IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 7, July 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

Impact of AWS WAF Rules Complexity on Web Capacity Units and Financial Costs

Manish Sinha

Senior Software Engineer, Meta Inc (Facebook), San Jose, California, United States

ABSTRACT: In the field of highly scalable and highly available architectures, malicious actors pose a significant threat to the integrity and security of sensitive data, which can include customer data or company's proprietary information. When using Amazon Web Services, it is possible to use Web Application Firewall to craft security rules to protect against the most common threats.

This paper explores the cost, performance and efficiency of the system based on the complexity of the firewall rules being implemented. Web Capacity Unit (WCU) is the measure of capacity consumed for executing a firewall rule and it corresponds to complexity. Higher WCU correspond to lower performance and higher cost. We will primarily discuss how the complexity of a WAF rule will affect WCU.

KEYWORDS: Cloud Computing, Cost analysis, Cybersecurity, Firewall, Performance Analysis.

I. INTRODUCTION

Amazon Web Services is the world's most popular cloud computing platform as per HG Insights, which corresponds to "50.1% amongst the top 10 cloud providers" [1]. With widespread use of AWS, there is a need for a firewall solution. AWS has an offering called "Web Application Firewall", which is based on the traditional firewall architecture. Typical firewalls like iptables and netfilter are primarily used on L4 level but can work on L7 network layer too, whereas Web Application Firewalls primarily operate on L7 level of networking stack [2].

While we might want to run firewalls on L7 layers using AWS WAF, broadly L7 firewalls come at a steep price, which can be determinant to the overall performance of the service, including higher latency and higher computational cost [3]. This means that we need to be aware of the impact we would have by adding, updating or removing AWS WAF rules which protect a web service.

There is no correct answer for the count or complexity of AWS WAF rules for a web service since it would depend primarily on the business requirements and the acceptable budget tolerance for a business. This also means that security and performance can be at odds with each other. Having no WAF rules would be cheapest of them all but would not provide any protection from malicious actors. On the other side having WAF rules to protect the service from all kinds of threats would end up being very cost prohibitive and at the same time would come at a steep price to latency [4].

This paper will start by explaining some related work and why this paper is required. It will be followed by explaining how AWS WAF rules are written. We will then proceed to give a quick outline of Web Capacity Unit (WCU). We will then provide four different kinds of WAF rules, starting from lowest complexity till highest complexity, while explaining what each of the rules are trying to achieve. Then, we will explain how we will be computing the WCU for each rule. We will properly plot the findings so that the results can be visualized briefly. Finally, we will end the paper by summarizing the conclusions.

II. RELATED WORK

There is very little prior research on the effect of WAF rule complexity on the WCU. Vast majority of related work focuses primarily on the usage of WAF in securing cloud-based architectures [5]. On a very high level, Amazon provides guiding principles called AWS Well-Architected Framework [6], for building cloud architectures, which includes WAF, but not specific to it. Kareem, Mohammed discussed the usage of AWS WAF to prevent SQL

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

injection attacks [7], but it is missing the impact of such WAF rules on the performance of the overall latency or cost. Lakhno, V et al. covers the usage of WAF to protect internal services in zero trust architecture [8], but their study was not focused on AWS WAF, but broadly the concept of using WAF for specific results.

The concept of performance analysis being at odds with security is not new and for a reference case, is covered in "Performance analysis of encryption algorithms for security" by M. Panda [9] which relates to how different cryptographic solutions consume significant number of resources, which makes it important to understand the relationship to make informed decisions.

Given the shortage of prior work related to efficiency and cost performance of AWS WAF rules, this study will have a much narrow focus on AWS Web Application Firewall.

III. METHODOLOGY

A. Writing AWS WAF Rules

AWS WAF has a concept of WebACL [10], which is a resource or a container for the rules. This WebACL can be attached to one or more AWS network resources like CDN, Load Balancers, API Gateway. For every request received on these network resource, the WebACL is triggered, and the rules start getting executed in order from top to bottom. If any of the rule matches and the action is set to ALLOW or DENY, the rule evaluation short-circuits, else the evaluation continues till all the rules are evaluated, in which case the default action on WebACL is considered the final action.

As we can see, it is crucial that we not only setup the rules properly, but also put them in correct order. A short circuit evaluation would be the cheapest course of action. We would ideally want to put cheap rules that can block most of the attacks. As per the AWS Whitepaper "Guidelines for Implementing AWS WAF" [11], the three most common kind of threats at Layer 7 DDoS, Web Application attacks like injection attacks etc. and Bad Bots like scrapers and crawlers.

