🔎 WAF Detection and Evasion 👻 🔥🧱🔥

Intro

The **WAF(Web Application Firewall)** They are security solutions that will filter requests that are made and will look for Payload patterns that could cause a security failure, such as **SQLi, Cross Site Scripting, Cross Site Request Forgery, etc.**

🔎 WAF detection

To know how to evade these Firewalls, the first thing is to know what type of WAF the application we are auditing has. And for that we have the following tool.

🐶WafW00f

Intro

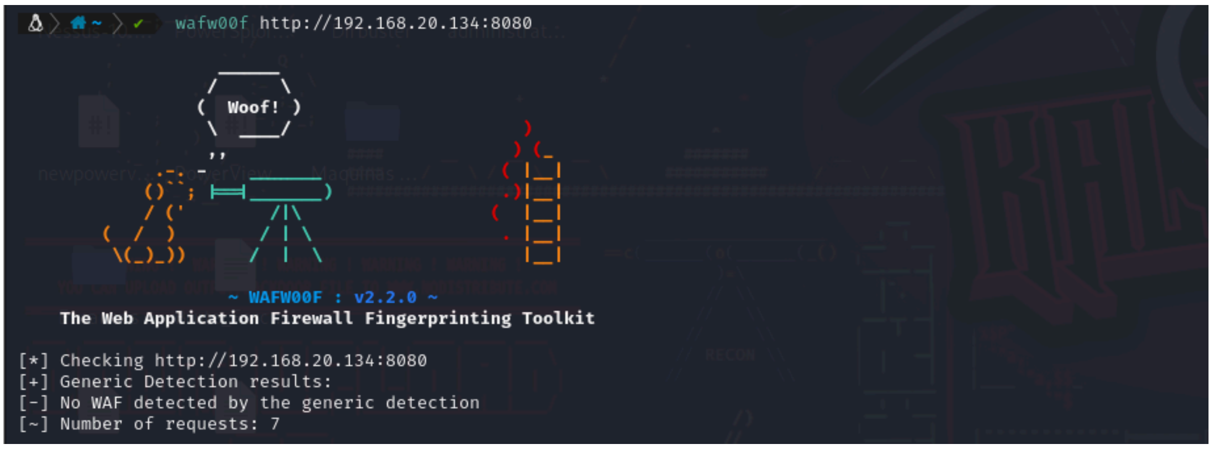
This is one of the most popular tools to know what type of WAF is running a web application. This has implemented the signatures of many WAF providers and based on these it makes the detection.

This comes by default in 🐉Kali and 🦜Parrot systems and will be managed from the terminal.

🐕 wafw00f in action

The use of this tool is very simple, we just have to call it from Terminal and place the website we want to scan

wafw00f http://target.website



```
wafw00f http://192.168.20.134:8080

                ~ WAFW00F : v2.2.0 ~
    The Web Application Firewall Fingerprinting Toolkit

[*] Checking http://192.168.20.134:8080
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7
```

Although the example image has not detected a WAF, the result is presented below the Website Checking

👻🧱 WAF Evasion

Intro

Once we have detected a WAF in a web application, it is important to take into account the fact that ***evading a WAF is not easy.*** Therefore, it would be advisable to **leave that domain aside a little** and start doing subdomain discoveries, either with **subfinder** the **sublist3r**, and scan them to see if ***some*** of these is not protected by a WAF.

Why *some*? Because when a company contracts these external WAF services, it is normal that the inclusion of subdomains is included in those packages, and there are organizations that include their main domains in those packages, but then there are other subdomains or applications that belong to that organization that ***They do not protect them with the WAF*** whether to save costs, have forgotten them, or for any other reason.

That is why it is so important that before considering making a *Bypass* of a WAF, let's do an analysis of all the resources, domains and subdomains that that organization has published on the Internet to find a related objective that is not protected by that WAF. That would be the most satisfying

Now, in the event that we do not find anything that is not protected by the WAF, then we have several alternatives to make a, so to speak, *WAF Bypass.*

👻 WAF Evasion Types

To carry out an evasion, so to speak, manually, what we have to do is try to get the **public IP address** of the website that we are auditing and access said site from it.

That is, instead of accessing the application from the browser via, for example, **http://target.website**, we do it for the public IP of that one, which would be something like, for example http://10.10.10.10.

So *Can* The public IP address of the website that we are auditing is published and we access and do a *WAF Bypass* manually

👻📖🧱 Bypass Firewall By DNS History

Intro
When we want to access a web page protected by a WAF, what we are really interacting with is the IP address of the WAF and once we interact with it, it redirects us to the website.

Now, if we consult the DNS history, it may be possible to obtain an IP address associated with the Domain name that is not listed. **Public IP address before you had the WAF** and it may still be accessible from the internet. And this tool helps us do that.

