29th September, 2018

Web application firewalls: Analysis of OWASP modsecurity core rule set and detection logic

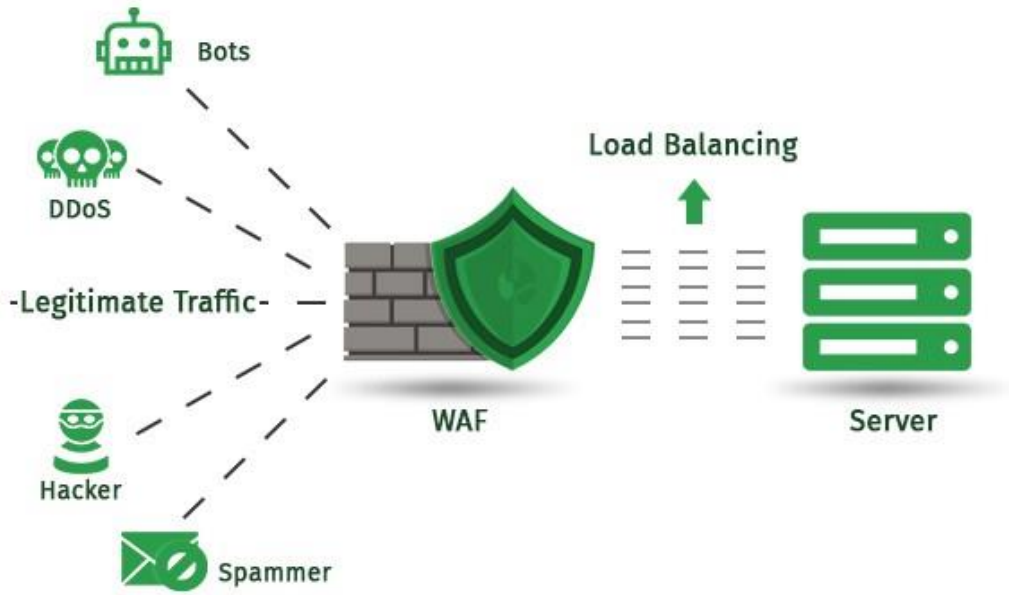
Josna Joseph

Agenda

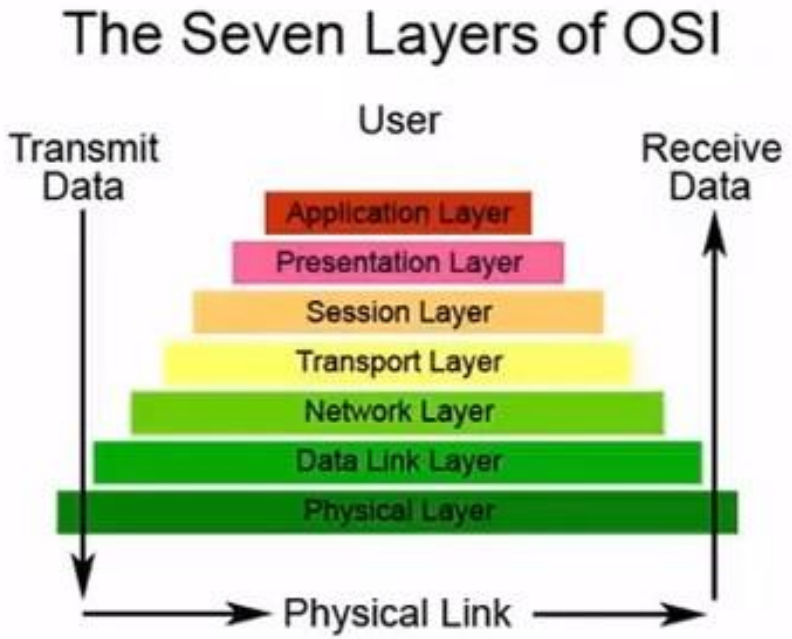
- What is a Web Application firewall?
- Operation modes & workflow
- Detection Logic
- What is Core rule set?
- Anomaly scoring/Thresholds
- Paranoia levels
- Handling of false positive

What is WAF?

“A WAF may be a hardware/software that sits between the Client and Web application in order to determine/block malicious requests before they are sent to the web application”.



Layers of Insecurity



Modes: Passive or Reactive?

Passive:

