API Penetration Testing Tools Techniques and security practices

Article · April 2023

CITATIONS
0

READS
70

1 author:

Enoch Anbu Arasu Ponnuswamy

9 PUBLICATIONS **7** CITATIONS

SEE PROFILE

PenTest

04 / 2023

ISSN 1733-7186

RADIO FREQUENCIES PENTESTING

SETTING UP A **RADIO FREQUENCY** PENETRATION TESTING **LAB**: A COMPREHENSIVE GUIDE

THE **ESSENTIAL** GUIDE TO RADIO FREQUENCY PENETRATION TESTING

EFFICIENT RECONNAISSANCE WITH BBRF: **ORGANIZING ENUMERATION**

API PENETRATION TESTING: **TOOLS, TECHNIQUES** AND SECURITY **PRACTICES**

EDITOR'S WORD

Dear readers

As our modern world continues its relentless march towards increasing connectivity, the silent and invisible realm of radio frequencies (RF) becomes more integral to our daily lives. From Wi-Fi networks in our homes to the Bluetooth devices we use, from the smart cars we drive to the drones that capture breathtaking aerial views - RF signals envelop us. Yet, this omnipresent wave of data, though invisible, is not invulnerable.

This issue is dedicated to exploring the intricate world of RF Pentesting: the practice of assessing the security of wireless technologies through ethical hacking techniques.

You will find here topics such as:

- Wireless Communication Security
- Pentesting Techniques
- Steps of RF Pentesting
- Hardware Tools for RF Pentesting

Radio Frequencies are like the silent whispers of our modern digital age, conveying invaluable data across airwaves. Yet, with great innovation comes great responsibility. The aim of this magazine issue is not just to inform, but to encourage responsible and ethical practices. As we navigate this intricate world, let's remember to wield our newfound knowledge with care, always striving to create a safer, more secure digital environment for all.

So, Dear Reader, whether you're an expert in the field, an enthusiastic newbie, or just a curious soul, there's something in this issue for everyone. Let's tune in to the frequency of understanding, exploration, and responsible action!

Helena Piorun and the **PenTest Team**
helena.piorun@software.com.pl

EDITOR-IN-CHIEF
BARTEK ADACH
BARTEK.ADACH@PENTESTMAG.COM

ASSOCIATE EDITOR
HELENA PIORUN
HELENA.PIORUN@SOFTWARE.COM.PL

Cover Image
Wiktoria Bukowska

Technical Editors
Lee Mckenziie

Cover Design
Wiktoria Bukowska

Reviewers
WhiteHatME, Jan-Tilo Kirchhoff, Anthony Zamore, Ifefemi Obadare, Zaher el-Siddik, Taher Afridi, Momen Eldawakhly, Zechariah Oluleke Akinpelu, Frank M, Daniel Palencia, Alex Samm, Shweta Chawla, Alex D, Olivia, FREREBEAU Laurent, Ivan Suarez, Gabriel Carvalhaes, Justin Smith, Alex Tray, Alberto, Jay Ferron, Luis Reyes, Anslem John, Rishalin Pillay, Fabrizio Baiardi, Kevin Goosie, Justin Smith, Paul Mellen, Daniel Boughton. Bibib, Ross Moore, Cory M, Ranjitha R, jens ulrich, Felipe Martins, Morgan Weetman, Jan-Tilo Kirchhoff, André Luiz, Johan Denoyer, Dan, Marshall Brown, Marco Pacchiardo, Gilbert Oviedo, Amit Chugh, David Michaud, Michal Jachim

PUBLISHER HAKIN9 MEDIA SP. Z O.O. 02-511 WARSZAWA UL. BAGATELA 10/30 PENTESTMAG.COM

Enoch Anbu Arasu

API Penetration Testing: Tools, Techniques and Security Practices

Overview

As we live in a digital era with rapid development of Application Programming Interfaces (API), safeguarding your application data is very important. Application Programming Interfaces have become an integral part of modern application development, facilitating seamless communication between different software components, platforms, and services. As APIs are increasingly used, they are equally targeted by cybercriminals and hence, it has become crucial to protect data.

