The Top Hacker Methodologies & Tools Notes

By Chase. ** *

Nuclei Templates

Concrete5 CMS : Identification, Mass Hunting, Nuclei Template Writing & Reporting

- My Methodology
- My Methodology Roadmap
- Jason Haddix
- Nehamsec
- Tomnomnom
- InsiderPHD
- Mayonnaise
- ZSeano
- Hacktify
- Dawgyg
- Hakluke
- XSS Rat
- Hackerish
- Jeff Foley

Tools/Tutorials/Other

- Noisy-Hacker Automated XSS
- Amass with Hacking Simplified
- Bug bounty recon Automation using bash | Automate your complete recon | @CyberUF
- The Mass Hunt XSS Technique Bug Bounty Hunters don't want you to know
- Mass Hunt XSS with and without GF Patterns
- Bugbounty Webinar - Recon,Live attacking, Tools by Secoceans | #@secoceans
    - Secoceans Introduction
- BHIS | Hack for Show, Report For Dough: Part 2 w/ BB King (1-Hour)
- https://medium.com/@coffeeaddict_exe/my-recon-process-command-line-1c9f603f4e9f

- [Deepak Dhiman to Virdoex Hunter](#Deepak-Dhiman-to-Virdoex-Hunter

https://youtu.be/-A0S2AkCLFg?t=1158

He likes finding thin client RCEs and examining buffer overflows on binaries using his <IDR??> Tool, mostly because hardly any hackers are looking in this area https://youtu.be/-A0S2AkCLFg?t=1158

This guy covers a lot of pretty basic things and appears to try and appeal to and please a driven yet younger/inexperienced crowd, but he has these moments where he just 'pops off' and starts spitting some fire :fire: knowledge and information :book:. Those moments along wi his positive attitude and postive really makes him appealing and oooov

He explains CVE vs CWE really well here

jeff foley amass https://youtu.be/H1wdBgY1rtg?t=3472

My Tools

Amass Duplicut FFuf Burp Hetty HttpToolkit Zap Nmap Postman Altair Wafw00f Katana DeveloperTools Cent GetJS Nuclei Jshole Subjs https://www.ipvoid.com/udp-port-scan/ https://dnsdumpster.com/ Aline - https://github.com/ferreiraklet/Aline/issues/2#issuecomment-1214206037 Use proxys to rotate ip addresses as needed Resources / Credit https://owasp.org/www-pdf-archive/OWASP_Cheatsheets_Book.pdf https://github.com/bbhunty/bbtips https://github.com/KingOfBugbounty/KingOfBugBountyTips

My Methodology (WIP ... OBVIOUSLY!)

Chase Jensen Methodology Roadmap

- Check for /login /log_in /sign_up /signup /api /users /admin /admin_panel /api/v1 /api/v2 /API /soap /robots.txt /sitemap.xml /wpscan
- Check for what the 404 page looks like and what error pageg loo like /.404 /404.html /404.php /%00%01
- Check for common payloads
- Check for POST /login /logout /log_in /log_out /users/new /users/1
- Check for site.example.com/site and site.example.com/example

#My Automated Methodology (WIP)

1. After Getting subdomains, run technology scan (whatweb? Nuclei? Other?) and if it detects certain stacks, (wordpress, graphql) then automatially recon those and alert discord or telegram

etc. Also, js vulns alert. Basically automate the checking

Hackers -

TODO https://youtu.be/uKWu6yhnhbQ?t=240 his methodology template

#NahamCon2022 - Jason Haddix (@jhaddix): The Bug Hunter's Methodology: Application Analysis v1

[#NahamCon2022 - Jason Haddix (@jhaddix): The Bug Hunter's Methodology: Application Analysis v1]{www.youtube.com/watch?v=HmDY7w8AbR4}

Creating Wordlists from historical based urls

https://youtu.be/HmDY7w8AbR4?t=4672 Hunt https://github.com/bugcrowd/HUNT fpr prioritizing parameters [1:19:50](https://youtu.be/HmDY7w8AbR4?t=4780] Keep an eye out for jhaddix /suss on github