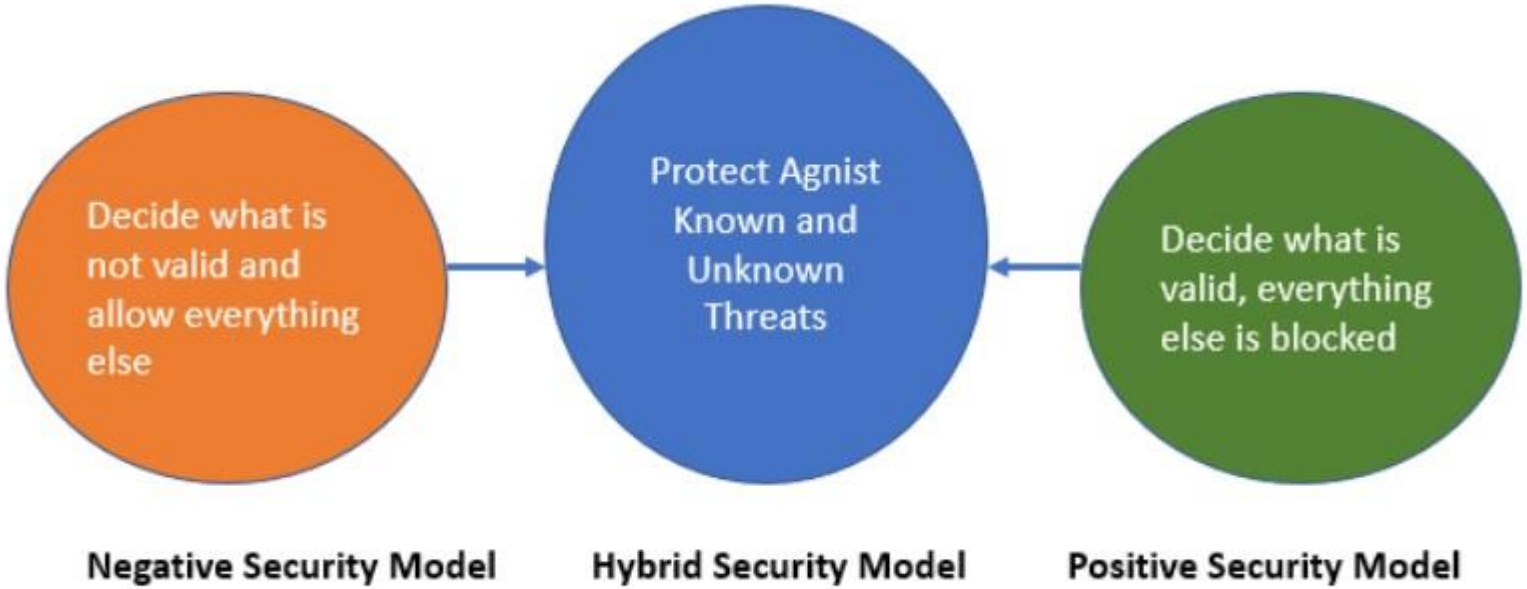
- WAF will act like as ID.
- Used during the first days, to prevent real users being blocked by false positives

Reactive:

- It will block the malicious requests.
- In production environments most WAF systems runs in reactive mode

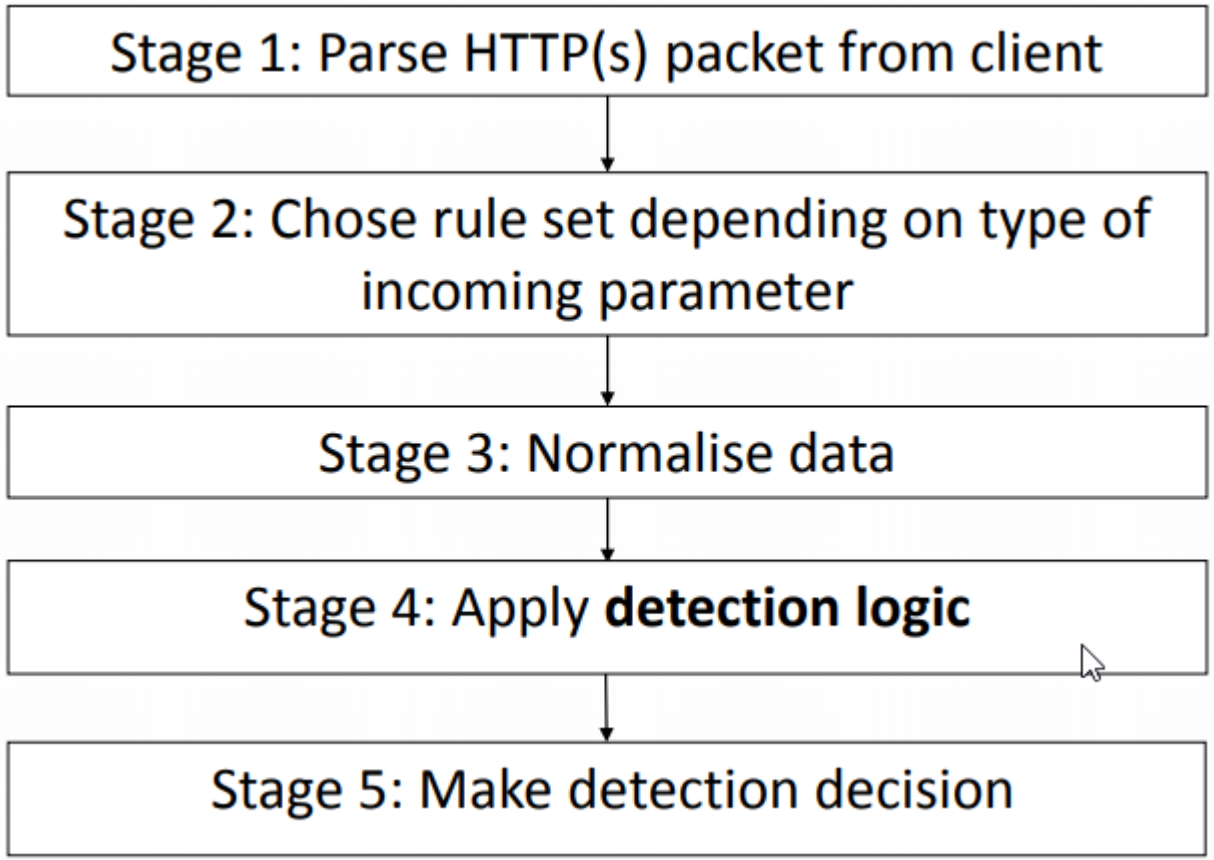
Operation models and work flow

- Negative model (black list based)
- Positive model (white list based)
- Hybrid model (mix positive and negative model protection)

