Wallarm on AWS

Application Security Platform Powered by AI
Wallarm AI-powered security platform automates application protection and security testing. Hundreds of customers already rely on Wallarm to secure websites, microservices and APIs running on private and public clouds. Wallarm AI enables application-specific dynamic WAF rules, proactively tests for vulnerabilities, and creates a feedback loop to improve detection accuracy.

Key Benefits

* Adapts security rules with AI as application evolves
* Actively verifies threats to minimize manual analysis
* Protects from OWASP Top 10 and 0-day attacks
* Protects against bots and API abuse
* Lowers false-positives by customizing security rules to the application logic
* Integrates with existing infrastructure and CI/CD pipelines

Installing Wallarm

Amazon Linux

Add nginx repo

```
echo "[nginx]
name=nginx repo
baseurl=https://nginx.org/packages/mainline/rhel/7/\$basearch/
gpgcheck=0
enabled=1
" |  sudo tee /etc/yum.repos.d/nginx.repo > /dev/null
```

Install and start NGXIN

```
sudo yum -y update
sudo yum -y install nginx
sudo nginx
```

Confirm you're running the correct version of nginx

```
$ nginx -v
```

Install Wallarm RPM

```
$ sudo amazon-linux-extras install epel
$ sudo rpm -i
https://repo.wallarm.com/centos/wallarm-node/7/x86_64/Packages/wallarm-node-repo-1-2.el7.centos.noarch.rpm
$ sudo yum -y update
$ sudo yum install wallarm-node nginx-module-wallarm
$ sudo vim /etc/nginx/nginx.conf
```

Paste this after `worker_process` **line**

```
    load_module modules/ngx_http_wallarm_module.so;
```

Exit

Confirm NGINX config and reload

```
$ sudo nginx -t
$ sudo nginx -s reload
```

Follow steps below to authenticate wallarm node

Amazon Machine Image (AMI)

Launch your Wallarm Node instance from Amazon Marketplace. The instance will launch with a pre-installed Wallarm Node.

Follow steps below to authenticate wallarm node

Docker

Run the docker container

```
$ USER=
$ PASSWORD=
$ BACKEND=
$ MEMVALUE=
```

```
$ docker run -d -e DEPLOY_USER=$USER -e DEPLOY_PASSWORD=$PASSWORD -e
NGINX_BACKEND=$BACKEND -e TARANTOOL_MEMORY_GB=$MEMVALUE -p 80:80
wallarm/node
```

Amazon Kubernetes Service

Create a Cloud Node in Wallarm console and copy token

```
$ helm repo add wallarm https://repo.wallarm.com/charts/stable
$ helm repo update

$ TOKEN=
$ helm install wallarm/wallarm-ingress -n ingress-controller --set
controller.wallarm.token=<CLOUD NODE TOKEN> --set
controller.wallarm.enabled=true
```

Authenticating wallarm node

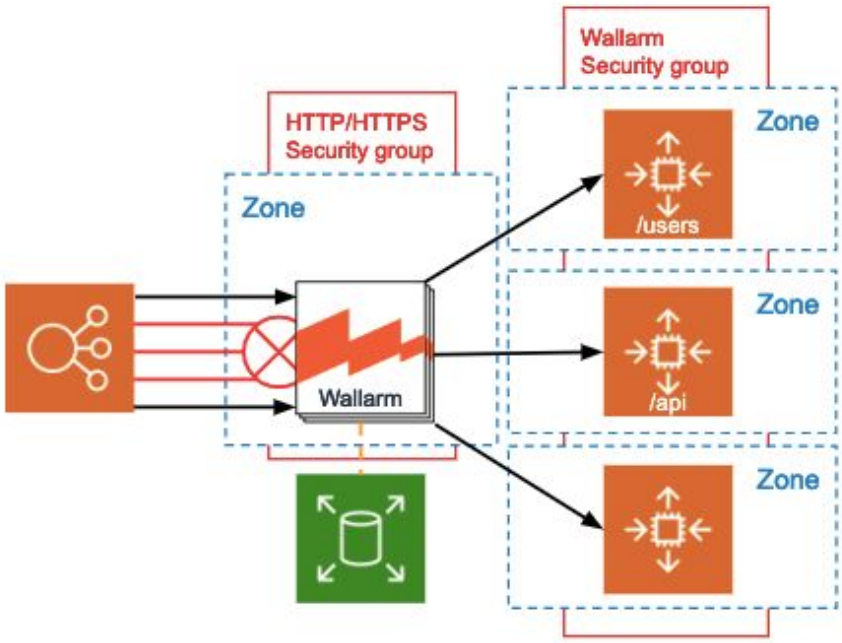
Interactive

```
sudo /usr/share/wallarm-common/addnode
```