Summary Mind Map For his analysis of www.tesla.com Content Discovery

1. Tech Browser Extensions
2. Pre-manual Testing and Automation - Scanners (DID NOT RUN) Nuclei, Jaeles, Portscan Naabu (Standard ports OPEN)
3. Content Discovery - Wordlist - Seclists-Discovery/URLS/urls-drupal-7-20.txt Wordlist2: Raft, HTTParchive_directories, onelistforall, content_discovery_all
4. Gau | Trashcompactor wordlistgen
5. Parsed Mobile Questions
6. How does the app pass data? Hybrid: Params and API
7. How/Where does the app talk about users? It didn't directly have a UID on tesla.com but tightlyh integrated into tesla.com was shop.tesla.com which had this endpoint (He searched burp history for uuid and uid and found) http://shop.tesla.com/user-status.json

```
{
  "signed_in": "true",
  "emailId": "jhaddix@gmaiol.com"
}
```

8. Does the site have multi-tenancy or user levels? Framework - Yes - Drupal should have admin role API - Yes, api should be tested for access and auth as it could return other users data
9. Does it have a unique threat model? Not for Tesla.com but possibly through mobile (does things with the car, can control the roof etc)

10. How does the site handle XSS? output is encoded per drupal rules on telsa.com still need to check api
11. How does the site handle CSRF? Token in Header, applied everywhere so far. Need to test API and Shop
12. Has the site had any past writeups / exploits? 2014 sqli twitquark.com/so/tesla-sqli-2014 (make sure ot include links) 2022 DOM XSS bugcrowd.com/disclosures/xss/3232332 2021 uploaded creds trt.teslamotors/itscript/pdx01.zip

Heatmap - Profile Sections

41:57 Heat Mapping

- Heat Mapping Mind Map He keeps this up on his desktop to remind him about where he wants to look and what he wants to look at
- 6 areas he looks
    - File Uploads very first - (usually vulnerable to injection, cross site scripting, xxe, ssrf, trick it to shell a server depending on the tech stack / parser, and determine where / how the data is stored for the application.
    - Content Types
        - He filters burp looking by all the returned content looking for 'multipart forms'. He has it outlined in RED because he has **never met a multipart form that was secure**. Bet. Whether you're shelling it, doing an injection
        - He looks for Content-Type XML for XXE
        - He looks for Content-Type JSON for api vulns
    - APIs
        - Hidden Methods / endpoints
        - Lack of auth
    - Profile
        - stored xss
        - APp custom fields
    - Integrations
        - SSRF, XXE
- Errors -exotic injection So he can (Smuggle Payloads in through third parties into the app) [https://youtu.be/FqnSAa2KmBI?t=2656]
    - App DOS
    - Versions? (my addition) -Paths or URLS passed as values
    - SSRF
    - REDIRS

45:34 He uses Notion to track these, here is an example.

1:06:36 xnlinkfinder javascript deminifyijg [1;11;29 https://youtu.be/HmDY7w8AbR4?t=4286
Methodology 1

1. Crunchbase - get acquisitions. 2- Get a ASN for thr company. He uses Hurricane Electric

Acquisitions

Metabigor 2-

2. Nehamsec

https://github.com/fuzz-security?tab=repositories

He sets up his ubuntu box using his BBHT Repo

> He runs Directory Brute forcing because everytime he finds a new host, automatically he runs directory brute forcing on it, just to make sure he finds all the juicy files and folders. He uses Dirsearch as shown here

```
python3 dirsearch.py -u www.hackerone.com -t 50 -e html
```

his dog coco! https://youtu.be/hdp-WDZ9dgw?t=5327

```
# /bin/crtsh
# Also found https://www.udemy.com/course/intro-to-bug-bounty-by-nahamsec/learn/lecture/249981

# Manual

# Automated
curl -s https://crt.sh/\?q\=%25.$1\&output=json | jq -r  '.[].name_value' | sed 's/\*\.//g' |
```

3. Tomnomnom

Tomnomnom Methodology

Live Recon and Automation on Shopify's Bug Bounty Program with @TomNomNom

Assetfinder & wildcards file

He gathers the wildcard from their scope into `wildcards` then runs `assetfinder` then he goes and finds subdomains. `cat wildcards | assetfinnder --subs-only | anew domains`

> If your subdomains end up containing characters at the front of them like `-cisco.shopify.com`, a nice low tech way to clean them is to `sort` (: `sort` in Vim) and then pop to the end (`G` in Vim) so any special characters will go down to the end (I suppose a reverse sort would make them at the beginning) making it easy to trim them out and get rid of any lines beginnig with a `*` and so on. He keeps the ones starting with `_` like `_dev.shopify.com`.

Httprobe --prefer-https & hosts file

To find what's listening he uses httprobe with a concurrency of 80 'or something like that' Final command here `cat domains | httprobe -c 80 --prefer-https | anew hosts` he does that to see what respoods on port 443 or 80. He wrote into Httprobe the ability to prefer-https with the `--prefer-https` flasg.