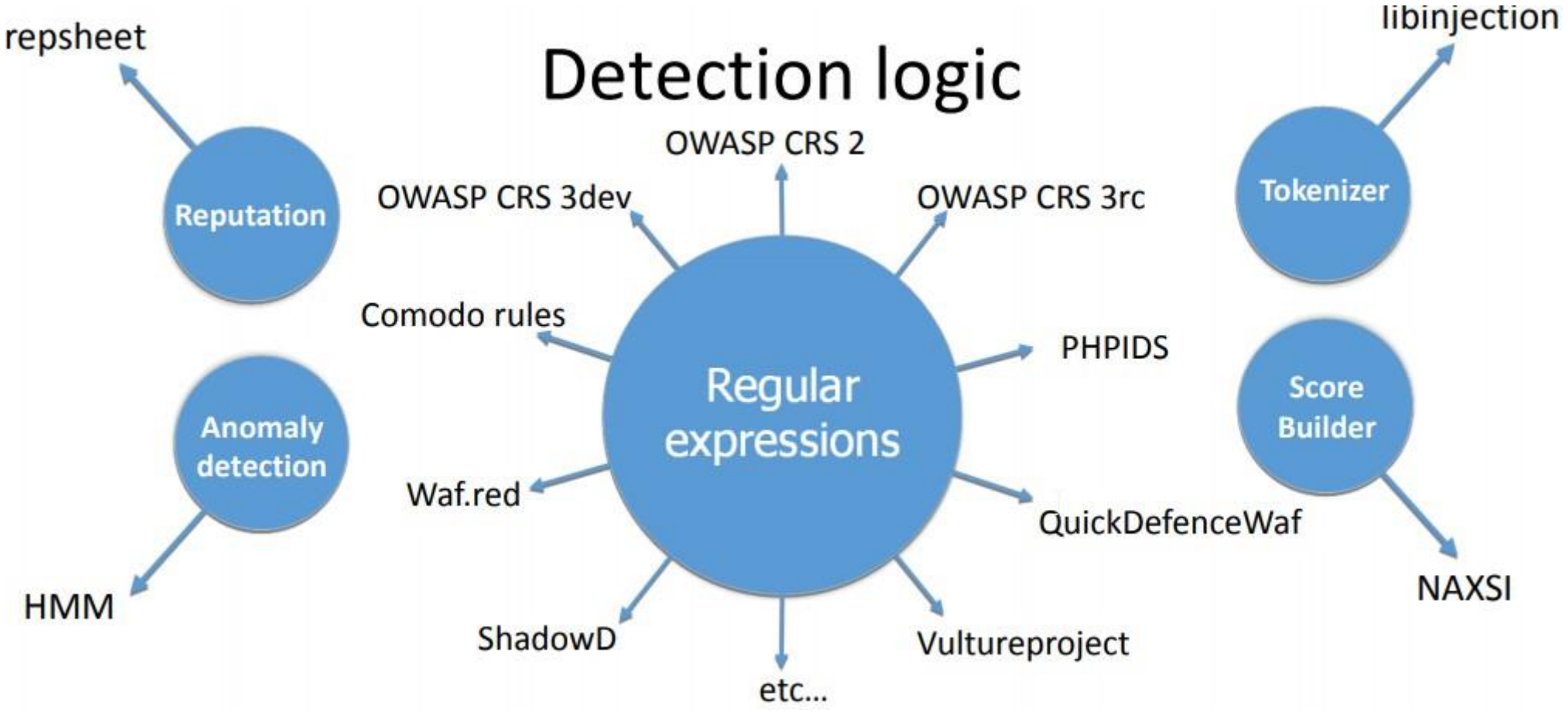

Decide what is not valid and allow everything else
Protect Agnist Known and Unknown Threats
Decide what is valid, everything else is blocked
Negative Security Model
Hybrid Security Model
Positive Security Model


