SANS

FINDING VULNERABILITIES PROCESS PYRAMID

POSTER

FULLY TEST THE SCOPE, EVERY SCRIPT, AND INPUT

sans.org/offensive-operations

PRODUCING HIGH-VALUE PENETRATION TESTS

Reliable and consistent testing is important. And not relying on a single individual's skills and efforts to complete a penetration test will help ensure the highest level of standards.

Team-Based Effort

Penetration testing is a team effort—not an individual effort. Utilize a team to maximize the penetration test efforts.

Thoroughly Map Attack Surfaces

Leave no stone unturned. Many vulnerabilities are found in the paths least traveled. Fully explore!

Reporting

Document findings, processes, discrepancies, and hypotheses. It will be useful now and later.

Hypothesis and Knowledge Sharing

A team is stronger. Produce hypotheses to uncover potential attacks across all layers. Strengthen the team knowledge by working as one.

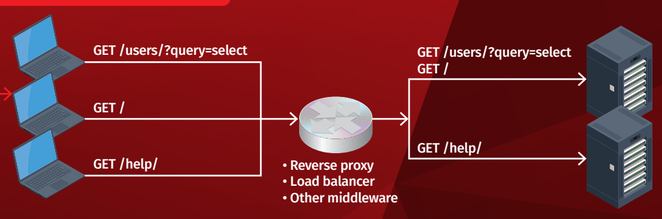
CONTENT DISCOVERY

Goal: Find Everything

Leave no stone unturned. Many vulnerabilities are found in the paths least traveled. Fully explore!

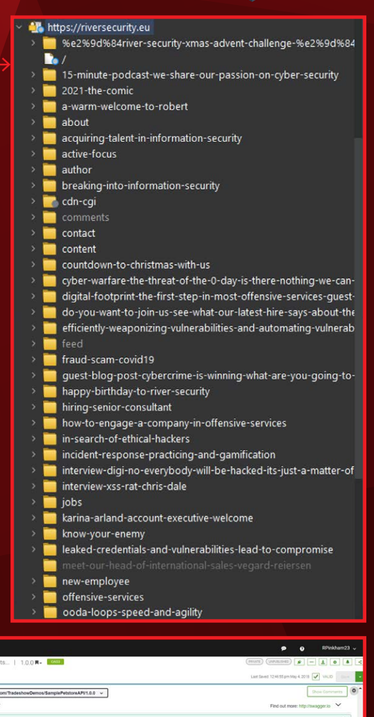
- Map browsable attack surfaces
- Find unlinked content and parameters
- Repeat for each "Platform Distinctions" of the application

Platform Distinctions

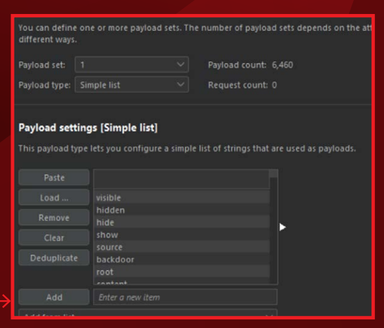


- A web application may have several "platform distinctions"
  - Load-balancers may balance on an endpoint
  - Reverse proxies do the same
- Do your best if the target is split into different platforms
  - Each platform distinction should receive a full-test process

Map Browsable Attack Surfaces

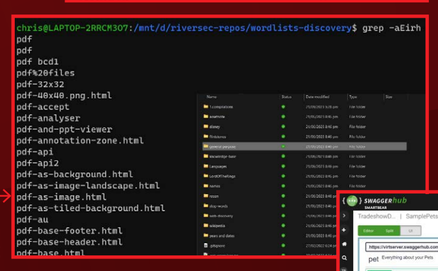


- Browse the entire application, discover all browsable content
  - Click
  - Use
  - Learn
- Use the Burp Suite Crawl feature on the top level of the application.
  - Has decent support for SPA as of Burp Suite v. >2
  - Helps build a complete sitemap
  - Use most complete configuration, which is the slowest
- For JavaScript, extract file paths and references.
  - CyberChef extract file paths module
  - GAP Burp Plugin
  - JSParser



