International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I11P101

Volume 72 Issue 11, 1-7, November 2024

Original Article

Examine the Role of WAAP, WAF, TLS and mTLS in Protecting APIs from Advance Cyber Attacks

Piyush Dixit

Director - Integrations & API Software Engineering, Cummins Inc., Indiana, USA.

Corresponding Author : piyushdixitwork@gmail.com

Received: 15 September 2024 Revised: 17 October 2024 Accepted: 03 November 2024 Published: 20 November 2024

Abstract - *Cyber-attacks on Application Programming Interfaces (APIs) have become extremely advanced and sophisticated, posing novel challenges in securing APIs. This has generated a dire need to use equally sophisticated cyber security tools for protection. APIs have become unarguably indispensable in connecting disparate software application systems both within and outside an enterprise. APIs help to move data effectively and even help organizations generate revenue by selling data and services. These factors have significantly surged the number of APIs that are being built, consequently increasing the cyber-attack exposure for the companies, exposing them over the web for bad actors to exploit. Attackers often exploit numerous vulnerabilities in APIs left behind due to poor cyber security practices during implementation or hosting. The vulnerabilities enable bad actors to gain unauthorized access to sensitive data and systems within an organization. What has worked as fuel to the fire is the easy availability of malicious no-code type software and tools that can launch automated attacks, bypass standard security measures in place, stay completely undetected, and sometimes even be untraced from intrusion detection systems. There is a gap in current research on these topics which only highlights the necessity to implement some basic cyber defense mechanisms but does not specifically highlight the role and usage of some advance tools like WAAP, WAF, TLS & mTLS, which help bolster API security. This study aims to examine and present these advanced protection tools available to defend against complicated modern cyber-attacks and establish an approach to how organizations can implement these security measures to protect APIs.*

Keywords - *API Security, WAF, WAAP, TLS, mTLS, OSI, Layer 7, Layer 4, TCP/IP, HTTPS.*

1. Introduction

Advanced API security involves implementing sophisticated security tools and adhering to mature security processes based on principles like zero trust, need-based access and, most importantly, Defence in Depth (DiD), which basically uses multi-layered controls to protect assets. There are numerous tools available to implement these security principals. Some of those tools and protocols are WAAP/ WAF, TLS, and mTLS, which, although sophisticated, are also extremely standard to implement. Just with some basic understanding, they can be utilized to implement a robust security regime. There is a significant research gap in this field when it comes to clearly highlighting the utility of these sophisticated security tools and technology, and that is the purpose of this article: to examine and understand these tools and protocols. The novelty of this study is to clearly detail the relevant nuances of these tools and layout a crisp case for their utilization. An API operates over OSI layer 7 using the HTTP protocol of the application layer, which is built on TCP/IP, which is layer 4 protocol, also known as the transport layer. The majority of organizations are very efficient when it comes to protecting layer 4 TCP/IP traffic. It is typically done using network layer firewalls and other network security devices that act as security watchdogs. At layer 4, the security measures typically revolve around scanning IP addresses, checking for ports, evaluating packet header trailers, and inspecting overall TCP/IP packet signatures to detect anomalies and ultimately block threats that fail to pass the scrutiny. But the information or data at layer 4 is typically encoded in TCP/IP protocol packet structure, making these firewalls partially or sometimes completely blind in detecting threats that exist in an encapsulated form and those threats are smartly packaged to reveal and specifically target the time of layer 7 decapsulations when the message is revealed in its final form for consumption in the application layer by communicating systems. Regarding encryption, another important pillar of security of API traffic, TLS-based encryption, is the gold standard for ensuring that all data in motion is encrypted. All modern API gateways and web servers hosting APIs come equipped with out-of-the-box support for the latest TLS version-based encryption, implemented using public and private keys or certificates often issued by independent and industry-recognized trusted third-party certificate authorities. Mutual TLS (mTLS) takes

this another notch by ensuring the server and client are authenticated using respective security certificates. mTLS works on the principal of zero trust and forces double validation to avoid impersonation. This article aims to examine and layout a study of these advanced mechanisms available to effectively secure APIs.