THE BLACKLIST

Workflow

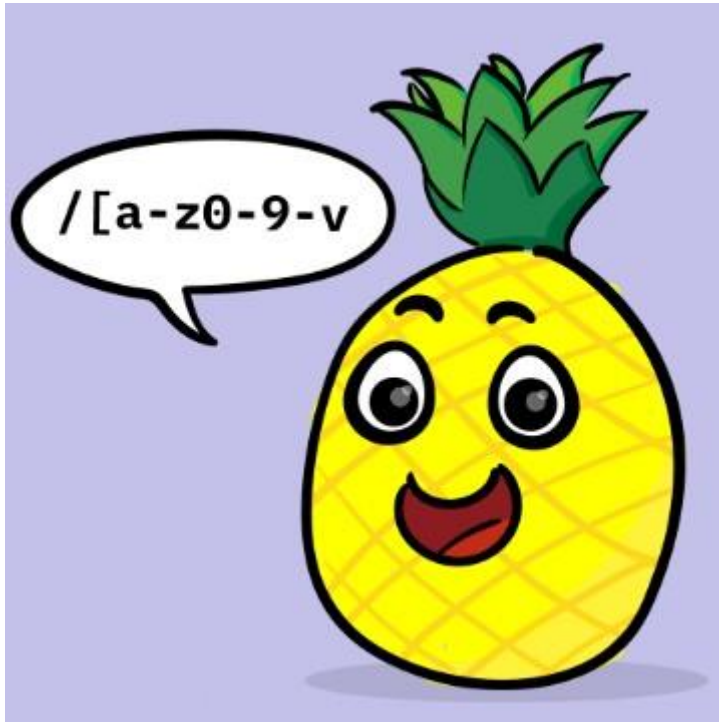

Stage 1: Parse HTTP(s) packet from client
Stage 2: Chose rule set depending on type of incoming parameter
Stage 3: Normalise data
Stage 4: Apply **detection logic**
Stage 5: Make detection decision

How detection logic works?



/^[Reg]ular[Ex]pressions$/

Sequence of symbols and characters expressing a string or pattern to be searched for within a longer piece of text.



Weak places in regular expressions

- Regexp should be case-insensitive (?i:)
- It is possible to bypass regex using upper or lower cases in words

Vulnerable regex example: http

Bypass example: hTtP

Weak places in regular expressions

- Number of repetitions of set or group **{}** should be carefully used,
- It is possible to bypass such limitation by lowering or increasing specified numbers.

Vulnerable regex example: a{1,5}

Bypass example: aaaaaa (6 times)

Weak places in regular expressions

- Regexp should not be vulnerable to ReDoS.
- Produces a denial-of-service by providing a regular expression that takes a very long time to evaluate.

Vulnerable regex example: (a+)+

Bypass example: aaaaaaaaaaaaaaaaaaa!

Weak places in regular expressions

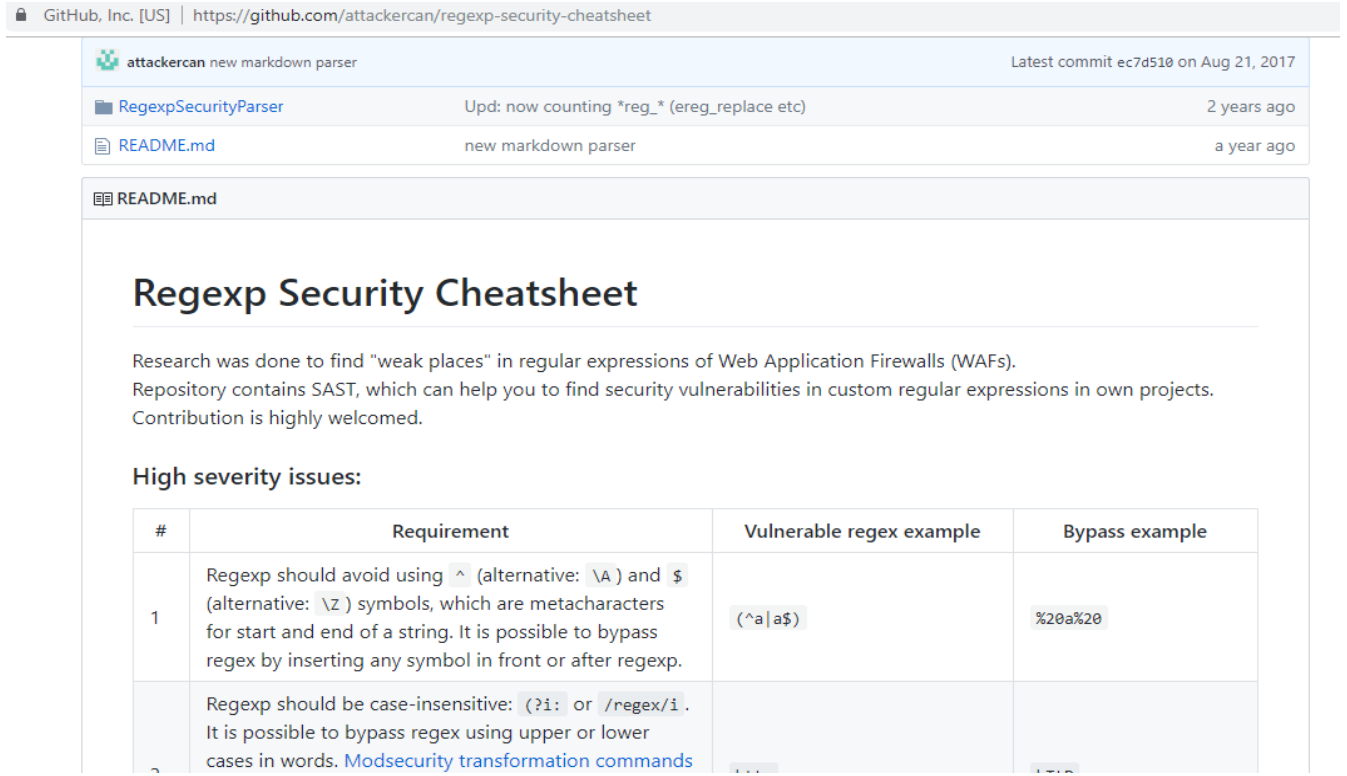
- Usage of the wrong syntax in POSIX character classes