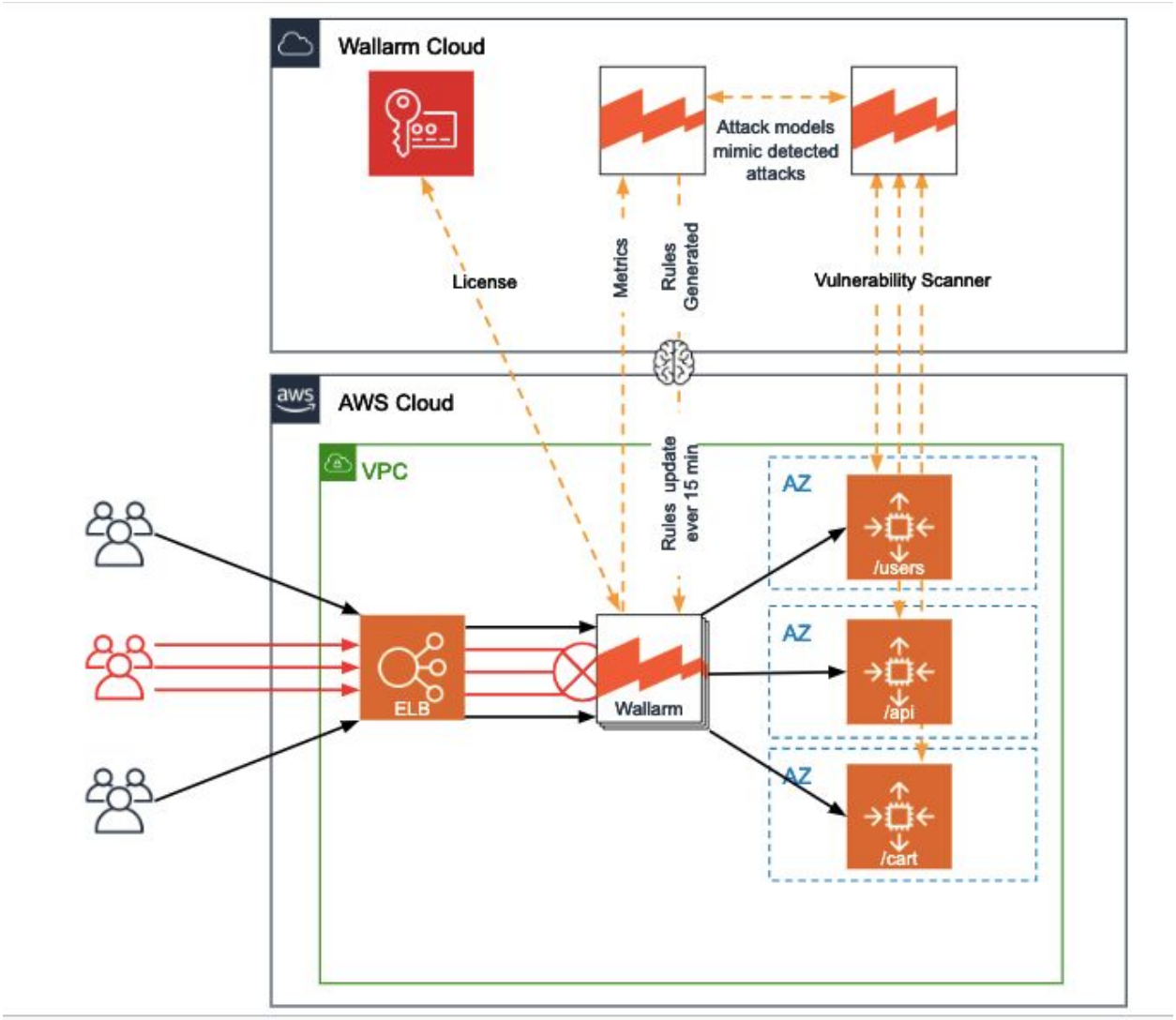
One-liner or CI/CD

```
sudo /usr/share/wallarm-common/addnode -u $EMAIL -p $PASSWORD
```