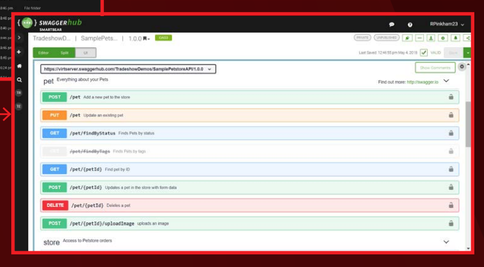
Find Unlinked Content

- For functionality such as **/?action=showUser&id=123**, try fuzzing the verb (e.g., show) with words like: **add**, **delete**, and **update**
- Useful wordlists inside of Burp:
  - Server-side variable names
  - Form Field values
  - Form Field names
- Use and create wordlists based on target functionality:
  - A website relevant to PDFs
  - `grep -aEirh '^pdf.*' * | sort | uniq`



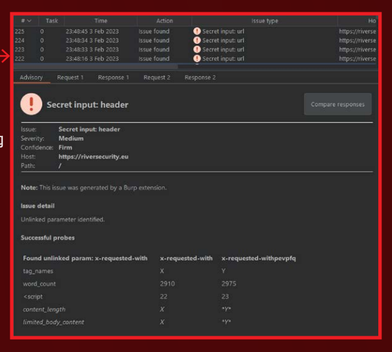
OpenAPI/Swagger Specs

- If developers can give you cheat codes, we should take it!
- Paints a picture of what the developers intended to include
- Still requires us to do content discovery



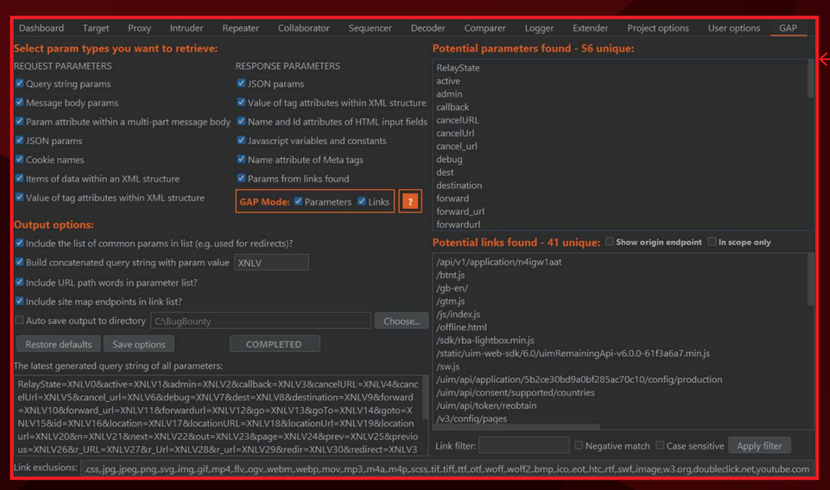
Unlinked Parameters

- Discover if there are any unlinked parameters
  - Particularly important on all **Platform Distinctions**
  - Any change based on a new parameter is interesting
  - GET, POST, Cookies, Headers
- Headers might bypass authentication
- Might find attack surface
- **Param miner extension!**



Other Value Adds

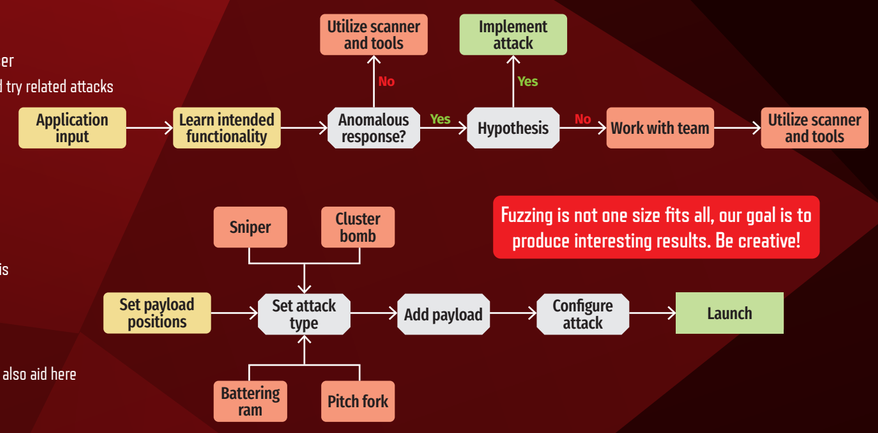
- There are so many ways to improve your content discovery. These are merely guidelines to help sharpen your methodology on finding content.
- Here are some other ideas:
  - Let the blue team help by giving you a directory listing
  - Crystal box testing – provide source code, logs and server access
  - Use archive.org's waybackmachine to help provide crawled content which may still be accessible
  - Use OSINT, such as search engines, to find more content