2. Examining WAF and WAAP

2.1. Layer 7, the most attacked layer

Layer 7, also known as the application layer in the OSI model, is vulnerable to the most sophisticated attacks because it is the most feature-rich layer. This layer offers several complex capabilities through its high-end protocols, like HTTP, SFTP or SMTP, that ultimately enable the effective exchange of information between systems, usually referred to as client and server information exchange. Figure 1 shows the OSI model with protocols supported in layer 7 and layer 4. Attackers these days get very crafty when it comes to exploiting these protocols and, hence, can either steal the data or gain unauthorized access. The most common and effective attacks available to be executed at this layer are the BoT attack, API JSON/XML injection attack, and API Parameter Tempering attack. With basic knowledge of the http request-response structure and access to nominal computing power, any of these attacks can be executed against unprotected APIs. This is why this layer is special and most lucrative to bad actors. WAF and WAAP solutions protect secure traffic, specifically in layer seven, the application layer.

2.2. Web Application Firewall – WAF

A web application firewall is a special layer 7 protection firewall software that protects web traffic by detecting and blocking threats that impact the layer's higher-level protocols. WAF can be installed as a network software either on-premise or on a private cloud, or SaaS providers can avail of it as a cloud offering. It can protect against common threats like zero-day exploits, malware infections, and impersonation. WAF can inspect each packet, and it uses a rich static rule repository and security policies to analyze Layer 7 web application traffic and filter out harmful traffic that can cause exploits. Figure 2 shows how a WAF solution that is based on pre-configured rules and restriction-based policies is implemented on top of the application or web servers rendering API and how a WAF can differentiate between valid requests versus malicious requests and can reject invalid traffic prohibiting that from reaching the protected asset, which in this case is a web server. Further through granular packet inspection, WAF can detect and prevent security threats, which traditional network firewalls and other intrusion detection systems and intrusion prevention systems might not be able to do. These days, it is very common to place a cloud-based WAF solution in front of the website, and that is exactly where it shines. However, the same cannot be said about protecting APIs using WAF. Although both APIs and Websites run on the HTTP protocol over the web, the request-response messages exchanged in both are totally different. Websites deal in HTML, JS, and CSS, while APIs deal in payloads, typically in JSON, XML, or sometimes flat files. A website's purpose and implementation structure fundamentally differ from an API. WAF has served the web world for a very long duration of time, and still, it is standing the test of time when it comes to protecting websites. However, in order to protect APIs, a more appropriate solution is needed. It has gained popularity in the security space recently; it is referred to as WAAP – Web Application and API Protection.