Vulnerable regex example: a[digit]b

Bypass example: aab

Regular expressions: Security cheatsheet

https://github.com/attackercan/regexp-security-cheatsheet



GitHub, Inc. [US] | https://github.com/attackercan/regexp-security-cheatsheet

attackercan new markdown parser — Latest commit ec7d510 on Aug 21, 2017

RegexpSecurityParser	Upd: now counting *reg_* (ereg_replace etc)	2 years ago
README.md	new markdown parser	a year ago

README.md

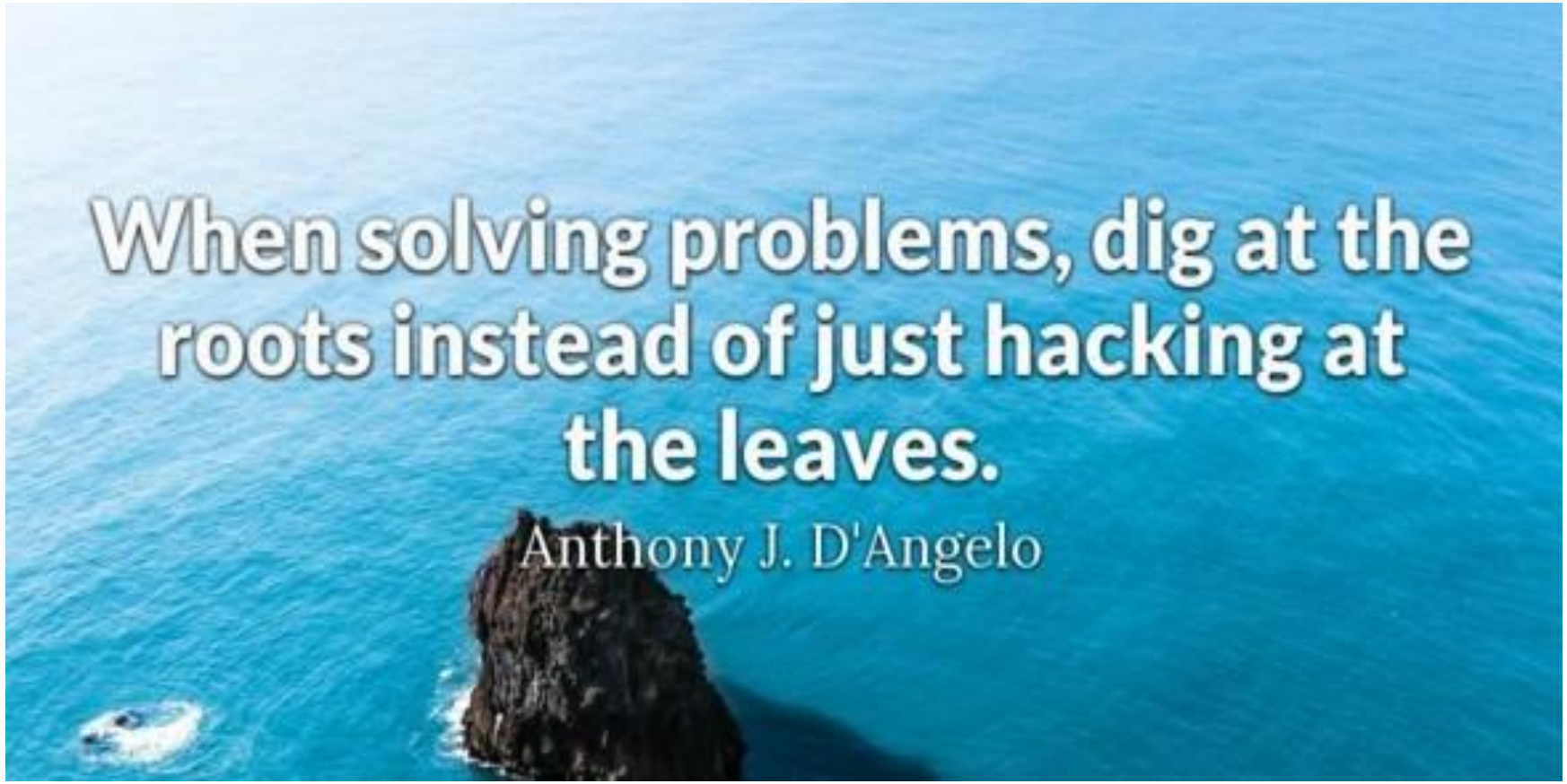
Regexp Security Cheatsheet

Research was done to find "weak places" in regular expressions of Web Application Firewalls (WAFs).
Repository contains SAST, which can help you to find security vulnerabilities in custom regular expressions in own projects.
Contribution is highly welcomed.