FUZZING

Fuzzing Bytes 101

- For each script and input
- Send their script to repeater/play with it in browser
  - Determine properly how the functionality works and try related attacks
- Send to intruder and fuzz
  - %00 to %FF
    - URL Decode targets Middleware
    - URL Encode targets App
  - Anomalies, discrepancies, interesting results?
    - Create hypothesis
    - Work with team if you cannot produce hypothesis
  - Use wordlists to stimulate finding anomalies
  - Let wordlists help conclude hypothesis
- Utilize vulnerability scanner
  - Backslash powered scanner—other extensions will also aid here
- Scanner results? Update methodology



Attack Types

Definitions:

- **Position**—A variable where you want to inject a payload of a certain kind (e.g., a word, a wordlist)
- **Sniper**—Loop a payload through all positions
- **Cluster Bomb**—Loop multiple payloads through all positions, like nested for loops
- **Battering Jam**—Push one payload into each position at the same time
- **Pitch Fork**—Apply a set of payloads for each position which is iterated at the same time

Avoiding Rabbit Holes

- A **rabbit hole** is a potential exploit condition which will take up a lot of time to research
- Prioritize **width** rather than **depth**
  - Focus on rabbit holes with the time left after the scope is covered
- Structure your work scope
  - Duration of the engagement/how much time do we have left?
  - Hours spent—work left
    - Each hour spent impacts the total value spent on the engagement
  - How many scripts, functions and other things do we have left to test?
  - **Do we need to get someone else to help us conclude a rabbit hole?**
- Large applications split into smaller parts to help team prioritize

Occam's Razor
Among competing hypotheses, the one with the fewest hypotheses is often correct.

Using Wordlists

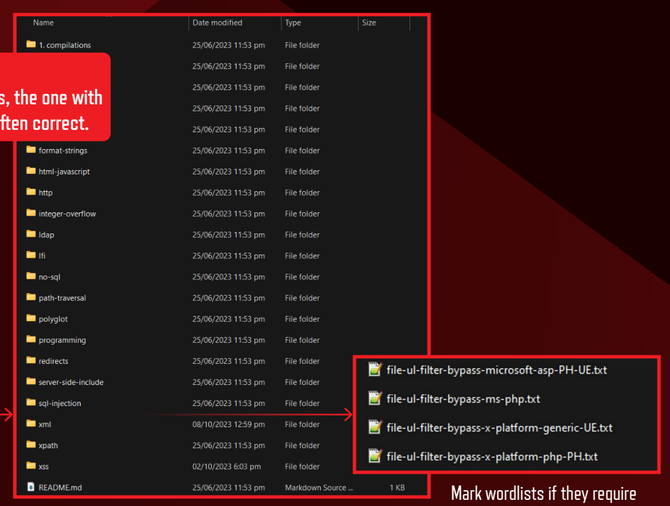
With our fuzzing efforts, wordlists can help produce valuable results (e.g., anomalies) in cases of:

- Different results
- Timing impacted
- External server interaction

Use wordlists that help you target technology and hypotheses.