Anew

Previously before anew he would use `tee -a`. A common use case him is he has a file of domains and he didn't want to have to de-duplicate them all the time, so he created `anew`. He uses a few different tools to look for domains e.g. `assetfider`, `findomain`. `anew` only adds nenw lines to the file, it works works similar to `tee -a` except if a line already exists in a file, it doesn't add it. It also outputws what the new lines were so you know what it is adding to the file if thats where stdout is putting the output.x

> Starts here

`findomain -f wildcards | tee -a findomain.out` and then `cat findomain.out | anew domains | httprobe -c 50 | anew hosts`

If he was to Httprobe the output of `findomain` before ran anew on them against the domains file which ensures only non duplicate hosts will be displayed as output, then it would run httprobe for ALL of them raher on unique domains. A cool tip to keep the noise low. This highlights a problem with anew / this workflow in my opinion, if something happened (a typo in the command, commad recieved a error code response it can't handle, etc) then httprobe might not get ran and anew would cover up the problem on accident.

Another Tomnomnom video / tutorial: TODO

Move from one working thing to another working thing. Make small modification https://github.com/BlackFan/client-side-prototype-pollution

2. InsiderPHD

Methodology Notes #1 - Katie Explains - My Methodlogy

> She clicks all the buttons always. She prefers to understand every tool and have a precise reason and goal for usage. Did I mention Click ALL the buttons? Click all the buttons. Make sure you've tried good ol' press all the buttons.

API Enumeration / Burp Intruder

Copy & Paste into Burp Intruder

Identify Interesting Endpoints - Learning to recognize interesting endpoits takes lots of time as a developer or lots of time hackinng. Rely on your intuition

> Things to look for - signs of interesting endpoints
>
> - ID = IDOR
> - Sequential? (ids?)
> - Reflected Input = XSS
> - Complex Processes = Busiess Logic
> - Lots of Data = Information Exposure

Wordlists Public / API / Custom wordlists

API Wordlists - A wordlist of API Names used for fuzzing web application APIs. When do we use wordlists? 2 main times in API recon specifically

> When we want to enumerate resources to find new endpoints When we want to enumerate paramters to find additional functionarlity Often this require 2 different Wordlists!

1. Public wordlists

1. Seclists (API Specific)
2. Fuzzdb (API Specific)
3. Seclists raft words (list of words)

2. Manual Wordlists

- Similar services might have documented APIS - check for those
- Or use the waybackmachine method on those API's (nehamsecs videos)
- Make sure you've tried good old press all the buttons Then start writing sensible words

- What does the app say?
- What actions does it let you do?

```
/api/v1/profile
/api/v1/users
/api/v1/posts
/api/v1/answers
```

DEMO: Enumeration API Endpoints using ffuf and Burp Intruder Scenario: Starting on a new webpage, we don't see much other than a login page. There is not much content to click on. Staring at you is a login. Finally, you check burp. You see `/api/users/6` in a BURP web request. Let the fun begin!

1. In burpsuite send an API request you want to fuzz to Intruder.
2. Remove the existing API function call and replace it with the two SS Characters for each text file you want to use
3. On the 'Positions' tab, set Attack type to 'Cluster Bomb'
4. On the Payloads tab, select 1 for the first payload set drop-dow then select a payload type of "Runtime file" and navigate to the driectory you downloaded these text files to, select "Actions.txt"

3. Mayonnaise

4. Jason Haddix

(00) Project Tracking

Take notes with Xmind. He finds Acquisitionos with Crunchbase. The he finds ASN numbers for all acquisitions found manually with bgp.he.net

(01) Finding Seeds/Roots

You can use [asnlookup] or [metabigor]

> ASN - The Autonomous System Number is a reference to their entire IP Space. A 'Reference Identifier' for all of their Registered Ips/IP Space. Collecting ASNs maually can be important so you don't accidently enumerate over an asset that is NOT in scope, or more in particular of a wrong target altogether.

> Take for instance he searched for "Office" and it came up with Office Depot Europe B.V and Office Depot (as it should.) It also came up with "The Office of fhe President". Which... you probably don't want to be enumerating willy Nilly!

Amass - Run amass on the found ASNs `amass intel -asn 138603 -config $HOME/.config/amass/config.ini`

Whoxy - While Amass is running he runs the company through whoxy. You can get a free api key through Whoxy. There are many Reverse "Who Is" databases find the one you like.