⚙️ Installation

This tool is not installed by default in 🐉Kali or 🦜Parrot so we have to download it from its GitHub repo

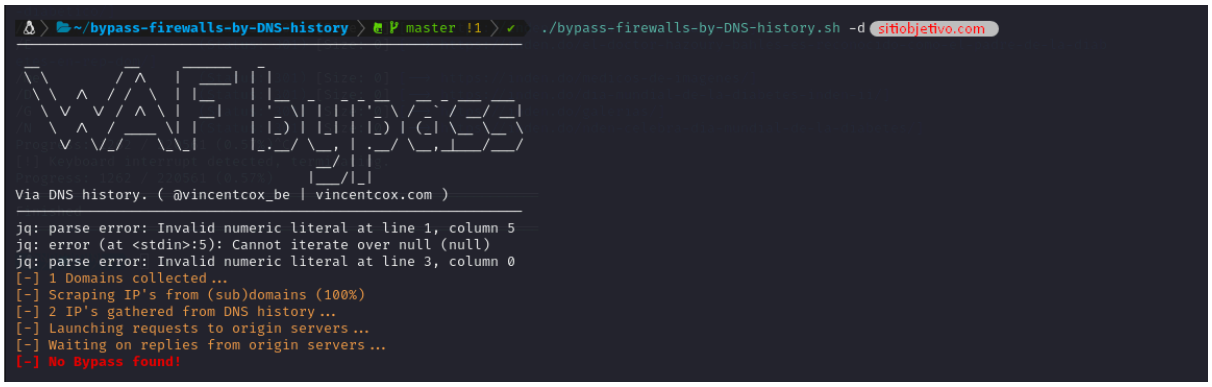
git clone https://github.com/vincentcox/bypass-firewalls-by-DNS-history.git

Access it from wherever we downloaded it and give it permission with **chmod 777** to the file **bypass-firewalls-by-DNS-history.sh**, then do a **sudo apt install jq** which is a dependency we need to run this tool.

And we will launch this tool from the folder found with the command

./bypass-firewalls-by-DNS-history.sh --d target.website

The result would be something like this



```
~/bypass-firewalls-by-DNS-history  master !1  ✓  ./bypass-firewalls-by-DNS-history.sh -d sitiobjetivo.com

Via DNS history. ( @vincentcox_be | vincentcox.com )

jq: parse error: Invalid numeric literal at line 1, column 5
jq: error (at <stdin>:5): Cannot iterate over null (null)
jq: parse error: Invalid numeric literal at line 3, column 0
[-] 1 Domains collected...
[-] Scraping IP's from (sub)domains (100%)
[-] 2 IP's gathered from DNS history...
[-] Launching requests to origin servers...
[-] Waiting on replies from origin servers...
[-] No Bypass found!
```

Note: The website for this example is shaded.

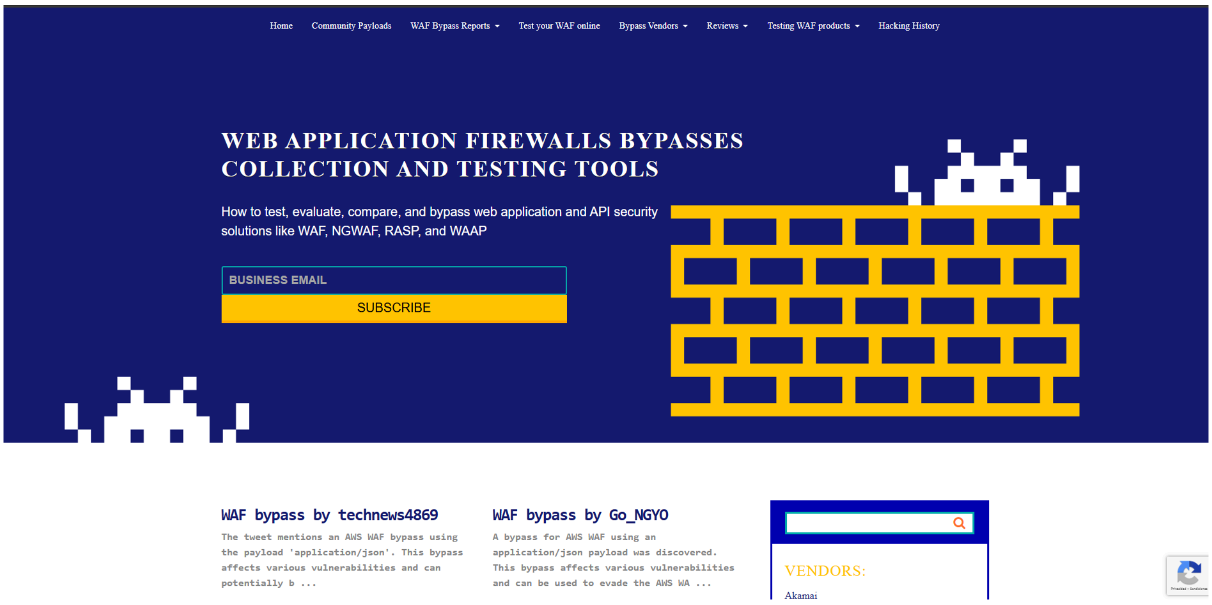
This tool also obtains the IP addresses of some subdomains because some of them, although *are not protected by the WAF*, sometimes **could *be pointing to the same IP address*** of the provincial domain that could be protected by the WAF.