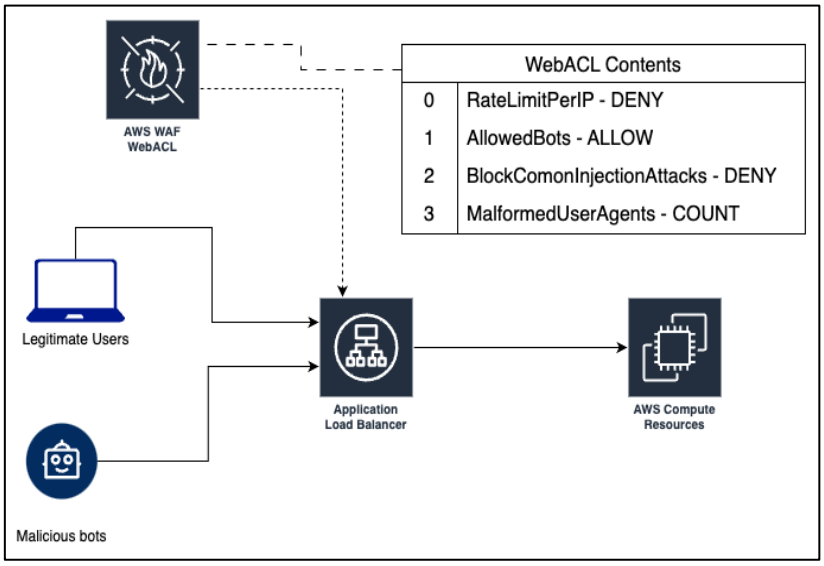

Fig 1: How WebACL are associated with network resources and their contents

B. Web Capacity Units

As mentioned in AWS's Pricing Calculator [12], the fixed costs are WebACL and rules count, which are cheap, but the variable charges are for Requests. We can chart it as

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

WCU Count	Cost per 1 million requests
<= 1500	$0.60
> 1500 and <=2000	$ 0.80
> 2000 and <= 2500	$ 1.00
> 2500 and <= 3000	$ 1.20

Fig 2: A typical pricing chart for WebACL assuming default body size

This means the best way to control costs using AWS WAF exclusively, without subscribing to any other AWS service, would be to reduce the total WCU of a single WebACL. Additionally, there is a limit of 5000 WCU [13] per WebACL.

There is another good reason for keeping WCU low for a WebACL, which is explained adequately by AWS Documentation

> *"AWS WAF uses web ACL capacity units (WCU) to calculate and control the operating resources that are required to run your rules, rule groups, and web ACLs. AWS WAF enforces WCU limits when you configure your rule groups and web ACLs. WCUs don't affect how AWS WAF inspects web traffic."*
>
> AWS WAF web ACL capacity units (WCUs) - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced. [13]

C. Computing WCU for each rule

To compute WCU for each WAF rule, we can write WAF rules locally in a file and then AWS CLI check-capacity command [14] to get the final value.

Let's say we put this WAF rule in file LimitRequestsPerIP.json. Then we install AWS CLI and execute the command

```
$ aws wafv2 check-capacity
      --scope REGIONAL
      --region us-east-1
      --rules LimitRequestsPerIP.json
```

Which should return a response

```
{
    "Capacity": 2
}
```

D. Sample WAF Rules

In this section we will list down some of the WAF rule we crafted to showcase the complexity of the rule. We wish to start with simplest rule, with one statement, then increasing the number of statements and eventually we wish to introduce nested statements. Finally, we will have one rule with multiple nesting at multiple levels.

D.1. Rate Limit to 1K – "RateLimit1K"

```
[
    {
        "Name": "RateLimit1K",
        "Priority": 0,
```

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

```
            "Statement": {
                "RateBasedStatement": {
                    "Limit": 1000,
                    "EvaluationWindowSec": 300,
                    "AggregateKeyType": "IP"
                }
            },
            "Action": {
                "Block": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "RateLimit1K"
            }
        }
    ]
```

Fig 3: The JSON WAF Rule to limit traffic per IP

D.2. Blocking Requests without Auth Header – "BlockAuthWithoutHeader"

Since this WAF rule is very extensive, we will just put forward the broad logic which can be used to reconstruct the WAF rule.

```
AndStatement (Level 1) BLOCK
|
|-- NotStatement (Level 2)
|    |-- ByteMatchStatement (Level 3) - authorization: does not start with
"Bearer"
|
|-- NotStatement (Level 2)
|    |-- ByteMatchStatement (Level 3) - authorization: does not start with
"Basic"
|
|-- NotStatement (Level 2)
 |-- ByteMatchStatement (Level 3) - authorization: does not start with
```

Fig 4: If the Authorization header does not contain either, "Bearer", "Basic" or "OAuth", then block the request

D.3. Block requests from known bad IP addresses – "AWSManagedRulesAmazonIpReputationList"

```
{
  "Name": "AWS-AWSManagedRulesAmazonIpReputationList",
  "Priority": 2,
  "Statement": {
    "ManagedRuleGroupStatement": {
      "VendorName": "AWS",
      "Name": "AWSManagedRulesAmazonIpReputationList"
    }
  },
  "OverrideAction": {
    "None": {}
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
```