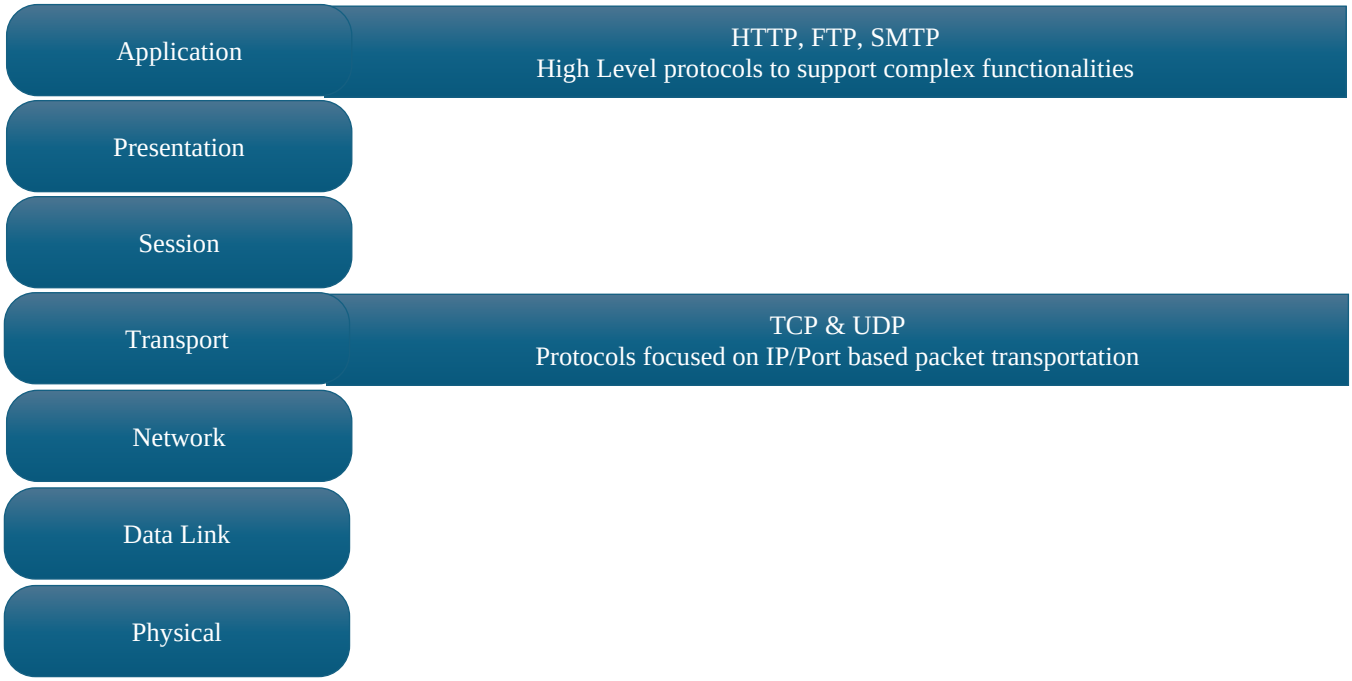

Fig. 1 OSI Model 7 layers with protocols in layers 7 and 4

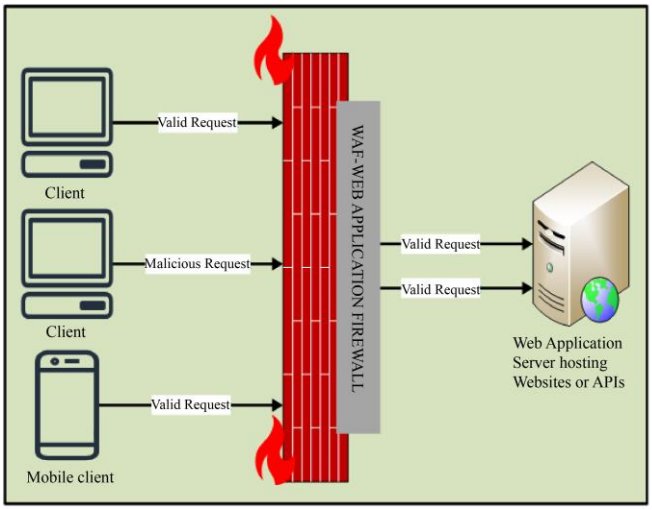

Fig. 2 WAF only allows valid requests to the web server

2.3. Web Application and API Protection – WAAP

WAAP Web application and API protection is not just a web application firewall like WAF. WAAP is a more powerful product that encompasses capability to protect web applications and provides exclusive tools referred to as holistic API protection solutions powered by artificial intelligence and automation, enabling it to provide end-to-end protection against a wide range of multi-vector attacks on APIs. WAAP can offer advanced features like automatic API discovery, adaptive detections, built-in bot mitigation, and continuous self-tuning as the threat landscape changes. Modern enterprise applications these days are either from SaaS products from some provider or are home-grown web and software applications that heavily rely on microservices and APIs for numerous business transactions, either within or to and from outside the organization with partners. These multiple digital applications, using APIs, eventually significantly expand the attack surface with a new and wide range of entry points now available for hackers to exploit to gain access or steal data.

The most common form, although not the only form, of implementing WAAP solutions that are available today is in the form of some cloud-based SaaS WAAP solution that can be put in front of the web traffic of an organization, especially API traffic as soon as it lands on the DNS server for IP resolution of the company domain. The domain names can be resolved to the IP addresses of the WAAP solution, where the entire traffic can be routed for deeper inspection and scrutiny via policies and security rules. It is post-competing all the evaluations in the WAAP, and the traffic can be routed to the regular network or application load balancers of the organization's application web server.