This article provides a comprehensive understanding about what is API penetration testing, types of testing, its key objectives, common API vulnerabilities, best practices to overcome vulnerabilities and penetration testing methodology. The article also explains the testing tools to ensure secure data handling, validating input and output mechanisms, and verifying authentication and authorization processes. The article also emphasizes API penetration testing as a critical component of a well-rounded cybersecurity approach, enabling organizations to fortify their API security posture and protect their assets against potential threats.

What is API Penetration Testing?

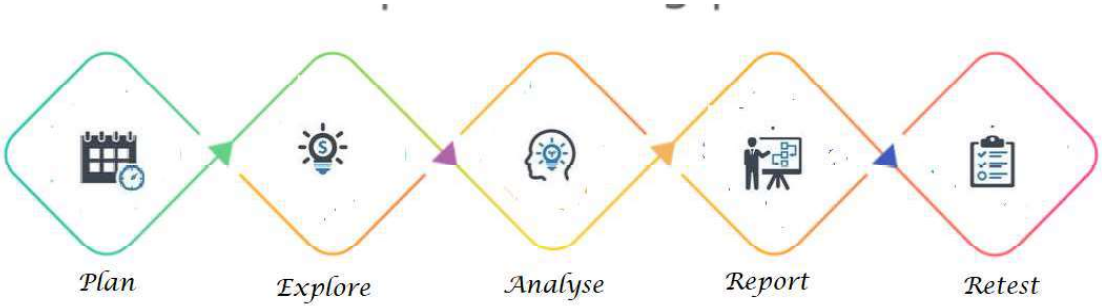
API Penetration testing, also known as pen testing, is a simulated cyber-attack that validates your software application for vulnerable threats. The penetration test helps in assessing the overall security to prevent unauthorized access or a data breach.

Now, let's see the different types of penetration testing you can perform:

- Network Service Testing: Evaluates your network system and the services for probable security issues. Issues could be on servers, routers, switches, DNS, IPs, etc.
- Infrastructure Pen Testing: Assessment of on-premises and cloud network infrastructure, including firewalls, system hosts and devices such as routers and switches. Can be framed as either an internal penetration test, focusing on assets inside the corporate network, or an external penetration test, targeting internet-facing infrastructure. To scope a test, you will need to know the number of internal and external IPs to be tested, network subnet size and number of sites.
- Web Application Test: Tests all functionalities and interfaces with API data in web applications.
- Client-side Testing: Performs actions in client-side application programs like email clients, web browsers and so forth to exploit vulnerabilities.
- Mobile Application Testing: Testing of mobile applications on operating systems, such as Android and iOS, to identify authentication, data, and session handling issues. To perform the test, you must know the operating system types and versions for the application to be tested on, number of API calls and requirements for root detection.
- Wireless Network Testing: Explores identification of wireless networks, vulnerability examination, exploitation, test report, and remediation.
- Cloud Pen Testing: Custom cloud security assessments to help your organization overcome shared responsibility challenges by addressing vulnerabilities in cloud and hybrid environments that expose critical assets.
- Social Engineering Testing: Emphasizes people and processes and vulnerabilities associated with them. This type of test consists of an ethical hacker directing attacks such as phishing or impersonating a person during the course of their work.

How API Pen Testing Works

Let's see how a pen testing process works and how you can perform effectively.



Plan scope. This phase determines the testing methods and the level of exploitation required when trying to find vulnerabilities in the API.

Explore vulnerabilities. This phase discovers all possible vulnerabilities or intrusions.

Perform pen testing. This phase tests all possible exploitations to discover new vulnerabilities. The web application is attacked to uncover vulnerabilities and check if a bad actor is able to gain access to the application.

Analysis risks and threats. This phase lists out the vulnerabilities found during the test and provides recommendations to address security threats. The penetration test engineer starts the enumeration task of the target API on both application and network layers. This involves identifying and noting the usernames, machine names, network resources and application services. This process helps to understand the weaknesses or loopholes present in the API.

Generate test report. This phase provides a detailed report of threats along with severities so that the organization can prioritize and fix the security issues. A well-structured, detailed, informative, and meaningful test report is a mark of a great test engineer. After completing the penetration test, the engineer must report the vulnerabilities, levels of penetration, and risk rating. All findings are detailed, and

recommendations are provided as to how the company can plug the vulnerabilities.