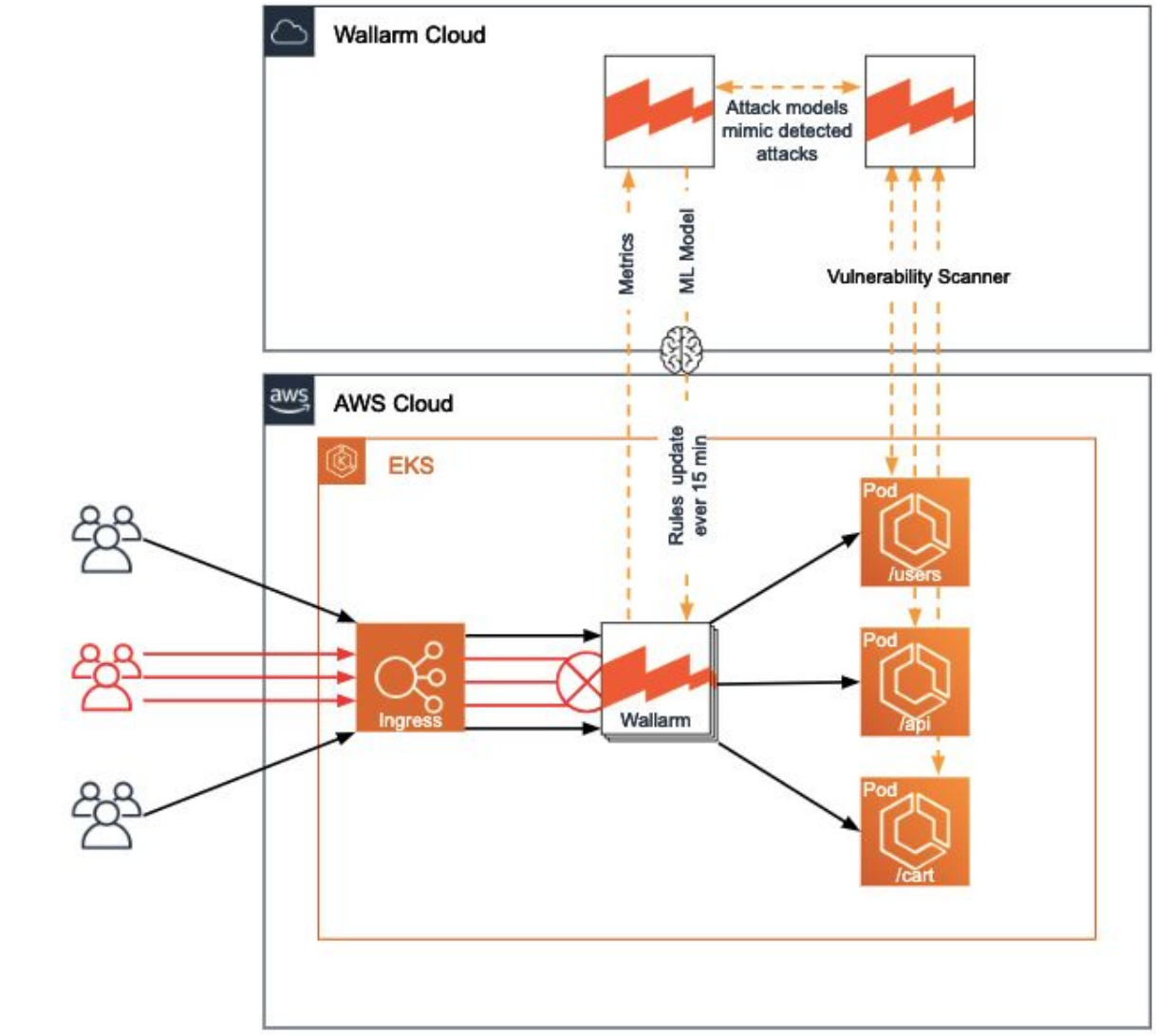
Customer Diagram



Architecture Diagram



Architecture Diagram- Kubernetes



Authentication and prescriptive guidance

Managing Wallarm Secrets & Creating the credentials

Create credential json file

```
$ echo ‘{"username":"elugo25111+deploy@gmail.com",
"password":"verysecretPa$$word1"}’ > mycreds.json
```

Create secret

```
$ aws secretsmanager create-secret --name wallarm/deploy
--secret-string file://mycreds.json
```

Confirm secret was created

```
$ aws secretsmanager get-secret-value --secret-id wallarm/deploy
--version-stage AWSCURRENT
```

From VM

From CLI Pull secrets

```
$ WALLARM_USER=$(aws secretsmanager get-secret-value --secret-id
wallarm/deploy --version-stage AWSCURRENT --query SecretString | jq -r
' fromjson | .username')
$ WALLARM_PASSWORD=$(aws secretsmanager get-secret-value --secret-id
wallarm/deploy --version-stage AWSCURRENT --query SecretString | jq -r
' fromjson | .password')
```

Add Node to license wallarm node

```
$ sudo /usr/share/wallarm-common/addnode -u $WALLARM_USER -p
$WALLARM_PASSWORD
```

Creating EC2 Security Groups

See architecture design for details

Create Outbound to Wallarm Cloud Security Group

- Go to Security Group section in your EC2 dashboard
- Click on “Create Security Group”
    - Name: Wallarm-Allow-Outbound
    - Select Outbound Security group rules:
        - Type: Custom TCP IPv4
        - Protocol: TCP
        - Port: 444
        - Destination: Custom 0.0.0.0/0

Create Inbound Rules for Scanner

- Go to Security Group section in your EC2 dashboard
- Click on “Create Security Group”
    - Name: Wallarm-Allow-Inbound
    - Select Inbound Security group Rules
        - Type: HTTP