As explained in Figure 3, a typical WAAP implementation solution flow starts from the client sending the request to a DNS server to resolve the IP address for a domain where the API is hosted to receive a desired response, which is typically some data or an action requested. The DNS resolved the domain into the IP addresses hosted by the WAAP SaaS solution. This WAAP solution deeply inspects the request packet, scrutinizing it using traditional static rules and policies, and compares it against newly learnt threat signatures with the power of AI and ML.

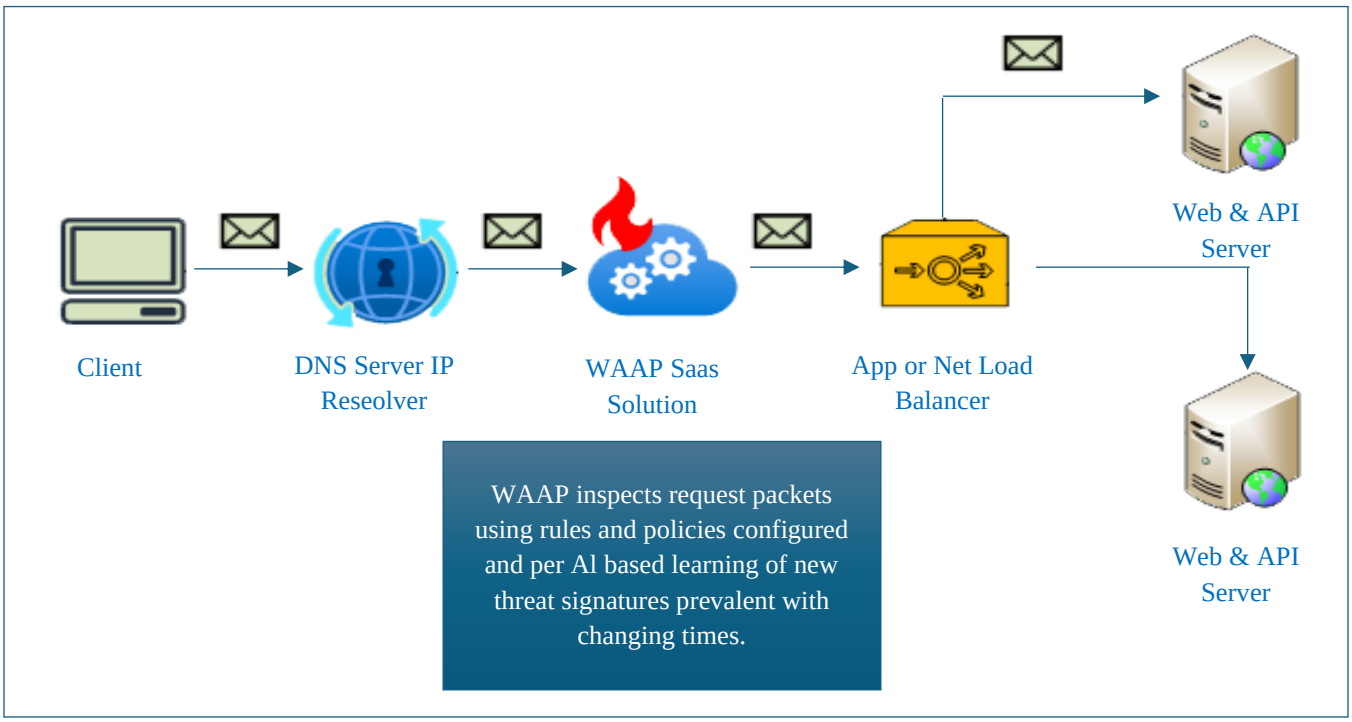

Fig. 3 Most common WAAP implementation solution flow

Only after WAAP determines the request packets to be safe does it finally route them to the load balancers that manage the load for the application or web server hosting or serving the API. Typically, WAAP solutions offer company admin personnel rich UI-based management portals that can be used to write new security policies, implement custom validation rules and, most importantly, audit the historical logs to have a detailed snapshot of the historical behavior of the type of traffic that has tried to reach companies web assets. This empowers organizations to manage and have clear visibility of existing threat assessments. The most lethal strength of a modern and effective WAAP solution is its underlying AI/ML models that constantly learn the change in the pattern of attacks by getting trained on daily traffic activities and actions either supervised due to actions taken by admin personnel or unsupervised by observing the signature of the traffic packets. This is why organizations dealing with high internet-based interactions over the web with clients or customers should invest in a robust and dynamically learning WAAP solution to protect APIs.

3. Examining TLS - Transport Layer Security

3.1. TLS is an encryption protocol

TLS is a data encryption protocol that uses high-end cryptography to encrypt data exchange between digital devices on the internet. Encryption of data being transferred ensures that any information exchanged between the client and server remains confidential and cannot be intercepted by unauthorized entities. Even if intercepted, it remains uninterpretable, protecting the data's confidentiality. TLS ensures that the entities exchanging information systemically over the web are the ones that they claim to be per the domain name. TLS enforces the use of digital certificates issued by trusted certificate authorities to achieve this goal of validating the web asset. The server shares a public certificate with the client trying to establish a connection; the client, on the other side, validates the authenticity of the certificate before proceeding with the communication with the server; this ensures that the client is establishing a connection with the right server and not a malicious imposter asset pretending to be the genuine server.