Retesting. This must be ideally performed by the same API penetration test engineer who performed the original activity as they are now familiar with the application, can appropriately attest to the effectiveness of the resolution, and ensure that no other security loophole has been exposed in the process.

By performing these tests, security experts can evaluate the robustness of an API's security measures and provide recommendations for improvements.

Why Is API Penetration Testing Essential?

API has become essential to modern software development, allowing different software applications to communicate and share data. However, the increased connectivity also poses significant security challenges. APIs can be vulnerable to attacks from malicious hackers trying to exploit data for their purposes. This has led to a growing focus on API security in recent years, and it is driven by the following key factors:

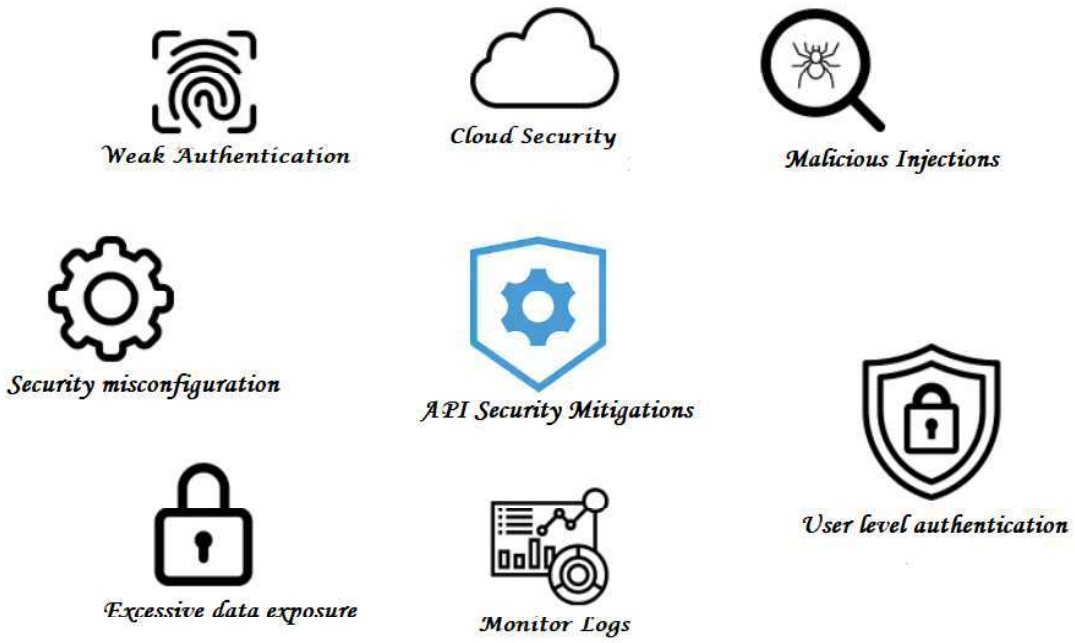
Digital Transformation – Today's businesses are moving their operations online and have digital technologies. They increasingly rely on APIs to integrate different systems and services. However, this also includes sensitive data transmitted through APIs, creating potential security risks.

Cloud Computing – Cloud-based applications and services rely on APIs to exchange data and interact with each other. Any security vulnerabilities in these APIs can have far-reaching consequences.

Easy to Bypass Security Measures – API vulnerabilities and security weaknesses are unique, and so are the security risks. Organizations rely on security solutions built for web apps to detect and secure them from API threats. Such solutions cannot detect unique vulnerabilities and gaps in APIs. So, attackers can effortlessly exploit APIs by bypassing security measures.

Common API Vulnerabilities and Their Mitigations

When securing API, it is important to identify the common vulnerabilities that consistently lead to data breaches that includes a financial cost to your organization. Here are some of the common API vulnerabilities, their implications, and recommended mitigation.



Weak authentication – Authentication verifies users or devices accessing the system. In some cases, weak authentication means that there is no authentication mechanism in place at all. In some scenarios, there are misconfigurations or improper settings that weaken authentication, including weak password complexity, high account lockout thresholds or relying on API keys as the only way to authenticate.

The potential outcomes of exploiting weak authentication are that outsiders can control user accounts or sessions, steal their data, or engage in fraudulent transactions.

Recommended mitigation:

- Enforce strong passwords.
- Avoid relying on API keys to authenticate users.

