

Forum:[Hack Forums](#)

Thread: [\[TUTORIAL\] Non-conventional WAF/IDS evasion techniques \[PART 1\]](#)

M L T - December 10, 2018, 6:07 am

Non-Conventional WAF/IDS Evasaion techniques - Part 1:

Introduction:

This is a tutorial detailing various non-conventional methods of circumventing signature based WAF or IDS software. Rather than focusing on signature evasion, and bypassing of blacklisted characters or keywords, this will instead be focusing on techniques that can allow an attacker to completely prevent the checks in place within the WAF from being applied at all. Using these techniques means that an attacker no longer has to focus on crafting a payload that slips through any signature-based checks in place, but instead they can sidestep the entire bypass process altogether, rendering the WAF ineffective.

Although it is certainly useful to have an understanding of the methodologies used for crafting payloads that circumvent WAF implementations, there are some simple methods that can be used to skip this process entirely. I will give an overview of each of these techniques, and explain how they can be used individually or in combination to skip the filter evasion process.

WAF - Web Application Firewall

IDS - Intrusion Detection System

Initiating connection via direct IP access:

This is a well-known and well-documented method for bypassing cloud-based WAF solutions, but I will be covering it anyway (along with some variations of it) — Cloud-based WAF solutions are tied to the DNS of the website, meaning that if an attacker loads the URL via its direct IP address, the entire process of having to find a payload which bypasses their signature checks is skipped. You would just access the vulnerable URL containing your payload, but with the direct IP address of the server in place of the DNS.

For example, imagine there is an RCE bug affecting example.com:

Quote:[http://example.com/vuln/whois.php?url=http://google.com/%26cat\\${IFS}/etc/passwd](http://example.com/vuln/whois.php?url=http://google.com/%26cat${IFS}/etc/passwd)

let's assume that there is a cloud-based WAF in place, which is blocking the user each time they make this request. An attacker could send a ping request in order to get the backend IP address for example.com:

Quote: Pinging example.com [93.184.216.34] with 32 bytes of data:

Reply from 93.184.216.34: bytes=32 time=94ms TTL=50

Ping statistics for 93.184.216.34:

Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 94ms, Maximum = 94ms, Average = 94ms

Once the IP address has been obtained, the URL could be accessed like so:

Quote:[http://93.184.216.34/vuln/whois.php?url=http://google.com/%26cat\\${IFS}/etc/passwd](http://93.184.216.34/vuln/whois.php?url=http://google.com/%26cat${IFS}/etc/passwd)

If working as intended, then this request will not trigger a WAF response and will instead output the contents of example.com's passwd file.

protip: rather than entering the IP address in place of the URL each time you make a request, you can just make an edit to your hosts file (/etc/hosts on Linux) in order to speed up the process.

If you can find the backend IP address through means of a simple Ping request or something along those lines, then you're lucky. Generally, more complex techniques will be required in order to reveal the IP. This is because with CDN's and DDoS protection solutions that also offer WAF protection (CloudFlare being a prime example) will display the IP address of their servers as the ping response, rather than revealing the true IP address of the website. It's worth noting that if we're discussing CloudFlare specifically, websites already exist (such as CrimeFlare) that allow people to query the DNS against a list of backend IP addresses associated with such websites, if you're lucky then the correct IP will already be stored in one of these databases and viewable through these free services. Below I will explain a few different methods that can be used to leak the correct backend IP address of websites using cloud-based protections like this.

Some basic recon is probably the best way to go about revealing the backend IP of a website. Historic records can reveal the backend IP, so try inputting the URL to services such as Netcraft. Misconfigurations are also often common in services like this, so that's always worth checking for. Sure, their main DNS is routed through CloudFlare and you can't get the backend IP, but what about subdomains? There's a chance that some of these subdomains are pointed to the same server as the main website, and maybe someone forgot to route them through CloudFlare's nameservers. Similar checks should be done for FTP servers, MX servers, and so on. It is also worth checking the likes of Shodan and Censys to see if any records expose the origin server of the domain.

If all reconnaissance techniques have been exhausted and the backend IP address is yet to be revealed, then you need to figure out a way to make the target server initiate a connection to a page on a server of your own, causing the IP address to end up in your access_log — there are a few different viable ways of doing this. mod_rewrite can be used with .htaccess in order to reveal it (I will be explaining this trick in my .htaccess exploitation series)

Ideally, you want to be finding some way of making the server initiate a direct connection to you. This could be through means of an SSRF Vulnerability (y'know, like those SSRF with "no impact" that allow you only to make the request and almost nothing else) or it could be intended functionality of the site that will cause a direct connection to be made. For example, if the site is a forum and users have a profile with a customizable avatar image which can be changed to a new image which is submitted via a remote URL, then you could host an image on a server belonging to you, request that image from the target website and grep your access_log for the IP. Another similar technique would be if the website had an email newsletter or something similar. You could subscribe to it, then check the mail headers when you receive an email from them in order to get the IP address associated with their mail server.