Great starting points:

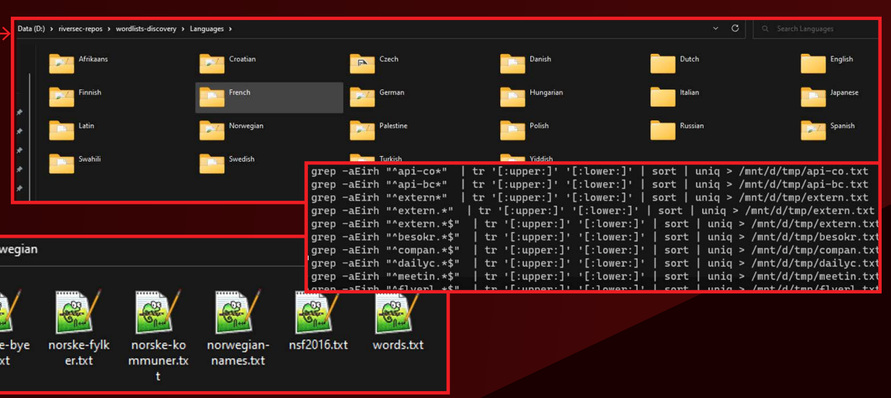
- SecLists: **https://github.com/danielmiessler/SecLists**
- AssetNote: **https://wordlists.assetnote.io/**

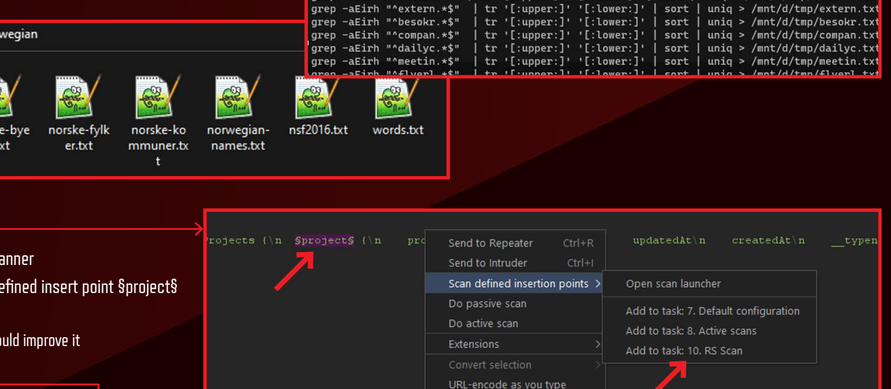


Take the time to learn what these wordlists contain—it will help you learn when to apply them.

Building Good Wordlists

- Roy Solberg's CeWLer
  - Filter away stop-words
- Burp Suite GAP extension
- URL Shortners bruteforce results
- **http_disallowed_entries_CiscoTopMillion**
- Wikipedia is a good source of wordlist





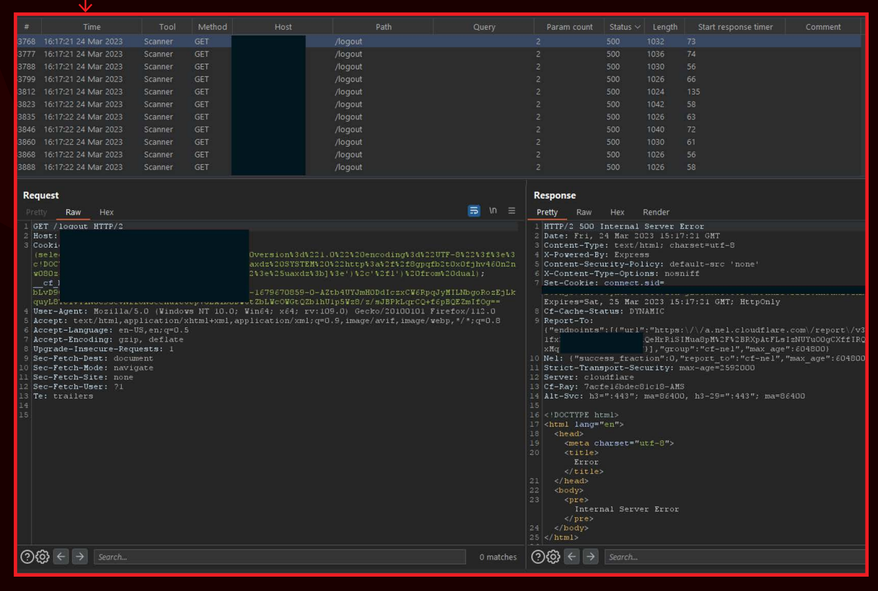
Mark wordlists if they require placeholders (PH) and if they are already URL encoded (UE)