Domlink - give it your api key and your domain and it will find everything for the 'company' and 'email' and all the related domains

He uses the search engine tool `Shodan` to glean a valuable question: is `twitch.amazon.eu` (Found prevously using _____) relevant to our testing? Example: `https://www.shodan.io/search?query=twitch.tv

5. ZSeano

Zseano Methodology 1

Live Recon on Rockstar Games With @zseano

He created bugbountyhunter.com Where he personally triages bugs you submit.

His Recon is "Figuring out how things work". He will be doing Recon on Rockstar Games. Before he does anythig he reads writeups. In the case of Rockstart Games he noticed a lot of XSS and Open basic vulnerabilities. That says to him that he should just go checkout the webapp and see how it works and log some requests.

Zseano Methodology 2

Zseanos Methodology Book

- Don't learn to hack, hack to learn
- Sharing is caring
- Hackers question everything!

Inspirations: @Yaworsk, @rohk_infosec and @ZephrFish

Search google.com, google.es etc. for terms like “responsible disclosure program”, “vulnerability disclosure program”, “vulnerability program rewards”, reward program”, inurl: vulnerability disclosure, inurl: responsible disclosureounty Programs

Toolkit

===

Burp, Collaborator, bappstore

Amass, httprobe, aquatone, anew, dnsgen, ffuf

Seclists, Commonspeak, Inputscanner, Linkfinder, Parameth, Mhmdiaa, Anychange

Process for testing for XSS & filtering:

> Step One: Testing different encoding and checking for any weird behaviour Test `<h1><img> <table>` |-> If it is reflected as < or %3C then try for double encodings like %253C and %26lt; |(TIP) => https://d3adend.org/xss/ghettoBypass Some interesting encodings to try can be
>
> This step is about finding out what's allowed and isn't & how they handle our payload. For example if `<script>` was reflected as <script>, but %26lt;script%26gt; was reflected as `<script>` , then I know I am onto a bypass and I can begin to understand how they are handling encodings (which will help me in later bugs maybe!). If not matter what you try you always see <script> or %3Cscript%3E then the paramerer in question may not be vulnerable.

Issues he finds

He first Starts with XSS to get a feel for the overall security of the app. By testing the filters, you can see what parameters are filtered and what is let through. The interestinng thing about his methodology is you know the site is vulnerable, but the filters were put into place by the developers to stop you from exploting it. The thinking behind that is it means even i f the

- [Burp](Burp Suite | https://support.portswigger.net) - Burp Collaborator | https://portswigger.net/bappstore
- DNSGen | If you want to be really thorough and possibly even find some gems, dnsgen by Patrik Hudak (https://github.com/ProjectAnte/dnsgen) works brilliantly: `cat amass-output.txt | dnsgen - | httprobe` 1
- ffuf | To discover files and directories, FFuF (https://github.com/ffuf/ffuf) is by far the fastest and most customisable, it's worth reading all the documentation, however for basic usage: `ffuf -ac -v -u https://domain/FUZZ -w wordlist.txt.`

Wordlists – Every hacker needs a wordlist and luckily Daniel Miessler has provided us with "SecLists" (https://github.com/danielmiessler/SecLists/) which contains wordlists for every type of scanning you want to do. Grab a list and start scanning to see what you can find. As you continue your hunting you'll soon realize that building your own lists based on keywords found on the program can help aid you in your hunting. The Pentester.io team released "CommonSpeak" which is also extremely useful for generating new wordlists, found here: https://github.com/pentester-io/commonspeak. A detailed post on using this tool can be found at https://pentester.io/commonspeak-bigquery-wordlists/

RESOURCES --- Breaking into information security: Learning the ropes 101” by Andy Gill (@ZephrFish).

Noisy Hacker: Automated XSS Workflow

wd

> Summary: He uses a combination of Param Spider, GXSS, and Dalfox to enumerate the endpoints for params,

> Goal: Find Xss automatically while doing Bounty Hunting. Video here testvuln.txt

ParamSpider

```
python3 paramspider.py --domain testphp.vulnweb.com/ -o /root/Desktop/testvuln.txt
```

Gxss

```
cat testvuln.txt | Gxss -p chocobo
```

Dalfox

```
cat testvuln.txt | Gxss -hoco | dalfox pipe --mining-dict /root/Desktop/Useer/db/params.txt --
```

Cyber UF

Live XSS Using dalfox - Paramspider | Automation Tools, Bugbounty |By-PJBorah #cyberuf

> Paramspider

```
python3 paramspider.py --domain http://testphp.vulnhub.com -o myparam.txt
```