        - Destination: Custom (add IP’s from list)
    - “Add Rule”
        - Type: HTTPS
        - Destination: Custom (add IP’s from list)
    - OPTIONAL: “Add Rule” - Optional if you’re using non-standard HTTP ports at your load balancer (eg. 8080, 8443, etc.)
        - Type: Custom TCP IPv4
        - Protocol: TCP
        - Port: (Custom port, such as 8080, 8443, etc.)
        - Destination: Custom (add IP’s from list)

Tagging EC2 Instances

Add Tags window, click the Add Tag button. Type Name in the Key field, and in the Value field type the instance name. This name is what will appear in the Name column of the summary table on the Instances tab of the EC2 dashboard.

List of Billable Services

NG WAF
Wallarm AI-Powered NG-WAF solution is offered with two pricing models.
BYOL, for current Wallarm accounts
Pay Per Usage, see Wallarm- AWS Marketplace for detailed pricing

Technical Support
24x7 technical support is currently included in all Wallarm licenses.
Contact support@wallarm.com for technical support.

24x7x365 AWS technical support is included. For more information visit AWS support.

Training and Onboarding
An initial 2 hr training is included for new Wallarm accounts on AWS. The initial onboard training will cover GUI, technical views, feature set tutorials, and how to manage and scale Wallarm across your AWS stack.

Contact support@wallarm.com for technical support services.

Licensing Costs

Easily deploy and scale Wallarm security across your AWS architecture. Wallarm offers 2 options to protect APIs and applications. A current Wallarm license or AWS facilitated pay per usage is required.

BYOL
For current Wallarm accounts, follow the instructions here to deploy Wallarm as an AMI (Amazon Machine Image).