High severity issues:

#	Requirement	Vulnerable regex example	Bypass example
1	Regexp should avoid using `^` (alternative: `\A`) and `$` (alternative: `\Z`) symbols, which are metacharacters for start and end of a string. It is possible to bypass regex by inserting any symbol in front or after regexp.	`(^a|a$)`	`%20a%20`
2	Regexp should be case-insensitive: `(?i:` or `/regex/i`. It is possible to bypass regex using upper or lower cases in words. Modsecurity transformation commands		

Fingerprinting WAF



Tricks to detect WAF

- Cookie values
- HTTP Response code.
- Connection close
- Server cloaking

Cookies

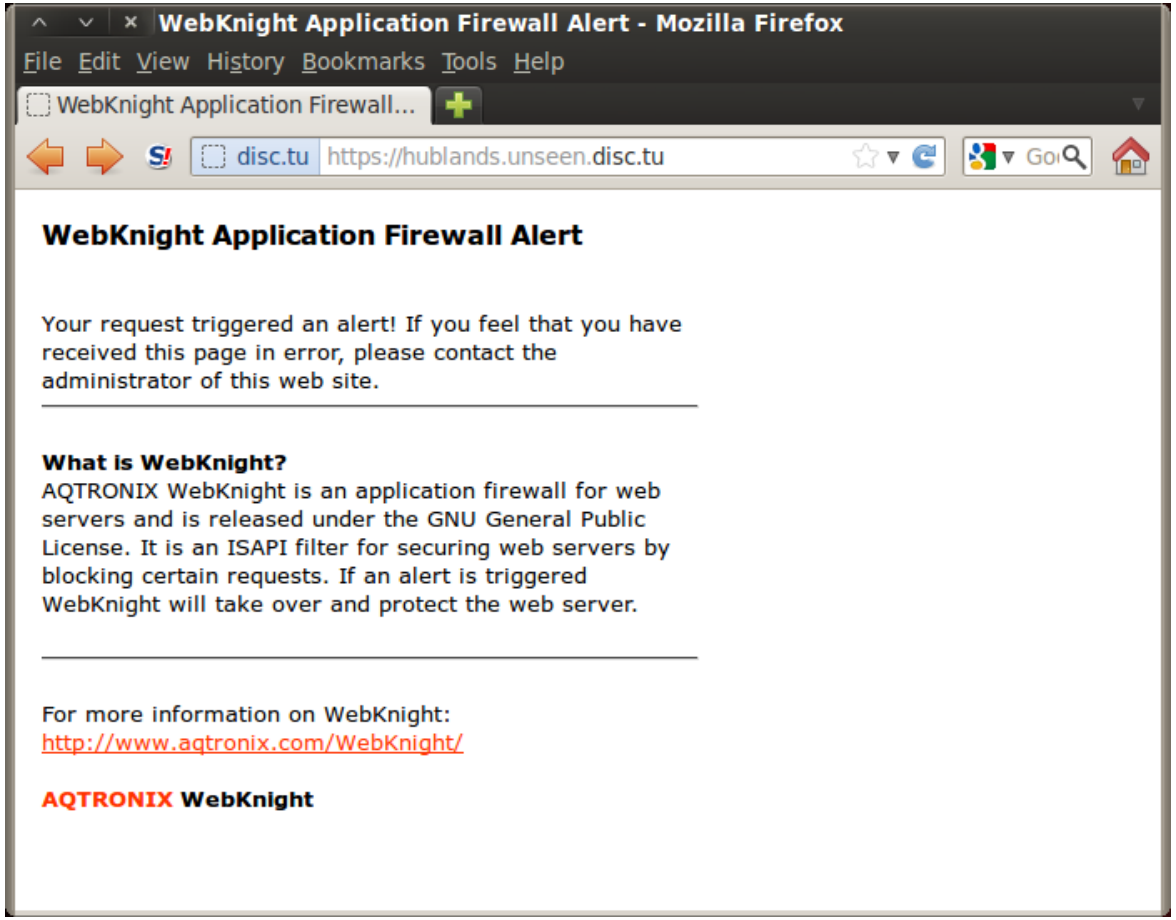
Example – Citrix Netscaler:

```
GET /news.asp?PageId=254 HTTP/1.1
Host: www.SomeSite.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.12)
Accept: image/png,*/*;q=0.5
Accept-Encoding: gzip,deflate
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://www.SomeSite.com
Cookie:ASPSESSIONCWKSPSVLTF=OUESYHFAPQLFMNBTKJHGQGXM;
ns_af=xL9sPs2RIJMF5GhtbxSnol+xU0uSx;
ns_af_.SomeSite.com_%2F_wat=KXMhOJ7DvSHNDkBAHDwMSNsFHMSFHEmSr?nmEkaen19mlrw
Bio1/lsrzV810C&
```