- Use multi-factor authentication to strengthen the possibility of account takeover.

Security misconfiguration – The API system is a collection of virtual machines, containers, on-premises, and cloud infrastructure. From the specification of each API, it is identified that there is a wide scope for security misconfigurations within each layer.

Recommended mitigation:

- Ensure APIs are tested for vulnerabilities coinciding with DevOps workflows, not at later stages.
- Set controls over request volumes to limit DDoS attacks.
- Assess and review API security configurations, using an accurate inventory of all your APIs.

Excessive data exposure – This vulnerability occurs when the API responses provide too much information to the client application. The reliance on the client-side can lead to excessive information being presented to end users. Unfortunately, threat actors are privy to these data exposures and can attempt to circumvent the client or analyze traffic to intercept API responses containing excessive information.

Recommended mitigation:

- Filter response data at the API level rather at the client level.
- Provide information in API responses that are strictly necessary for a client request.
- Control authentication for transmitting sensitive personal information.

Broken object level authorization (BOLA) – This is a simple API vulnerability that occurs when a user tweaks the object IDs in an API request. There is no check in place to determine ownership of those objects. By simply changing the ID in a request, the malicious actors get access to information.

Recommended mitigation:

- Combine identifiers with a check so that the user can access the resource.
- Use random and unpredictable values for IDs rather than easily guessable values.
- Consider obfuscating resource identifiers altogether.

Malicious Injections – Attackers use a range of data injections through JavaScript, SQL, NoSQL, and OS command lines. When there are injection flaws in the code, such as directly linking client-supplied data to SQL/NoSQL, JavaScript queries, or OS commands, the API's interpreter bypasses security and executes the malicious commands.

Recommended mitigation:

- Validate data so that your API only allows valid values.
- Implement JavaScript dependency injection.
- Use a security protection that identifies anomalous client-side behavior, including unexpected scripts, parameters, or other code injections.

Best Practices for API Security

The following security best practices can help mitigate attacks and secure your APIs:

Use throttling and rate-limiting – Throttling involves configuring a temporary state that allows the API to evaluate every request and is often used as an anti-spam measure or to prevent denial-of-service attacks.

On the other hand, rate-limiting helps you to manage REST APIs by avoiding DoS and Brute force attacks. In some APIs, developers set soft limits, which allow clients to exceed request limits for a brief duration. Setting timeouts is one of the most straightforward API security best practices, as it can handle both synchronous and asynchronous requests. Request queues enable you to create APIs that accept a maximum number of requests and then put the rest in a waiting queue.

Each API application comes with a library to implement the request queues.

Scan for API Vulnerabilities – To secure APIs, it is important to automate API scanning, identify vulnerabilities, and mitigate them in software lifecycle stages. Automated scanning tools autonomously detect security threats by comparing the application's configuration against a known vulnerabilities database.

Use HTTPS/TLS for REST APIs – HTTPS and Transport Layer Security (TLS) offer a secured protocol to transfer encrypted data between web browsers and servers. Apart from other forms of information, HTTPS also helps to protect authentication credentials in transit. As one of the most critical practices, every API should implement HTTPS for integrity, confidentiality, and authenticity. In addition, security teams should consider using mutually authenticated client-side certificates that provide extra protection for sensitive data and services. When building a secure REST API, developers should avoid redirecting HTTP to HTTPS, which may break API client security. Adequate steps should also be taken to divert Cross-Origin Resource Sharing (CORS) and JSONP requests for their fundamental vulnerabilities for cross-domain calls.

Restrict HTTP Methods to Secure APIs – REST APIs enable web applications that execute various possible HTTP verb operations. Data transmitted over HTTP is unencrypted and some HTTP methods are intercepted and exploited by attack vectors. As a recommended practice, HTTP methods (GET, PUT, DELETE, POST, etc.) that are inherently insecure must be forbidden.

If you are not able to completely forbid the usage, security teams can apply policies to control the usage with a strict allow list, whereby all requests that do not match the list should be rejected. It is also recommended to utilize RESTful API authentication best practices to ensure that the requesting client can use the specified HTTP method on the action, record, and resource collection.