To add additional nodes for microservices, or for your AWS infrastructure, contact request@wallarm.com

Pay Per Usage
If you do not have a current Wallarm license, simply add Wallarm through the AWS Marketplace and pay per usage. Typical price is $.20/hr for software and infrastructure services hosted in the U.S. See AWS Marketplace for detailed pricing and applicable taxes and fees.

For Public Sector pricing, Academic, or Enterprise, contact request@wallarm.com

EC2 Instance Guidance

It is recommended to use at least m4.xlarge EC2 instances. You will need at least 4GB of RAM and 4 vCPUs for the filtering node and 32GB of RAM with 4 vCPUs for the behavioral analytics module. If your nodes are processing more than 500 average RPS higher memory allotment may be required.

EC2 Instance Guidance

It is recommended to use at least m4.xlarge EC2 instances. You will need at least 4GB of RAM and 4 vCPUs for the filtering node and 32GB of RAM with 4 vCPUs for the behavioral analytics module. If your nodes are processing more than 500 average RPS higher memory allotment may be required.

Detecting Faults

Multi-Region Availability and DR

- Create two wallarm instances in two different regions
- Follow the {SET UP GUIDE}
- Create a health check endpoint for route 53 in your `/etc/nginx/nginx.conf`

```
server {
  location /healthz {
    access_log off;
    return 200;
  }
}
```

- Restart nginx `nginx -t & nginx -s reload`
- Go to your Route 53 console and create a health check
    - Name: wallarm-node
    - What to Monitor: Endpoint
    - Specify endpoint: IP
    - Protocol: HTTP
    - IP Address: [your wallarm nodes ip]
    - Port: 80
    - Path: /healthz
- In the Route 53 console create a traffic policy to create a primary and secondary endpoint

Instance and/or Zone Fault

Using Application Load Balancers (ALB)

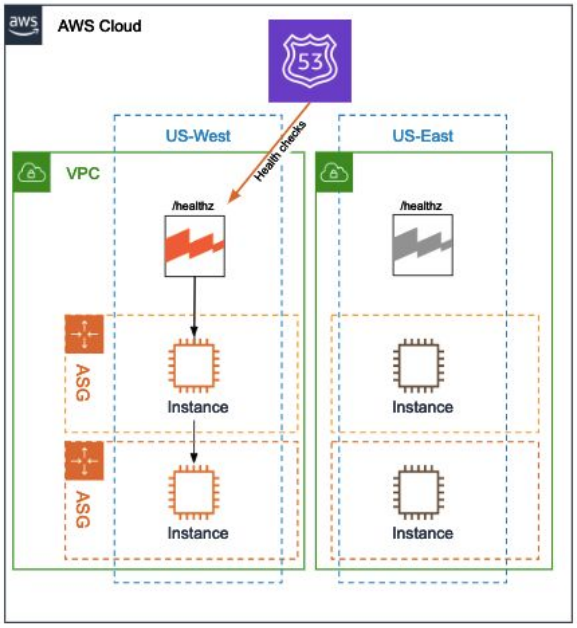
- Create a health check endpoint for route 53 in your `/etc/nginx/nginx.conf`

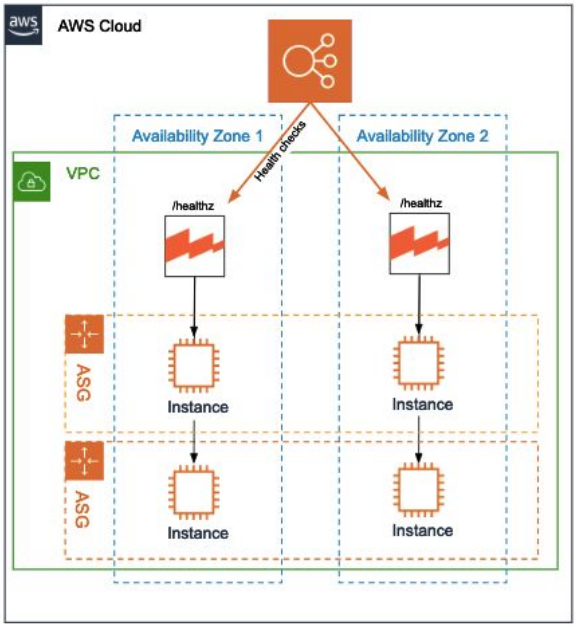
```
server {
  location /healthz {
    access_log off;
    return 200;
  }
```

```
        }
```