Switching up protocols and nameservers:

In addition to changing the IP address, techniques such as loading the page over https when it loads through http by default (or vice versa)- sounds ridiculous, but this will prevent some WAF's from applying their blocks. Try accessing things over a different available port, and specifying the port within the URL, for example:

Quote:<https://example.com:443/vuln.php?payload=../../../../etc/passwd>

You can also add or remove 'www' to or from the URL and this can achieve the same result (this particular method allowed me to get a working XSS on eBay's main domain) — or if the target has mirrors setup for server load balancing (like when you see 'www2', 'www3', and so on), then load one of the mirrors (e.g. change 'www' to 'www2') as sometimes the WAF will not be applied to these.

One often overlooked method to bypass a WAF is to send your request over IPv6 rather than IPv4 — there are many Intrusion Detection Systems in place which also only monitor IPv4 traffic whereas any IPv6 traffic can slip through undetected.

Manipulation of HTTP Headers:

There has been cases where certain HTTP headers can be manipulated in order to trick a WAF into believing it is talking to either:

- A Whitelisted Host

- Another machine on the same network

A user can set the value of specific headers to localhost (127.0.0.1), local addresses (for example 192.168.1.2), or the remote IP of the target site. The following headers can be manipulated in this manner:

- X-forwarded-for

- X-remote-IP

- X-originating-IP

- X-remote-addr

- X-client-IP

So for example, if a HTTP request was sent with the following header value:

Quote:X-Originating-IP: 127.0.0.1

The WAF would be tricked into thinking it was communicating with itself, and would therefore treat the HTTP request as if it were whitelisted. I first saw this method documented by Jason Haddix, although I'm not certain as to whether he was responsible for finding it.

Manipulation of the content-type header with each sent request can also result in a bypass. This can be achieved in a number of ways. This value can be stripped from requests, malformed through means of modified characters, or spoofed to another content type, for example, take the following content-type:

Quote:Content-Type: text/html

Here are a few examples of how it could be modified:

- Content-Type: (Removing content-type value)

- Content-Type: text/htmlzzzzzzzzzz (Malforming content-type value)

- Content-Type: application/octet-stream (Modifying content-type value)

There has also been cases where sending a different valid content-type with each HTTP request can result in a bypass, like so:

- Content-Type: application/octet-stream (request #1)

- Content-Type: text/html (request #2)

- Content-Type: text/xml (request #3)

I'm not sure why this one works exactly, but sending a varying content type alongside each HTTP

request has confused WAF's for me in the past and got the exact end-goal that I was looking for.

Setting the MIME type to multipart/form data then malforming the request has resulted in bypass for many commercial WAF in the past, including mod_security, this can be done like so (these examples are from Ivan Ristic's Blackhat USA talk):Content-Type: multipart/form-data ; boundary=0000

Content-Type: mUltiPart/ForM-dATa; boundary=0000

Content-Type: multipart/form-datax; boundary=0000

Content-Type: multipart/form-data, boundary=0000

Content-Type: multipart/form-data boundary=0000

Content-Type: multipart/whatever; boundary=0000

Content-Type: multipart/; boundary=0000

The HTTP Host Header can also be manipulated in some cases to attempt to circumvent WAF software. Sometimes simply adding a dot to the host value can result in a bypass. Removing or modifying the host header can result in a bypass — sometimes the WAF will choose which security policies to implement based upon the hostname. If the hostname is not recognized (due to manipulation from the user), then in some cases the WAF will not be applied at all. Some WAF's will stop monitoring requests (for performance reasons) after being tricked into believing that the requests aren't intended for that site due to discrepancies in the host header, but then further requests can be submitted using the correct hostname without being monitored whatsoever.

pid999 - December 10, 2018, 1:57 pm

The following headers can be manipulated in this manner:

X-forwarded-for
X-remote-IP
X-originating-IP
X-remote-addr
X-client-IP
Spoiler ([Click to View](#))
Code +
X-Online-Host
:D

M L T - December 10, 2018, 2:23 pm

(Dec 10, 2018 02:57 PM)pid999 Wrote: [\[CITE\]](#) The following headers can be manipulated in this manner:

X-forwarded-for
X-remote-IP
X-originating-IP
X-remote-addr
X-client-IP
Spoiler ([Click to View](#))
Code +
X-Online-Host
:D

Ahh nice. I wasn't aware of that one :)

pid999 - December 10, 2018, 3:15 pm

(Dec 10, 2018 03:23 PM)M L T Wrote: [\[CITE\]](#) Ahh nice. I wasn't aware of that one :)
.: just ma favorite 4 injecting header.
nice tread.
?
?

Novad0x - December 11, 2018, 9:39 am

Another excellent thread, looking forward to the rest if the series.

CyberStrike - December 11, 2018, 9:38 pm

For sure it's a good read, not typical of the average. Well done.

hacker1 - February 16, 2019, 1:56 pm

nice tutorial.thanks.do you have youtube link for this?

hacker1 - February 17, 2019, 2:11 pm

thanks for the tutorial.where can I find more tutorials

Totes Mcgoats - February 20, 2019, 10:55 am

Do you have more information on how this would work?