👻📏 Evade the Engine Rules

This technique is very advanced and requires us to be up to date when it comes to web programming and scripting, why? Because this technique basically consists of circumventing the rules engine of a WAF, **encryption and coding payloads** To circumvent the system's WAF rules, which we remember consists of comparing codes that are in its rules to define what can interact with the web application and what cannot.

For this reason we have to be very up to date, because a Payload that we have manipulated, encoded, encrypted or whatever could **TODAY** circumvent that WAF and in a week it has already been patched by the company providing said WAF.

Something that can make our lives a little easier in this is to use tools like waf-bypass.com where current bypasses are published to evade WAFs specifically with techniques that will not help to evade that rules engine.



WEB APPLICATION FIREWALLS BYPASSES COLLECTION AND TESTING TOOLS

How to test, evaluate, compare, and bypass web application and API security solutions like WAF, NGWAF, RASP, and WAAP

WAF bypass by technews4869

The tweet mentions an AWS WAF bypass using the payload 'application/json'. This bypass affects various vulnerabilities and can potentially b ...

WAF bypass by Go_NGYO

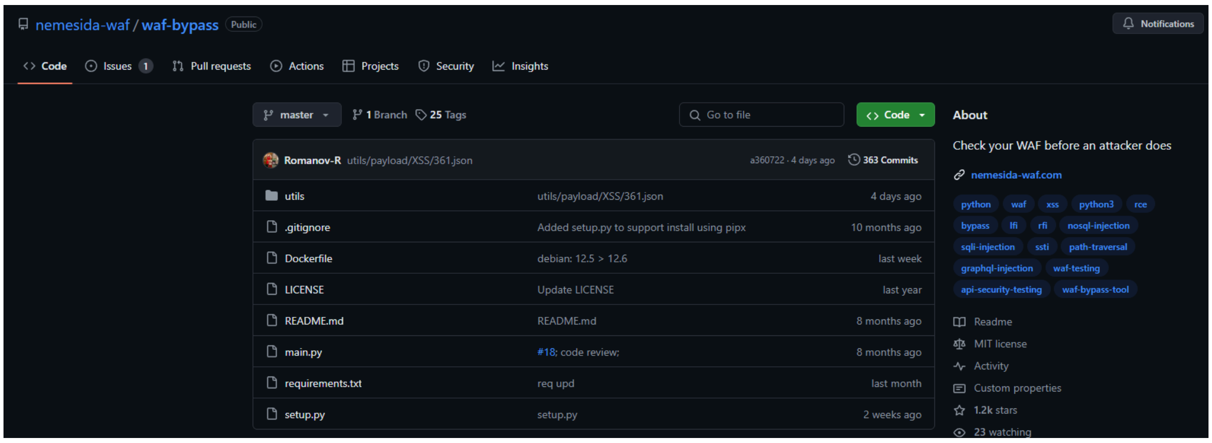
A bypass for AWS WAF using an application/json payload was discovered. This bypass affects various vulnerabilities and can be used to evade the AWS WA ...

VENDORS:

Akamai

The idea is to go to websites where these Bypasses are shared, but having my survey is quite complicated and they usually patch them very quickly.

And we also have another way and that is to rummage through the Github repositories where we can find one or another tool like **Waff Bypass Test Tool By Nemesida.**



Romanov-R utils/payload/XSS/361.json a360722 · 4 days ago 363 Commits

utils	utils/payload/XSS/361.json	4 days ago
.gitignore	Added setup.py to support install using pipx	10 months ago
Dockerfile	debian: 12.5 > 12.6	last week
LICENSE	Update LICENSE	last year
README.md	README.md	8 months ago
main.py	#18; code review;	8 months ago
requirements.txt	req upd	last month
setup.py	setup.py	2 weeks ago

About

Check your WAF before an attacker does

nemesida-waf.com

python waf xss python3 rce bypass lfi rfi nosql-injection sqli-injection ssti path-traversal graphql-injection waf-testing api-security-testing waf-bypass-tool

Readme
MIT license
Activity
Custom properties
1.2k stars
23 watching

But keep in mind that these tools generate a lot of traffic and are very intrusive and in these cases it is best to have fun if it is prudent to use tools of this type and notify the organization that we are carrying out the audit that we are going to use this type of tools

Conclusion: The truth is that evading a WAF is just as complicated, if not more so, than evading an antivirus, EDR, or IDS. And these techniques and tools are designed ***to try*** to evade this security tool.