Scan with Plugins and Web App

- Get a second opinion from your vulnerability scanner
- In this case, Burp Suite is tasked to scan the defined insert point §project§
- Does scanner find something?
  - Revisit methodology and ask yourself how you could improve it

Review Logger 500 Errors

- Review the request log
- Look for 500 Error Messages
- There could be potential for exploitation
- Once done, clear the log



SANS TRAINING

SEC542: Web App Penetration Testing and Ethical Hacking™

SEC542 enables students to assess a web application's security posture and convincingly demonstrate the business impact should attackers exploit the discovered vulnerabilities. You will practice the art of exploiting web applications to find flaws in your enterprise's web apps. You'll learn about the attacker's tools and methods and, through detailed hands-on exercises, you will learn a best practice process for web application penetration testing, inject SQL into back-end databases to learn how attackers exfiltrate sensitive data, and utilize cross-site scripting attacks to dominate a target infrastructure. www.sans.org/SEC542

SEC504: Hacker Tools, Techniques, and Incident Handling™

SEC504 helps you develop the skills to conduct incident response investigations.

www.sans.org/SEC504

SEC560: Enterprise Penetration Testing™

SEC560 prepares you to conduct successful penetration testing for entire modern enterprises, including on-prem systems, Azure, and Entra ID.

www.sans.org/SEC560

SANS OFFENSIVE OPERATIONS

FINDING VULNERABILITIES PROCESS PYRAMID

- CONTENT DISCOVERY
- FUZZING
- HYPOTHESES AND TEST CASES
- BUSINESS PROCESSES AND LOGIC FLAWS
- FRAMEWORKS
- TOOLS

CONTINUITY

HYPOTHESES AND TEST CASES

Be creative and utilize your team. Test and conclude hypotheses.

Utilize the Team

- Pen testing is a team effort, not an individual effort
- Utilize a team to maximize the penetration test efforts
- Ensure you can work together
- If you can't properly explain and create valid hypotheses
  - Ask your team
  - Work together— transfer knowledge
- Source your rabbit holes to team members

Hypothesis:
I am seeing that : > < and * are influencing file reads of the file server.
I want to explore Local File Inclusion, SSRF, and similar kinds of vulnerabilities.

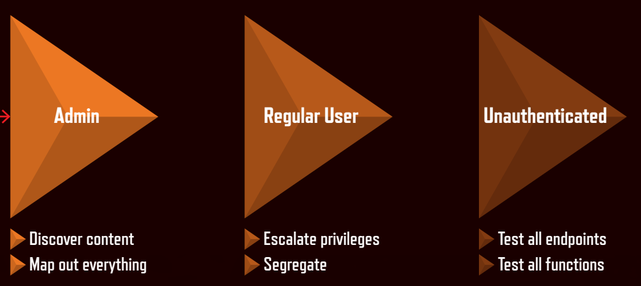
	A	B	C	D	E
1	HTTP Status Code	Byte	URL decoded	Reasoning	Comment
2	500	%25	%	URL	
3	500	%26	&	URL	
4	500	%2A	*	FILE	Wilcard
5	500	%3A	:	FILE	ADS
6	500	%3E	>	FILE	Redirect
7	500	%3F	?	URL	
8	500	%3C	<	FILE	Redirect
9	404	%2B	+	URL	