Figure 4 shows a TLS handshake as a very high-level flow diagram. The TLS handshake is a process that establishes a secure connection between the client and server. During this handshake process, before any business data is exchanged, the client and server agree on the TLS version cipher suite that would be used, and some other cryptographic parameters necessary for establishing a trusted and secure connection are exchanged. As shown in Figure 4, the process starts with Step 1, when a client initiates a connection request to a server by sending a hello message. In step 2, the server responds by sharing server public certificate issues by a trusted certificate authority. In Step 3, the client first validates and verifies the certificate shared by the server to ensure that the server is authentic.

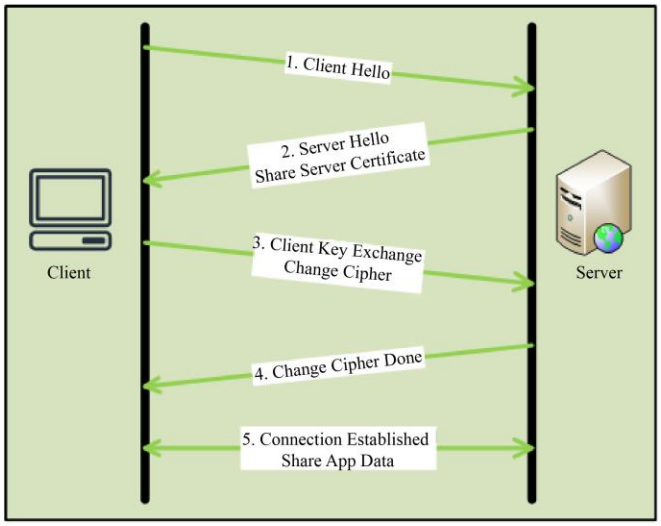

Fig. 4 TLS Handshake between Client and Server

After confirmation, the client returns to the server with the preferred cypher specification for communication. In Step 4, the server responds with a confirmation of the change in the cipher. Finally, in Step 5, a trusted connection is established between client and server to exchange fully encrypted data due to the TLS handshake preceding data exchange. There are few compliance standards globally and sometimes within the boundary of a region or a country that mandate enforcement of TLS-based encryption to protect sensitive information. So, TLS not only provides information security for an organization dealing with sensitive data, but the implementation of TLS also ensures that the organization is not violating most of the prominent compliance standards like PCI-DSS, HIPPA, GDPR, etc.

However, a few things require very careful attention while setting up and managing TLS configurations. This includes, although not only limited to, a few key considerations like selecting strong cipher suites, regularly checking expired certificates and updating certificates, and disabling deprecated protocols and older TLS versions by upgrading to newer ones. Without these careful considerations, there is always a risk of inheriting vulnerabilities arising from weak TLS implementations.