> Dalfox)(https://youtu.be/zI7U7Z1u0Gc?t=355)

```
dalfox file myparam.txt -b me2.xss.ht --proxy http://127.0.0.1:8080
```

> Check Amass Config File

```
amass enum -list -config config.ini
```

Amass with Hacking Simplified

Download the resolvers as see here and here

He has passive and active commands for amass as seen here

```
# Passive Recon
amass enum -passive -d hackerone.com -src -dir h1_amass -o output_h1.txt -rf 50-resolvers.txt

# Active Recon
amass enum -active -d hackerone.com -src -dir h1_amass -o output_h1.txt -rf 50-resolvers.txt

# Track
amass enum -track -config ~/HOME/.config/amass/config.ini -d hackerone.com -dir h1_amass

# Viz
amass viz -d3 -dir h1_amass

# Run Python Server
cd h1_amass && python3 -m http server
```

If runing on a domain with a lot of hosts run using the `screen` command.

Bugbounty recon Automation using bash | Automate your complete recon | @CyberUF

He walks us through setting up a short script (3-4 lines) Here is the basic outline wd:

File: **recon.sh**

```
#!/bin/bash

domain=$1
if [[ -z  $domain ]];

then
echo -e "Usage ./recon.sh <domain>"
exit 1
fi

mkdir "$domain"
```

He then Enumereates over subdomains using `Subfinder` .

```
echo -e "Enumerating subdomains for "$1""
subfinder -d "$domain" -silent > "  $domain"/subs.txt
```

You can run it using `chmod u+x /path/to/recon.sh` and then calling it `./recon.sh google.com` .

Next he gets all the live subdomains using Httpx.

```
echo -e "Enumerating Live Subs for "$1""
cat "$domain"/subs.txt | https -title -tech-detect -status-code > "$domain"/live-subs.txt
```

Next he gets all Urls for the target.

```
echo -e "Enumerating All Urls for "$1""
cat "$domain"/live-subs.txt | waybackurls > "$domain/All-urls.txt"
```

To test his script at this point he comments out httpx because it takes the longest to run out of the all.

This is he end of his Part 1. He is going to release the next part to this in a week which I will post the notes on here. He Notes how you can easily add onto this any of the subdomain enumeration tools like Shodan etc.

[Xss Using Automation | s + grep | Bugbountytrick |By-PJBorah]https://www.youtube.com/watch?v=goclfp6a2IQ](https://www.youtube.com/hashtag/cyberteach)

Extract Parameters, Callbacks, Endpoint etc. Using echo . s and regex.

```
echo 'google.com' |   s | grep "redirect="
echo 'google.com' |   s | grep "url="
```

Enter Dalfox - as we know by now Dalfox is a great tool for hunting against XSS, and also SQLi, SSTI, and more.

```
echo 'https://google.com' |   s | tee outputurls.txts
```

He has the next video which goes over dalfox

Hacktify

Concrete5 CMS : Identification, Mass Hunting, Nuclei Template Writing & Reporting

Video here

Mass Hunt XSS with and without GF Patterns

Video

From Here He has over 99 GF Patterns. List is there. Get a list of them when you need ideas.

He found 136 links. After GF Patterns, it was 36. The point is, GF patterns can greaty limit you when hunting for xss aka paramters. If you grep by "=" you will find way more.

```
curl --path-as-is --insecure "$HOST" | grep -qs "<script>confirm(1)" && echo "$HOST \033[0;31m
erable\n" || echo "$HOST" \033[0;32mNot Vulnerable\n";done
```

First Iteration

```
waybackurls $HOST | tee testphp1.txt | gf xss | egrep -iv ".(jpg|jpeg|gif|css|tif|png|ttf|woff
```

With GF

```
waybackurls $HOST | tee testphp1.txt | grep '=' | egrep -iv ".(jpg|jpeg|gif|css|tif|png|ttf|wo
```

Without GF

```bash
waybackurls $HOST | tee testphp1.txt | grep '=' | qsreplace '"><script>confirm(1)</script>' |
```

The Mass Hunt XSS Technique Bug Bounty Hunters don't want you to know

[[Video](https://www.youtube.com/watch?v=FWtradkn3Vo)] by Hacktify Cyber Security

In a previous video he [came up](https://youtu.be/FWtradkn3Vo?t=122) with the following payloa

```bash
waybackurls testphp.vulnweb.com | tee testphp1.txt | grep "=" | egrep -iv ".(jpg|jpeg|gif|css|
```

He broke this video down into two parts.