BUSINESS PROCESSES AND LOGIC FLAWS

With extensive knowledge of the target, explore process and logic flaws

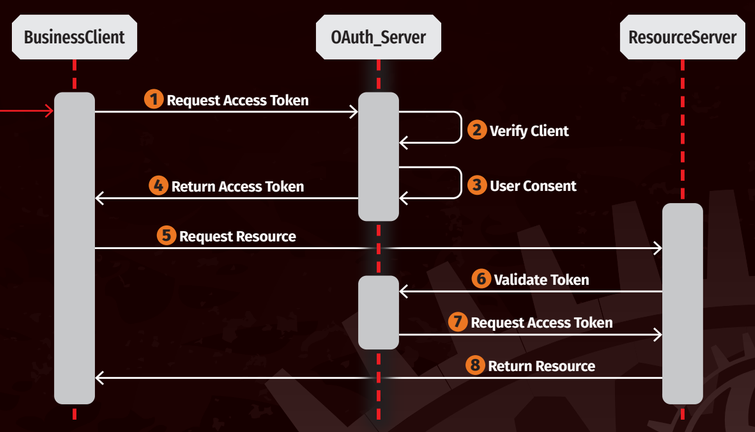
Authentication Example

- Technically a part of discovery/scoping/planning
  - Pentesting is not a one-size-fits-all
  - Work with the customer to find THEIR needs
- Applications typically have different privileges levels:
  - Super Admin
  - Customer admin
  - User
  - Unauthenticated
- Regardless of the scope you have worked through with your customer, ask for super admin
  - Map out everything as super admin—you don't have to pentest it—but build overview of functionality
- Make sure customer admin, user, and unauthenticated are secure, and provide segregation



Map Out Application Flows

- Mapping out the flow of behavior
- **Draw.io** and **Diagrams.net** is an easy quick win
- Helps see things from a bird's-eye perspective
- Map out requests and response
- Example flows:
  - Purchasing
  - Authentication
  - Impersonation/privilege escalation
  - Password reset flow

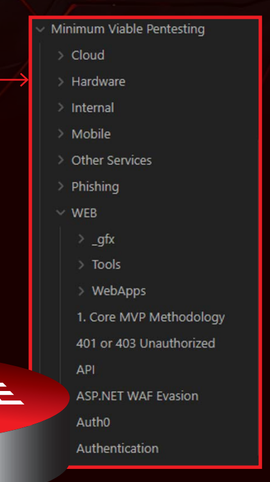


FRAMEWORKS

Compliance and pentest support—utilize frameworks

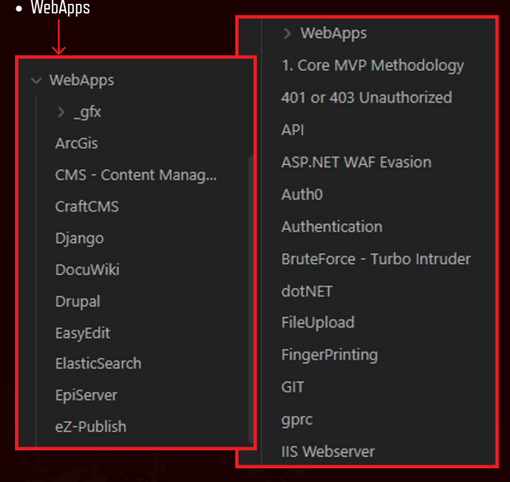
Minimum Viable Pentesting (MVP)

- Define an **absolute minimum** of activity to perform on a certain system or piece of technology or application
- Allow experience from previous tests to be reused
- Find a way to support pentesters—don't start from scratch
  - Your own refined Google/Hacktricks.xyz/etc.
- This is not training on concepts, but simple bullets of what needs to be done
- Make pentester accountable to:
  - Check the things which needs to be checked
  - Ask team for help when there are interesting anomalies
- There are application and technology-specific MVPs



Application and Technology-Specific MVP

- Attack the Stack
  - Middleware
  - Web Server
  - Managed Code
  - Backends
- Testing Frameworks
  - ASVS – Application Security Verification Standard
  - WSTG – Web Security Testing Guide
- WebApps



TOOLS

Vulnerability scanners, application, and technology-specific tools

Tools Are Plentyful

Go find them, review them, or build them yourself. Tools are useful for assisting and automating, but always remember to **seek to understand, not just to solve**. Don't run the tools without understanding how they work, if they fail, and what they're trying to achieve.

THE ULTIMATE GOAL IS CONTINUITY

TESTING IS NOT ONCE A YEAR, BUT THROUGHOUT THE YEAR

- **To effectively prevent threat actors from exploiting new vulnerabilities, continuity is imperative**
- **Trigger penetration testing when changes happen**
- **Example:**
  - Status code changes: 401 unauthorized to 200 OK
  - Swagger.json with new definitions
  - Crawling results with new dynamic scripts
  - CTI with new hacking techniques

sans.org/offensive-operations