3.2. Choosing strong Cipher Suites

Cipher suites are cryptographic algorithms used to encrypt the data. They often are an amalgamation of multiple algorithms offering logic to govern key exchange, message authentication, hashing mechanism and overall encryption. However, the same ciphers cannot always stay relevant. As threats evolve with time, the ciphers need upgrades, which is why any TLS implementation needs to ensure the cipher suites being utilized are of the latest and greatest version available and compatible. Weak an outdated algorithms can be exploited with the increases compute power available today at the disposal of any bad actor. Advanced Encryption Standard

(AES) and Secure Hash Algorithm 256-bit (SHA-256) are very common and strong algorithms prevalent today.

3.3. Regularly updating TLS security certificates

A TLS certificate is a digital artifact that sometimes allows systems like servers or even clients to prove and verify identity. The certificate also enables the establishment of an encrypted network connection between the two systems using protocols like TLS. These are preferably issued by a recognized certificate authority that acts as a trusted third party between client and the server by providing public certificates that act as a trusted digital identity card for web assets. Figure 5 shows some contents of a TLS Certificate. The figure above only shows the public key that is contained inside the certificate, but another part of the key called the private key, is not included in the certificate. Both public and private keys are generated as a pair of asymmetric keys.

Where the public key is used to encrypt the data, and this key is publicly accessible, while the private key is used to decrypt the data and is only accessible to the owner of the private key; this is commonly referred to as asymmetric key-based encryption that is implemented utilizing these certificates. Certificates also offer an important feature that enables parties to digitally sign the message, which enables the receiving party to confirm that the rightful owner was sending the message.

An important feature of TLS certificates is the limited time-bound validity, after which certificates expire and need renewal from the issuing trusted third party. This ensures periodic governance and checks on the identity of the party requesting the issuance of the certificate. It also reduces the chances of misuse of the certificate, retaining its utility. This is why it is important for organizations implementing TLS mechanisms to monitor certificate expiry dates systemically via some certificate management software to ensure timely renewal of certificates and avoid any disruption or risk caused by lapse.

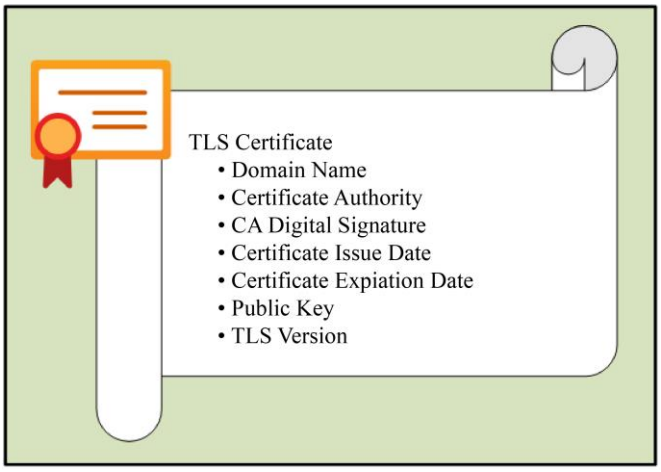

Fig. 5 TLS Certificate list of contents

3.4. Regularly upgrading TLS protocol versions.

As technology advances daily, so do the exploitation techniques, which means any current TLS version becomes outdated and is superseded by the latest version that offers stronger protection relevant to current times. With time, some vulnerabilities get discovered and exposed with an existing version, and newer versions often are seen as a guarantee to address newly emerged vulnerabilities and weaknesses from the older versions. Upgrading the version and keeping it up to date ensures protection from known vulnerabilities. Just upgrading to the latest protocol version is not enough by itself; it's also important that older versions are disabled or deprecated to ensure that there is no fallback option to communicate using an inferior protocol that is still active, putting all the upgrade effort to waste and jeopardizing the security of the digital asset and the information.