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

```
        "CloudWatchMetricsEnabled": true,
        "MetricName": "AWS-AWSManagedRulesAmazonIpReputationList"
      }
    }
```

Fig 5: Just use AWS managed IP list to block requests. This list gets updated by AWS regularly

D.4. Block known bad inputs and injection attacks – "AWSManagedRulesKnownBadInputsRuleSet"

```
    {
      "Name": "AWS-AWSManagedRulesKnownBadInputsRuleSet",
      "Priority": 3,
      "Statement": {
        "ManagedRuleGroupStatement": {
          "VendorName": "AWS",
          "Name": "AWSManagedRulesKnownBadInputsRuleSet"
        }
      },
      "OverrideAction": {
        "None": {}
      },
      "VisibilityConfig": {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "AWS-AWSManagedRulesKnownBadInputsRuleSet"
      }
}
```

Fig 6: There are 11 known bad input types which are blocked with this managed rule.

D.5. Allow with Bearer auth, but not curl – "AllowBearerAuthJsonContentTypeNoCurl"

```
    AndStatement (Level 1) ALLOW
    |
    |-- ByteMatchStatement (Level 2) - content-type: application/json
    |-- ByteMatchStatement (Level 2) - authorization: starts with Bearer
    |-- NotStatement (Level 2)
    |   |-- ByteMatchStatement (Level 3) - user-agent: contains curl
    |-- SizeConstraintStatement (Level 2) - cookie: <= 1000 bytes
    |-- ByteMatchStatement (Level 2) - accept-language: contains en-US
    |-- OrStatement (Level 2)
        |-- ByteMatchStatement (Level 3) - accept: contains application/json
        |-- ByteMatchStatement (Level 3) - accept: contains text/html
        |-- ByteMatchStatement (Level 3) - accept: exactly text/json
```

Fig 7: We want to accept requests with JSON content-type using Bearer authentication, using en-us language, but not using curl. It should accept content type json and html

D.6. Mozilla browser, no curl with Bearer Auth and JSON type, not from localhost and only allows certain accept-encoding "ComplicatedRuleWithMozillaNoLocalhost"

```
AndStatement (Level 1) ALLOW
|
|-- ByteMatchStatement (Level 2) - content-type: exactly application/json
|-- ByteMatchStatement (Level 2) - authorization: starts with Bearer
```

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

```
|-- ByteMatchStatement (Level 2) - user-agent: contains Mozilla
|-- NotStatement (Level 2)
|   |-- ByteMatchStatement (Level 3) - user-agent: contains curl
|-- ByteMatchStatement (Level 2) - referer: starts with https
|-- SizeConstraintStatement (Level 2) - cookie: <= 1000 bytes
|-- NotStatement (Level 2)
|   |-- ByteMatchStatement (Level 3) - host: contains localhost
|-- ByteMatchStatement (Level 2) - accept-language: contains en-US
|-- ByteMatchStatement (Level 2) - accept-encoding: contains gzip
|-- ByteMatchStatement (Level 2) - accept-encoding: contains deflate
|-- NotStatement (Level 2)
|   |-- ByteMatchStatement (Level 3) - origin: starts with http://
|-- OrStatement (Level 2)
    |-- ByteMatchStatement (Level 3) - accept: contains application/json
    |-- ByteMatchStatement (Level 3) - accept: contains application/xml
    |-- ByteMatchStatement (Level 3) - accept: contains text/html
    |-- ByteMatchStatement (Level 3) - accept: contains application/javascript
```

Fig 8: We want to accept requests from Mozilla browser, but not curl. The request should have json content-type and use Bearer authentication, not using HTTP, but accepting json, xml, html and javascript content type.

IV. EXPERIMENTAL RESULTS

A. Collecting the data on WCU and statement count

Using check-capacity command, we determine that the WCU for each rule is

Rule Name	WCU	Statement Count
RateLimit1K	2	1
BlockAuthWithoutHeader	6	3
AWSManagedRulesAmazonIpReputationList	25	3
AllowBearerAuthJsonContentTypeNoCurl	77	9
ComplicatedRuleWithMozillaNoLocalhost	181	12
AWSManagedRulesKnownBadInputsRuleSet	200	11

B. Visualizing the Results

In this chart we observe the relationship between WAF rule statement count and the corresponding Web Capacity Units, which was evident in the table in the previous section. As the statement count increased, the WCU increased accordingly. There is not a linear correlation as the content of the rule would play a role in the overall WCU.