1st part - as experienced xss hunters know, in order to find XSS, we need to find parameters. We have all paramters from `waybackurls` with the help of `gf` xss patterns / `grep "="` works as well to find them. Doing an `egrep -iv` clears the clutter. When we have the parameters, we just need to check if they reflect special characters.

1. ***Need to check special chars***

If special characters are ***not*** filtered, and they ***do*** reflect back into the response, we confirm the need to put an xss payload in it.

After we have done this step, we are going to narrow down the scope from say 500 to 50 so we send less requests. So now that we have identified special parameters and narrowed our scope down, it's time to come up with a valid xss payload.

2. ***Valid xss Payload***

It will put dynamic xss payloads to this narrowed down scope, and *only* send a valid xss payload to those that are **not** filtering special parameters.

Essentially, we are being nice to the server and not triggering the WAF.

> cat testphpwayback.txt --> read the file
> | kxss --> filter special chars
> | sed 's/=.*/=/' --> remove everythig after =, add =
> | sed 's /URL: //' --> remove URL: and white space
> | dalfox pipe --> dalfox tool for xss payload
> -b https://me2.xss.ht --> BXSS payload adder.

Read the output of s

After Dalfox pass all urlss to Dalfox usin pipeline mode

For submitting Markdown Reports, Use StackEdit

Blindxss payloads

Bug Bounty Webinar - Recon, Live Attacking, Tools | Live Bugbounty Hunting | #SECOCEANS

Dec 12, 2020

Bugbounty Webinar - Recon,Live attacking, Tools | Live Bugbounty hunting | #SECOCEA

You should know Operating Systems Html CSS Javascript PHP

But if you don't know how to code that's ok.

6. Bash, Python (A programming language) Follow Nehamsec, Pentesterland, Hackactivity, Integrity blbsposgedi q

Information Gathering

What is:- Reconnaissance (Recon) is an imporant tenchinque for penetration testing and the beginnning point of many data breaches. It involves gathering of information about the target which can be useful for finding flaws or vulnerabilities. Many people never do proper reconnaissance and start attacking the targest which is a wrong way.

Step 1: Choose a Target. He uses Google Dorking to find a program.

> responsible disclosure powered by bugcrowd

Tools To Gather Information

- Subfinder => to find the Subdomains
- Httpx
- Waybackurls
  - `cat domains.txt | waybackurls > urls`
- Dalfox

Subfinder Syntax

> Https

Google Dorks

> Google Dorking is justs an advanced technique that is used to search Google's index in a better way. Using this technique you can do a lot of things. You can search for a very specific query or find someone's email and even passwords.

Next he highlights some powerful dorks. **Dork** #1:

```
inurl:circonuss.com intitle: "index of"
```

Dork #2: To find out Database Password

```
inurl:nokia.com filetype:env "DB_PASSWORD"
```

Dork #3: To find Registration Pages

```
site:oanda.com inurl:signup | inurl:register
```

Dork #4: Find exposed Configuration Files - find information exposure

```
site:better.com ext:xml | ext:conf | ext:cnf | ext:reg | ext:inf  | ext:rdp | ext:cfg | ext:tx
```

GitHub Dorking

Starts here there is a repo startinghere that has a list

Search here

```
site:outreach.io "Api_key"
```

end here

Dawgyg

Tools used Aquatone Amass Dirsearch Turbo Intruder

He uses a lot of burp extensions shown here

- Content Type Converter
- Software Versions Repeater
- Software Vulnerability Scanner
- UUID Detector
- XSS Validator
- Wayback Machine
- TokenJar
- Site Map Fetcher (He doesn't have this one active)
- PsychoPATH
- PHP Object Injetion Check
- FIle upload Traverser (Ruby) *Not Active
- Authy
- CMS Scanner

- Collaberator Everwhere
- CSurfer
- J2EE Scann
- Python Scripter (Not Active) Add Custom Header Turbo Intruder Active Scan++ Additional Scanner Checks Backslash Powered Planner Additional CSRF Checks HTTP Request Smuggler Flow Asset Discovery Command Injection Attacker Copy as pyton requests CSP Auditor CSP Bypass * (not active) Directory Importer NGinx Alias Traversal Param Miner Wordlist extractor

Google foo is important and awesome https://youtu.be/GeNJvOvzVSk?t=3987

He almsot always starts with dirsearch. If the 401 page is coming back for everything he switches to TurboIntruder so he can write python and process it Protip from here Always try going to site.com/site, where the directrory folder name is the same as the subdomain name.