4. Examining mTLS – Mutual TLS

4.1. Mutual authentication

While TLS enables only the client to validate the server it is connected to, mTLS takes the capability to another notch by allowing the server to validate the calling party or client. mTLS enforces a true zero-trust connection to ensure further security and authenticity for safe data exchange. So, in the case of an mTLS-based connection, both the client and the server hold a public certificate that trusted third-party issues, and both sides can authenticate using their respective private keys. Figure 6 shows a high-level handshake flow between client and server using mTLS.

As visible, both client and server share respective certificates and only after verification on both sides is the connection established, which is trusted and secure in both directions. mTLS provides a robust mechanism to defend against some common attacks that target APIs. The two most prominent and common attacks carried out against APIs are Man in The Middle and stuffing stolen credentials attacks. These attacks often circumvent the secure channel established by TLS by impersonation and spoofing. These days, these attacks can be easily launched using simple UI-based tools with minimal effort.

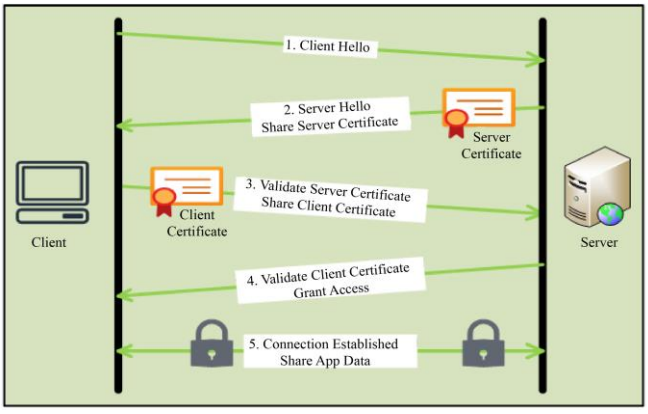

Fig. 6 TLS Certificate list of contents

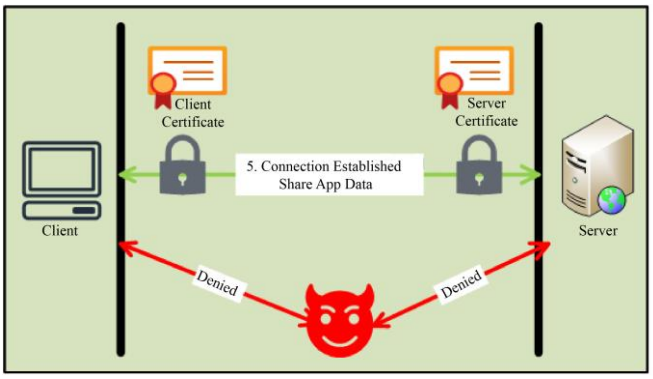

Fig. 7 TLS Certificate list of contents

Figure 7 shows when a malicious actor tries to execute a man-in-the-middle attack that does not get successful because both client and server have the ability to verify the identity of each other using an MTLS-based connection and respective certificates from each side of the connection. The malicious actor does not possess a trusted certificate that is verifiable on either side and, hence, is unable to establish any kind of connection. The bad actor, while executing a man-in-the-middle attack, tries to spoof itself as an authentic server to the client, and on the other hand, it tries to spoof itself as a genuine client to the server, in case of a TLS-based handshake, there is still a chance that the bad actor would be able to convince at least the client of being a legitimate web server, but in case of mTLS even if the client's machine is tricked the server on the other hand will not initiate a connection due to lack of validity of the client. To protect web interactions that typically involve a client requesting resources from a server, TLS is a robust mechanism. However, when protecting API-based interactions between a client and server, mTLS proves to be a much more relevant mechanism. Organizations exposing APIs often either share business data or expose key business logic using those APIs over the Internet, which means in the case of an API based communication, there is often a risk of an untrusted client misusing either the data or business functionality exposed by the API to cause harm to an organization, this is exactly why it is a much safer approach to verify the API consuming client in addition to authenticating them, and that can be made possible by utilizing mTLS. It requires managing certificates on both sides and implementing logic to validate. This may seem more complex, but it is a strong mechanism to implement two-way or zero trust-based communication between client and server.