The last rule AWSManagedRulesKnownBadInputsRuleSet has higher WCU as it contains logic to look for well-known exploits like Sprint Core RCE [15] and Log4J [16]. Additionally, the managed rule [17] looks for those bad inputs in four places – Request headers, body, URI and Query string. This would explain how it crossed ComplicatedRuleWithMozillaNoLocalhost in WCU with lesser number of statements.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

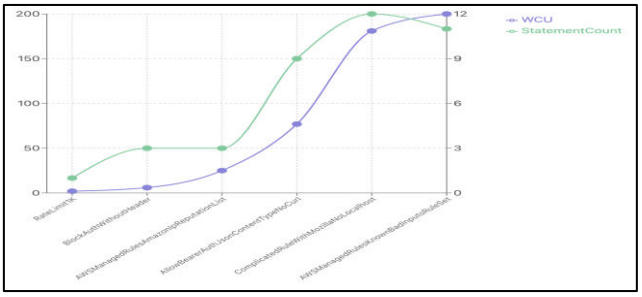

Fig 9: A plot of statement-count and WCU for each of the above 6 WAF Rules

V. CONCLUSION

The simplicity or the complexity of the AWS WAF rules is a major factor in the total Web Capacity Units. For services that serve a million of customers, the security team might find themselves having to implement different WAF rules to mitigate ongoing threats, some of them custom to their business. Special care must be taken to ensure the WAF rules are efficient, and the aim should be to use the minimum number of rules that address majority of the threats. It is possible that not all threats can be addressed as addressing all threats comes at a penalty of performance and cost.

The objective is not to reduce the WCU value of each WAF rule, but to the entire collection of rules contained inside a WebACL. The limits and cost are applied per WebACL, so clever ordering of rules can used to preserve performance and still address most of the threats. If we know the HTTP request pattern of legitimate customers, then we can add an ALLOW action for the rule at the beginning of the WebACL with order 0. This ensures legitimate customers do not experience higher latency.

REFERENCES

[1] HG Insights. (2024, June 20). AWS Market Share 2024: Insights into the Buyer Landscape. https://hginsights.com/blog/aws-market-report-buyer-landscape

[2] Configure application layer (layer 7) DDoS protections with AWS WAF - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced. (n.d.). https://docs.aws.amazon.com/waf/latest/developerguide/ddos-get-started-web-acl-rbr.html

[3] Swart, T. (2024, June 18). What Load Balancers Do at Three Different Layers of the OSI Stack. Equinix Blog. https://deploy.equinix.com/blog/a-deep-dive-into-layer-3-layer-4-and-layer-7-load-balancing/

[4] Agbenyegah, F. K., & Asante, M. (2017). Impact of firewall on network performance. International Journal of Scientific & Technology Research, 6(3), 32-38.

[5] Cosola, D. (2024). Analysis and development of a monitoring system for WAFs using AWS and ELK Stack (Doctoral dissertation, Politecnico di Torino).

[6] AWS Well-Architected Framework - AWS Well-Architected Framework. (n.d.). https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

[7] Kareem, Mohammed, "Prevention of SQL Injection Attacks using AWS WAF" (2018). Culminating Projects in Information Assurance. 47.

[8] Lakhno, V., Blozva, A., Kasatkin, D., Chubaievskyi, V., Shestak, Y., Tyshchenko, D., & Brzhanov, R. (2022). Experimental studies of the features of using waf to protect internal services in the zero trust structure. J Theor Appl Inf Technol, 100(3), 705-721

[9] M. Panda, "Performance analysis of encryption algorithms for security," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 2016, pp. 278-284,

[10] AWS WAF web access control lists (web ACLs) - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced. (n.d.). https://docs.aws.amazon.com/waf/latest/developerguide/web-acl.html

[11] Understanding threats and mitigations - Guidelines for Implementing AWS WAF. (n.d.). https://docs.aws.amazon.com/whitepapers/latest/guidelines-for-implementing-aws-waf/understanding-threats-and-mitigations.html

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307181|

[12] Pricing - AWS WAF - Amazon Web Services (AWS). (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/waf/pricing/
[13] AWS WAF web ACL capacity units (WCUs) - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced. (n.d.). https://docs.aws.amazon.com/waf/latest/developerguide/aws-waf-capacity-units.html
[14] check-capacity — AWS CLI 1.34.0 Command Reference. (n.d.). AWS CLI Command Reference. Retrieved August 17, 2024, from https://docs.aws.amazon.com/cli/latest/reference/wafv2/check-capacity.html
[15] NVD - CVE-2022-22963. (n.d.). https://nvd.nist.gov/vuln/detail/CVE-2022-22963
[16] CVE website. (n.d.). https://www.cve.org/CVERecord?id=CVE-2021-44228
[17] Baseline rule groups - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced. (n.d.). https://docs.aws.amazon.com/waf/latest/developerguide/aws-managed-rule-groups-baseline.html#aws-managed-rule-groups-baseline-known-bad-inputs

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details