He always starts with dirsearch and whenever the return code is odd like 401 for everything that is when he will move to other tools like Turbo Intruder so he has more control over things. He uses them for content discovery for Content Discovery. He like how you can use Python to determine what your ouput looks like with turbointruder here

he has a script he did research on and doesn't want you to run it on synact but anywhere else is fine! Good idea to find that research it sounds legit.

Hackluke

Video: #Nahamcon2022 - @hakluke: Blackbox Monitoring for Timely Bug Detection

In this video Hakluke talks about ways to monitor domains for change. He identifies 3 areas to monitor, Freshness, Changes, and Risk Markers.

Changes

Here for complete list

Risk Markers)

Section 3 Risk Markers[https://youtu.be/LvtCHRIZ0Ac?t=1046]

- Copyright 1995
- Server: Apache 2.2
- Expired SSL Cert
- Keywords like 'internal' in hostname
- Shodan Returns CVE - Shodan does this cool thing Where you put in an ip address and it returns any CVE's it thinks the ip has. Risk markers to look for:

The summary of his talk:

Robots provide:

Uncover Vulns + Risk Prioritize hosts based on risk Tracks Changes Tracks Freshness

Humans Provide: Review Changes Found by Robots Hacks on Hosts that have the highest risk (also found by our robot friends.)

The XSS Rat

Setup your Basic Context

https://youtu.be/5UxdFpd340Q?t=780)

Hackerish

https://thehackerish.com/javascript-enumeration-for-bug-bounty-hunters/
https://thehackerish.com/bug-bounty-tools-from-enumeration-to-reporting/
https://labs.detectify.com/2016/12/15/postmessage-xss-on-a-million-sites/
https://labs.detectify.com/2016/12/08/the-pitfalls-of-postmessage/
https://labs.detectify.com/2012/11/28/xss-where-you-least-expect-it/

https://labs.detectify.com/2016/12/08/the-pitfalls-of-postmessage/ You can check if a page has a registered message listener (and which script registered it) by using Chrome Devtools, under Sources -> Global Listeners:

Jeff Foley

Jeff Foley

How to Use Amass Efficiently by by @jeff_foley #NahamCon2020)

51:37 -So how can I make Amass go faster? You will need to set`-max

You use the `-rf` flag to set the resolvers to use. However, yyou need to keep in mind or take into account that the maximun number of DNS request queries that Amass is going to send out at any given moment is also that same number you gave to -rf. So, if you only change the

He literally 'pulls assetfinder' into his script as a library of sorts.

Zap

Zap in ten

Automating Basic Authorization and Digest Authentication Loggin in Automatically with Zap

Challenge Labs - Basic Authenticion In basic authorization, the server sends a HTTP Response Header `www-authenticate` to signal that the resource is protected behind Basic Authorization.

```
www-authenticate: Basic realm="test"
```

Then when accessing the lab and usingthe supplied `guest:guest` credentials, you see a HTTP Request Header is sent, and set to

```
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
```

That's all find and dandy but we want to do that automatically tomake things easier. Everything is done through contexts. So he is going to

1. right click the url (The request with the Authorization Header) and include it in The Default Context.
2. Then go to Session Management and change it from Cookie Based to HTTP Authentication Session Management.
3. Then, go to the Authentication Page and change it from Manual to HTTP/NTLM.
4. Then fill in the Hostname with 'jigsaw.w3.org' and the port to 443 ~~and the Realm with "test" as the header specified~~ (he did not add the realm.)
5. Add the username and password as 'guest'. The first Username up top is how it displays in Zap, the other one is the actual username.

```
That should be all the info we need. Though, if we actually go in and right c
```

https://play.sonatype.com/watch/ttqKANDzJCAdBUkPrsz6Td?autoplay=2&second=193.21

hacking simplified and Tess #good one his amass command
his nuclei command
his amass -> httpx command

his ffuf command here
ffuf -c -u https:/au.conv.indeed.com/FUZZ -mc Actually his ffuf command here

7:45 - Amass is his go-to subdomain scanner, followed by httpx 20:00 - His Nuclei Command 26:30 - If he finds a 302 and it's a vhost (`httpx -sc -vhost`)
27:00 - Enumerating 302 with ffuf
27:30 - One4all Wordlist - onelistforallshort.txt
28:00 - Github English Words Wordslist words_alpha.txt
28:30 - Using English words wordslist to fuzz for WORD.html and WORD.zip 32:00 - His Nuclei Template for same problem he solved. 40:30 - His Puredns command

As he states quite a few times, he's obsessed with finding subdomains. but it also turns out it's his 'hacking superpower'. He says it is what seperates him. He doesn't manual hakc. He sends that to other hackers. THey split the bounty 50/50 bnecause the trust each other. He uses Amass -> httpx -title `amass enum -passive -norecursive -noalts -d indeed.com | httpx -title

If he sees dashboard or grafana or things like that he goes and checks [them](https://youtu.be/1-IB8TE0Hro?t=484 for a daboard.