5. Conclusion

The advanced API protection mechanism examined and described in this study goes way above basic security safeguards when protecting APIs against advanced cyber-attacks. Table 1 below summarises all the mechanisms discussed in the study and can be used by organizations willing to bolster API security as a quick-start security checklist.

Table 1. Advanced security mechanisms checklist

Steps	Description
1	Start with implementing basic WAF as a must.
2	Upgrade eventually to WAAP.
3	Use TLS-based encryption above APIs.
4	Use very strong Cipher suites for TLS.
5	Keep upgrading TLS versions with time.
6	For highly sensitive data APIs, use mTLS.

These checklist steps will significantly help protect APIs and business data. However, API security is a never-ending and always-changing process. While these advanced mechanisms provide a solid foundation and starting point, it is extremely important to enforce strong governance and audit mechanisms to adjust and adapt to the changing security landscape and keep pace with advancing cyber threats daily.

References

[1] Kinza Yasar, Web Application Firewall (WAF), 2023. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/Web-application-firewall-WAF

[2] Ronghua Sun, Qianxun Wang, and Liang Guo, "Research Towards Key Issues of API Security," *CNCERT 2021*, *Communications in Computer and Information Science*, pp. 179-192, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Josué Alejandro Díaz-Rojas et al., "Web API Security Vulnerabilities and Mitigation Mechanisms: A Systematic Mapping Study," *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*, San Diego, CA, USA, pp. 207-218, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Fatima Hussain et al., "Enterprise API Security and GDPR Compliance: Design and Implementation Perspective," *IT Professional*, vol. 22, no. 5, pp. 81-89, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] What Is WAAP, Akamai. [Online]. Available: https://www.akamai.com/glossary/what-is-waap

[6] Web Application and API Protection (WAAP), Imperva A Thales Company. [Online]. Available: https://www.imperva.com/learn/application-security/web-application-and-api-protection-waap/

[7] What Happens in a TLS Handshake SSL Handshake, Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

[8] PCI DSS Quick Reference Guide, PCI Security Standards Council, 2018. [Online]. Available: https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf

[9] Summary of the HIPAA Privacy Rule, US Department of Health and Human Services, 2022. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html

[10] What does the General Data Protection Regulation (GDPR) Govern, Reform of EU Data Protection Rules, European Commission, 2016. [Online]. Available: https://commission.europa.eu/law/law-topic/data-protection/reform/what-does-general-data-protection-regulation-gdpr-govern_en

[11] Dionisie Gitlan, Cipher Suites Explained in Simple Terms: Unlocking the Code, SSL Dragon, 2024. [Online]. Available: https://www.ssldragon.com/blog/cipher-suites/

[12] What is an SSL/TLS Certificate, AWS. [Online]. Available: https://aws.amazon.com/what-is/ssl-certificate/

[13] Why use TLS 1.3? Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ssl/why-use-tls-1.3/

[14] Josh Lake, TLS (SSL) Handshakes Explained, Comparitech, 2023. [Online]. Available: https://www.comparitech.com/blog/information-security/tls-ssl-handshakes-explained/

[15] Arthur Bellore, The TLS Handshake Explained, Autho by Okta, 2023. [Online]. Available: https://auth0.com/blog/the-tls-handshake-explained/

[16] What is Mutual TLS mTLS, Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/

[17] Neil Madden, API Security in Action, Manning Shelter Island, 2020. [Online]. Available: https://cdn.ttgtmedia.com/rms/pdf/bookshelf_apisecurityinaction_excerpt.pdf

[18] OWASP Top 10 API Security Risks, OWASP, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

[19] API7:2023 Server Side Request Forgery, OWASP, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0xa7-server-side-request-forgery/

[20] API8:2023 Security Misconfiguration, OWASP, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0xa8-security-misconfiguration

[21] API9:2023 Improper Inventory Management, OWASP, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0xa9-improper-inventory-management/

[22] API10:2023 Unsafe Consumption of APIs, OWASP, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0xaa-unsafe-consumption-of-apis