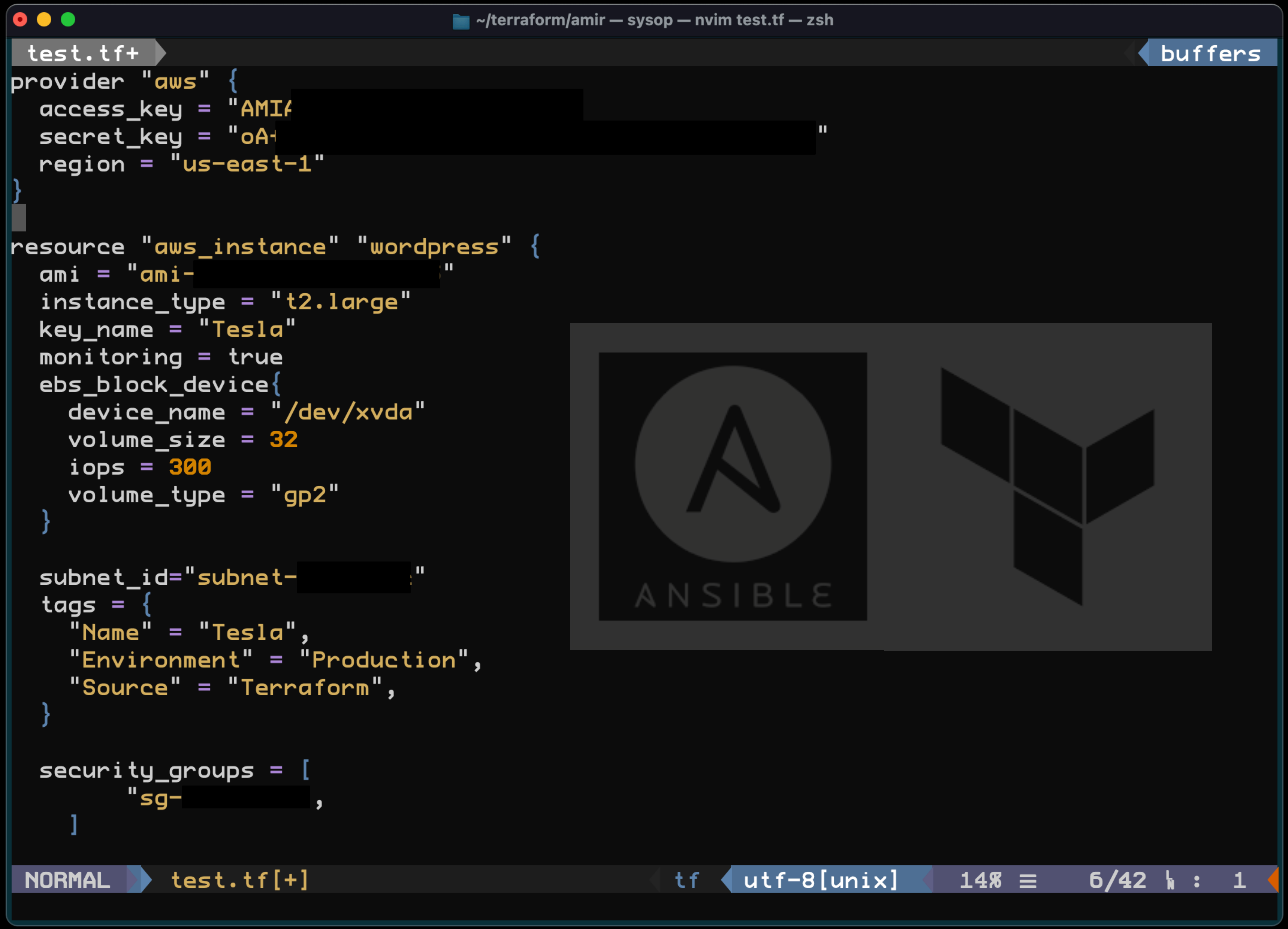
```
ssmtp "$RECIPIENT" < domain.notification
```

```
curl X POST -H 'Content-type:
application/json' --data ‘{“text”:”Hey,
there’s a new yummy (open) PostgresDB @
1.2.3.4”}’ YOUR_WEBHOOK_URL
```