Then he copies them and throws them in nulcei to check for any cves

After amass, you think you hve all the subdomains but you actualy don't until you hve portscanned. THey arent running on port 80 or 443

Amass and one4all to subdomain enumeration manually then [sort unique to rengine]. On rengine he runs portscanning. If his Portscaning / finds anything useful it uploads that info to his Discord Server. He has his phone on him 24/7 and if he sees something like [port 22687]((https://youtu.be/1-IB8TE0Hro?t=904), a CGI Panel, Grafana, or 'the Deep Console is up', he just throws that link to Nuclei t THen he uses httpx with -title flag He uses Ports he scans 1-1000 on the fly then He uses Rengine for cron jobs and all port scanning

His Nuclei Command to scan ports after porn scanning `nuclei target http://indeed.hire.emploetfindeed.com:2086 -t nuclei_templates/tehnologies` Another things he does is Enumate ASN. Search Subdomains on Shodan by asn:16509 and then filter by http.title and you will find DIretory Listings olike "Index of /"

nuclei -target http://indeedhire-employer.indeed.com:2086 -t nuclei-templates/technologies/

one file looks like this:

and his other file looks like this: aarp.indeed.com AS16589 AS16589 - AMAZON.COM, INC. US 54.202.0.0/15 about.indeed.com AS16589 AS16589 - AMAZON.COM, INC. US 54.202.0.0/15 account.indeed.com AS16589 AS16589 - AMAZON.COM, INC. US 3.16.0.0/14 activity-report.indeed.com AS62 AS62 - CYPRUSONE LLC US 198.58.5.0/24 From ASN Scanning e

Secret tip Facet Analysis in shodan - The Tops Ports are listed and it shows 5 of them, he clicks on 'more' and that opens up the Facet Analysis Check the top ports for example google what is port

5432. Postgres. his favorite shodan query

Someting aobut client ASNs i'm note xactly sure what he was getting at, check back ter.

This sums up the Shodan / Seccret tip portion, he finds the ASN, searchs it in Shodan, goes to facet analysis, go to http.title and filter out the results. Another one he will do is redis.key when hes on the shodan page of a redis server that is running, ports is another option he will filter by.

He get those ASN's from his portscanner (on rengine)

After running HTTPx,he runs ffuf on 302s, vhosts, 403 he will send it to ffuf. He uses six2dev's Onelistforall, in particular OnelistforallSHort.txt. He also uses Github words wordlist He uses the english WL with ffuf like so: `ffuf -c -u https://au.conv.indeed.com/FUZZ.html -w words_alpha.txt

His Puredns command `puredns bruteforce /root/w/best-dns-wordlist.txt dell.com -r /root/resolvers.txt.1 > trcketst.dell.txt`

My recon process (command-line) by Coffeeaddict

https://medium.com/@coffeeaddict_exe/my-recon-process-command-line-1c9f603f4e9f

Deepak Dhiman to Virdoex Hunter

Deepak Dhiman to Virdoex Hunter | The journey of a Bug bounty hunter | Podcast Episode -2 #bugbounty

His Mentor/Friend Sechunt3r shared a program with him and he found his first bug there which was a file upload restriction bypass. He made a goal to submit 20 bug reports daily

He Doesn't use Checklists, he uses The Latest Writeups and says if you aren't checking (for what was found in the writeup) you are missing out

He talks about how he is making 80k a month

He likes to figure out origin IP and getting access to Admin Panels. He got into 4 admin panels in one month

- Finding Origin IP

and has a tip for us. His notes sneak peak here

Stay Calm While hacking. He [recommends](https://youtu.be/tV6iIhPWgGU?t=2409 thebugbountyhunter.com by zseano

Offtopic SDR & Forensics - Linux OS

His priorities for future hacking are

Web3 SDR Cloud

He recommends http://flaws.cloud/ SDR https://dragonos.org/ for SDR hacking which hes getting into and creating his own radio https://www.rtl-sdr.com/dragonos-debian-linux-with-preinstalled-open-source-sdr-software/ Traceable linux (I couldn't find anything) https://forensictools.dev/listing/deft/ https://www.redhat.com/en/topics/linux/what-is-selinux https://www.youtube.com/watch?v=tV6iIhPWgGU