HTTP Response: Dot Defender

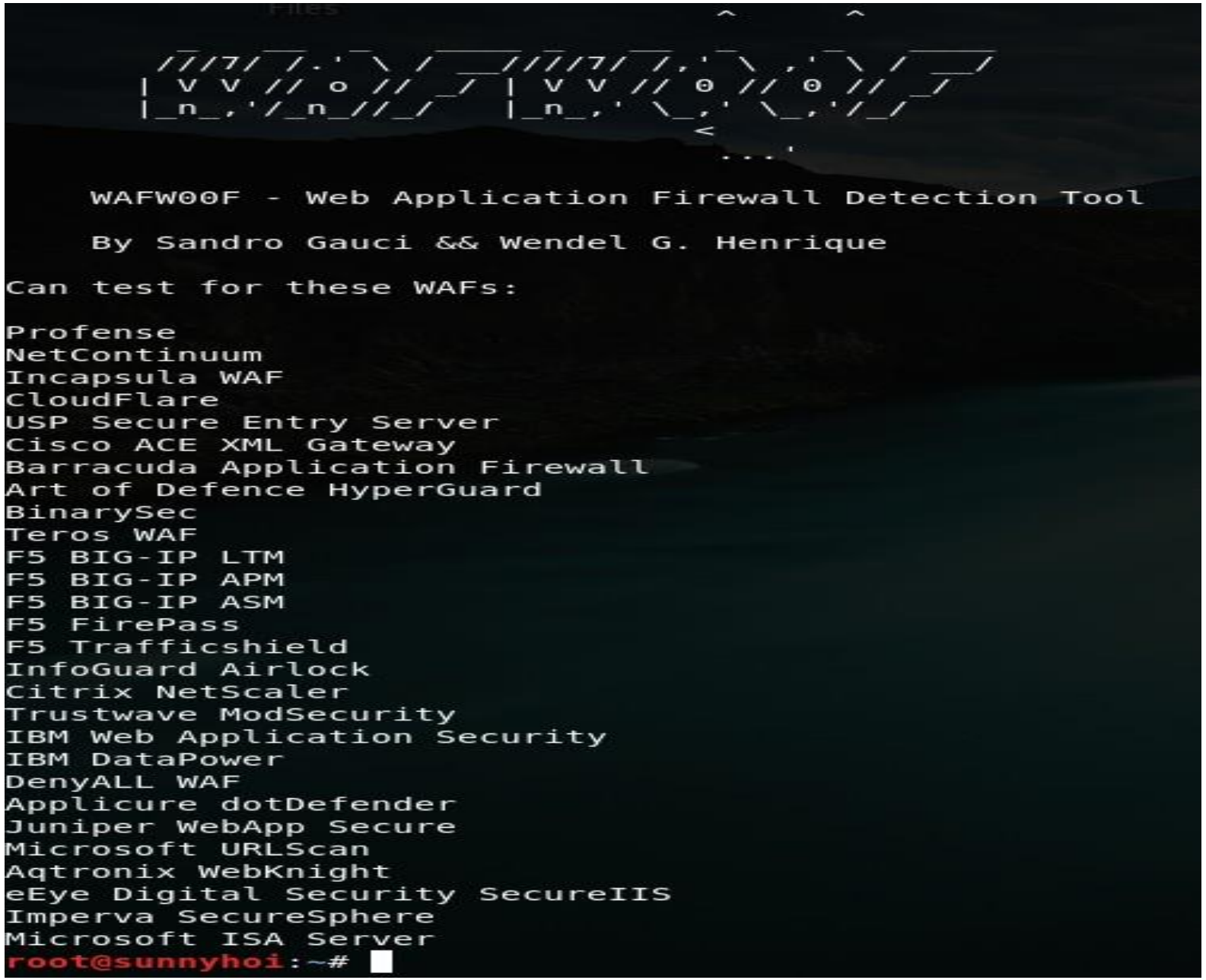
Dot Defender [HTTP Response Codes]

HTTP Response: WebKnight



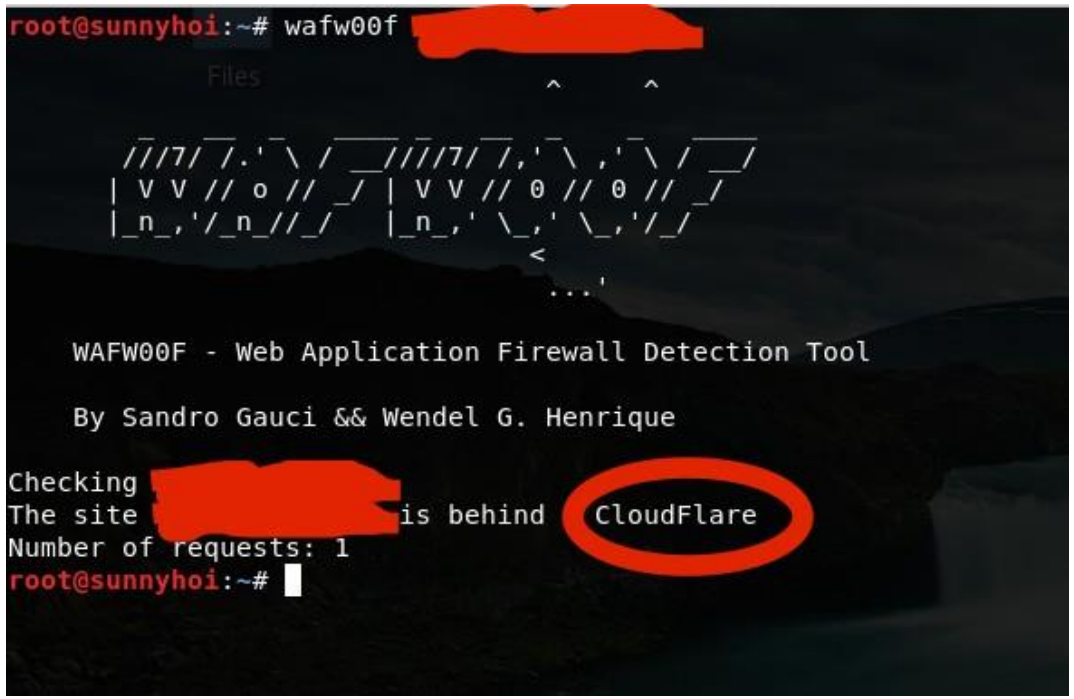
Profiling WAF using WAFw00f.py

WAFW00F identifies and fingerprints Web Application Firewall (WAF) products.