Implement sufficient input validation – The data supplied by the API client must not be trusted blindly since the authentication server may

execute a malicious script from unauthorized users or application services. To avoid this, security teams should implement input validation mechanisms on both the client and server sides to prevent unhealthy input. While client-side validation involves interactive indication of errors and advice to a user on acceptable inputs, server-side validation additionally checks the data received to avoid the different types of XSS and SQL Injection attacks.

Use an API Gateway – An API gateway decouples the client interface from the collection of backend APIs, delivering a centralized resource for consistent availability and scalability of API services. Apart from managing various API services, the API management platform also handles standard functions, including telemetry, rate limiting, and user authentication, to maintain security between internal services. The gateway acts as a reverse proxy gatekeeper that accepts all API calls, coordinates the resources required to service them, and returns the appropriate results post-authentication.

Do not leave any open endpoints – One of the most important points to keep in mind when you're developing the API is that you should never create an open API endpoint. It means that every API endpoint needs to be secured by the authorization headers or tools you're using to secure your API.

Monitor logs – Monitor every detail of your API. Be prepared for any error, breach, or service down. Maintaining a log can help you in the long run as it keeps a record of everything, especially the errors. Logs can help you debug API issues and be used to generate a dashboard to assist consumers when there is downtime.

API Security Testing Tools to Mitigate Threats

With frequent cyber-attacks looming, the challenge is to protect your API resources from data breaches that can permanently damage your organization's reputation. Following are some top best API security testing tools to mitigate risks.

APIsec – The APIsec platform acts like a penetration tool for APIs. There

are many tools that can scan for common vulnerabilities to typical attacks, like script injections, but APIsec stress tests every aspect of targeted APIs to ensure that everything from the core network to the endpoints accessing it are protected from flaws in the API's code. One big advantage to APIsec is that it can be deployed in the development phase while APIs are being programmed. A full scan of your application that is in the process takes afew minutes, with results comparable to old-school penetration testing operations that used to take days or weeks to complete.

Postman – Postman, while prominently serving as a platform to build secure APIs, also provides API pen testing features. Its notable attributes include a user-friendly interface, availability as a Chrome plugin as well as for Windows and Linux, and other integration capabilities.

Synopsis API Scanner – One reason why the Synopsis API Scanner is so powerful is because in addition to security testing, it also incorporates fuzzing as part of its deep scan and tests. The fuzzing engine sends thousands of unexpected or random inputs to APIs to see how they behave or if they break when there are a very large number of odd commands.

It also maps out all the paths and the logic of an entire API, including all the endpoints, parameters, authentications, and specifications that apply to its use. This gives developers a clear picture of what functions their APIs perform, compared with what they might sometimes do. It makes it clear why an API might be subject to unexpected behavior or security vulnerabilities.

AppKnox – This tool, with its easy-to-use interface, is a good choice for organizations that don't have large security teams dedicated to their APIs. AppKnox starts with a scan to locate APIs either in the production environment, on endpoints, or wherever they may be deployed. Once located, users can select which APIs they want to submit for further testing. The tool tests for all the common problems that can cause an API to break or become compromised, like command injection vulnerabilities in HTTP requests, cross-site tracing, and SQL injection

vulnerabilities. This includes a complete analysis of web servers, databases and all components on the server that interact with the API.

After the API scan, users can submit their results for advanced analysis with a human security researcher, a process the company says normally takes between three and five days.

HAWK Authentication – HAWK is a relatively new authentication technology created by the original developers of OAuth. HAWK aims to replace the 2-legged authentication and use a much simpler form of authentication. It is primarily used for HTTP authentications and uses HMAC digests.

Like HTTP Basic authentication, HAWK also uses client credentials, including a cryptographic verifier and a key. However, the key is never used for authentication but for calculating the MAC value of the request. To make the HTTP requests, the client needs to send a token id and token key from the server. When the server needs to send the token, it uses Hawk-Session-Token as the header. The client can break the header's value into two parts, i.e., HAWK ID and HAWK token, which will be used in the subsequent request.

Conclusion

In today's world, many applications are fast developed and deployed without security testing and validation. To have a secure and reliable application, it is important to continuously test and validate it for vulnerable threats in real time. Always ensure the authentication of your users with the right access and right set of permissions. Always follow the best practices to mitigate attacks and secure APIs to the best of your ability. Use the right API security tools that keep you ahead of the threats endangering you.

Always turn on security to keep vulnerabilities at bay!