Deployment | Build & Configuration

Use modern tools to automate your entire build and deploy

- It's hard to maintain all this code and configuration as one-offs
- I highly recommend moving to a config management technology like Terraform, Ansible, Git, or Axiom—by Ben Bidmead
- Axiom by @pry0cc (Ben Bidmead) is a sick way to deploy a Linux stack to Digital Ocean
- I personally use Terraform & Ansible to deploy boxes to AWS
- I make all changes locally, do a quick `terraform apply` and my monitoring and alerting goes live on the internet



```
provider "aws" {
  access_key = "AMI[redacted]
  secret_key = "oA[redacted]"
  region = "us-east-1"
}

resource "aws_instance" "wordpress" {
  ami = "ami-[redacted]"
  instance_type = "t2.large"
  key_name = "Tesla"
  monitoring = true
  ebs_block_device{
    device_name = "/dev/xvda"
    volume_size = 32
    iops = 300
    volume_type = "gp2"
  }

  subnet_id="subnet-[redacted]"
  tags = {
    "Name" = "Tesla",
    "Environment" = "Production",
    "Source" = "Terraform",
  }

  security_groups = [
      "sg-[redacted],
  ]
```

```
NORMAL   test.tf[+]                         tf   utf-8[unix]   14% ≡   6/42 ㏑ :  1
```

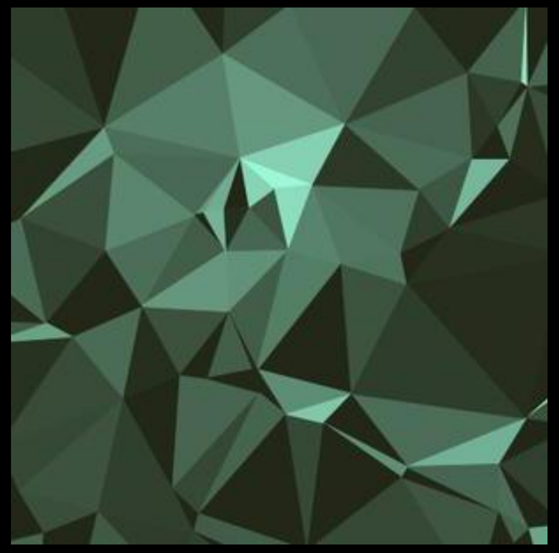
Continuous Improvement | Learn, Adjust, Repeat [10 People to Follow]

Once you have automation in place, you can improve it by adding new techniques

Jason Haddix
@jhaddix

Ben Sadeghipour
@nahamsec

Tom Hudson
@tomnomnom



Michael Skelton
@codingo

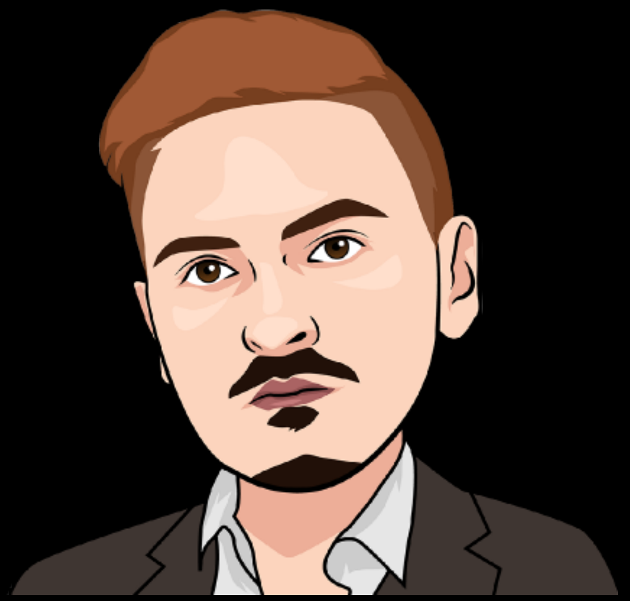
Luke Stephens
@hakluke

Stök
@stokfredrik

Jeff Foley
@caffix

Naffy
@nnwakelam

Ben Bidmead
@pry0cc



The Cyber Mentor
@thecybermentor

Takeaways | How to Run an Automated Testing Platform on a $10 Linux Box

Think of every technique as a distinct tool with one function

1. Break your techniques into questions
2. Create a separate UNIXY module for each step
3. Create intuitive output artifacts that can be used as inputs to other modules
4. Chain those modules according to a methodology that resonates with you
5. Continuously run those modules using Cron
6. Use Amazon SES or Slack for alerting
7. Wire up your full config using Terraform/Ansible/Axiom for easy deployment
8. Follow the best testers/creators in the industry to stay up on new techniques
9. Come back and hack manually on what your automation finds
10. Profit (in relaxation, time, money, or all of the above)

Mechanizing the Methodology

How to find vulnerabilities while you're doing other things

DANIELMIESSLER.COM

DANIELMIESSLER.COM/PODCAST

TWITTER.COM/DANIELMIESSLER

GITHUB.COM/DANIELMIESSLER

Thank you!