WAFW00F - Web Application Firewall Detection Tool
By Sandro Gauci && Wendel G. Henrique
Can test for these WAFs:
Profense
NetContinuum
Incapsula WAF
CloudFlare
USP Secure Entry Server
Cisco ACE XML Gateway
Barracuda Application Firewall
Art of Defence HyperGuard
BinarySec
Teros WAF
F5 BIG-IP LTM
F5 BIG-IP APM
F5 BIG-IP ASM
F5 FirePass
F5 Trafficshield
InfoGuard Airlock
Citrix NetScaler
Trustwave ModSecurity
IBM Web Application Security
IBM DataPower
DenyALL WAF
Applicure dotDefender
Juniper WebApp Secure
Microsoft URLScan
Aqtronix WebKnight
eEye Digital Security SecureIIS
Imperva SecureSphere
Microsoft ISA Server
root@sunnyhoi:~#

Profiling WAF using WAFw00f.py

wafw00f http://www.targetsite.com

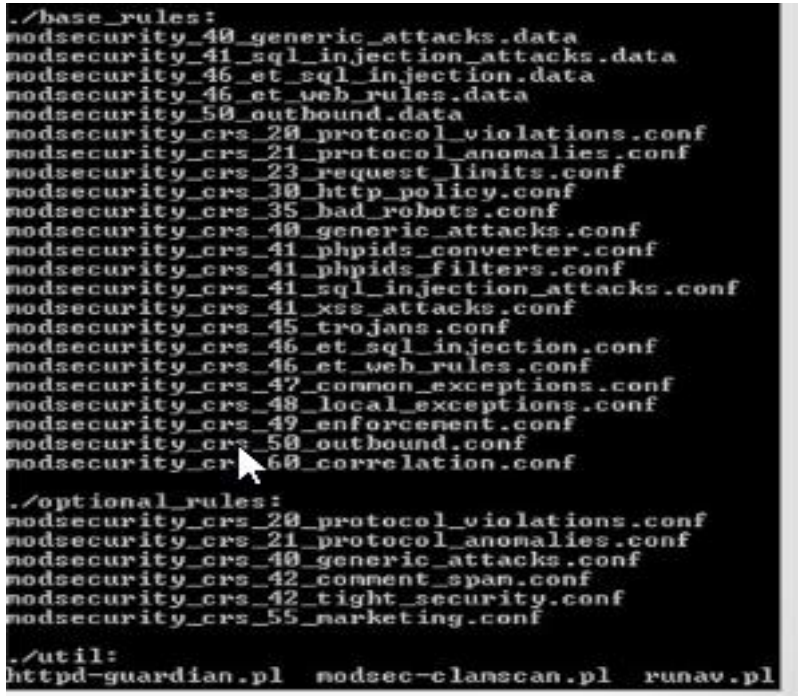


OWASP ModSecurity Core Rule Set

THE 1ST LINE OF DEFENSE

OWASP CRS

A set of generic attack detection rules for use with ModSecurity or compatible web application firewalls.



Installation

- Clone the repository

  $> git clone

  https://github.com/SpiderLabs/owasp-modsecurity-crs

- Copy the example config:

  $> cp crs-setup.conf.example crs-setup.conf

- Include in server config(depends on path)

  Include /etc/httpd/modsec.d/owasp-modsecurity-crs/crs-setup.conf

  Include /etc/httpd/modsec.d/owasp-modsecurity-crs/rules/*.conf


Anomaly Scoring
Adjustable Limit • False Positives

Anomaly scoring

- In version 2.x of the CRS, OWASP introduced the concept of anomaly scoring as a better way to detect attacks more accurately.

- Each rule is built in such a way that it only holds one piece of the puzzle and is assigned a score.


<script>
(973331)
Alert
(973300)
<script
(958051)
Alert(
(958052)

- WAF parses a request through the multiple WAF rules .
- it keeps track of the rules that fire and adds the score of each rule to compute the total anomaly score for a request.
- The WAF will then compare the request anomaly score with an inbound risk score rule threshold.
- If the score exceeded, the request is more likely to be malicious,
- Otherwise the request is judged to be safe.

Paranoia levels:

- The Paranoia Level (PL) setting in crs-setup.conf allows us to choose the desired level of rule checks.
- Can adjust the Paranoia Level in the configuration file(crs-setup.conf).
- With each paranoia level increase, the CRS enables additional rules.
- Higher paranoia levels also increase the possibility of blocking some legitimate traffic

Paranoia levels:

Paranoia Level 1: Minimal amount of False Positives
Basic security

Paranoia Level 2: More rules, fair amount of FPs
Elevated security level

Paranoia Level 3: Specialised rules, more FPs
Online banking level security

Paranoia Level 4: Insane rules, lots of FPs
Nuclear power plant level security

Handling of false positives

- Fight FP’s with Rule Exclusions.

  https://www.netnea.com

Q&A


Which of the following is the largest?
A: A Peanut
B: An Elephant
C: The Moon
D: A Kettle


Thank you!