- Restart nginx `nginx -t & nginx -s reload`
- `Create a ALB, and use HTTP and/or HTTPS protocols`
- `Select at least two subnets in two different zones`
- `Select a SecurityGroup that has HTTP and/or HTTPS opened`
- `Name your target group`
- `Create a health check against /healthz for 200 response`
- `Register your wallarm nodes to the new target group`
- `Review and finish`

AZ and Instance Fault





Backup and Recovery

Instance, Zone, and Region Failure

While rules are stored locally on each Wallarm node, the rules are generated and pulled from the Wallarm Cloud.

The backup recovery, for instance, zone, and region failure:

- In the case of a total loss of node either due to the instance, zone or region faults, a new node coming up will automatically begin to download the existing rule set from the Wallarm cloud. Each node also comes with a predefined ruleset trained to identify most of the attacks. It's not adjusted to the customer app profile, which provides some additional failover protection.

Storing Wallarm Configs in Git

- Wallarm configs can be stored in your AWS CodeCommit repository or any preferred code repository. These configs will be stored in /etc/nginx/config.d
- During a recovery, a new EC2 instance will be built using your own AMI or the Wallarm provided AMI. Once the EC2 instance is built, you'll want to pull the config from AWS CodeCommit, and restart nginx

```
nginx -t
nginx -s restart
```

Note: Do not store secrets in your git repo, you should store your Wallarm API credentials as well as any other passwords, TLS private keys, and anything else that shouldn't be checked in, into Amazon KMS or some other secret store

Wallarm in Kubernetes (EKS)

- To store config in Kubernetes, use the ConfigMap resource

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-conf
data:
  nginx.conf: |
    user nginx;
    worker_processes  auto;
    error_log  /var/log/nginx/error.log;
```

```
load_module modules/ngx_http_wallarm_module.so;
[ ...rest of config]
```

- As with an EC2 instance, you do not want to store your wallarm api key, credentials, TLS keys, passwords, etc in your configs, create a secret. Replace $APIKEY with your own Wallarm API Key

  ```
  kubectl create secret generic wallarm-creds -n default
  --from-literal=wallarmAPI=[APIKEY]
  ```

Instructions for Conduct Test Recovery

After the recovery, proper operation of Wallarm node can be verified via collected by running Nagios compatible scripts.
collectd_nagios

Specific parameters that can be monitored are described in this document:
https://docs.wallarm.com/en/admin-en/configure-monitor-en.html

Updates and Patches

To update wallarm

```
# RHEL based
yum update wallarm-node
```

or

```
# Debian based
apt-get update
apt-get dist-upgrade
```

NOTE:
Your system must have access to `https://repo.wallarm.com` to download the packages. Ensure the access is not blocked by a firewall.

Check that nginx config is valid

```
nginx -t
```

Restart nginx

```
service nginx restart
```

Support Details and Requests

Contact support by opening the Contact Support window in the interface or by writing to support@wallarm.com.

Once an MSA is executed, support requests can be placed via dedicated chat channels such as slack or telegram, or via telephone, as specified in the MSA.

Technical Support

24 x7 Technical support is included with the subscription at no extra charge. Wallarm support can be reached at support@wallarm.com, or see support requests for further options.

Different Support Tiers and SLA

Support Tier Levels

Currently, there are no support tier levels. All active subscriptions receive 24x7 technical support. Contact support at support@wallarm.com or see the Support Request section for more information.

Maintenance Notification

No less than thirty (30) calendar day's prior written notice to Customer of All non-emergency maintenance to be performed on the Services, will be provided in no less than thirty (30) calendar days. Written notice including a detailed description of all maintenance to be performed will be given to the contact assigned to the account.

SLA

Wallarm ‘Service’ and SLA applies solely to Wallarm solution(s) and excludes any third-party application, integration, service, or platform. The following services are included for all active Wallarm subscriptions and accounts: service to bug fixes, corrections, modifications, enhancements, upgrades, and new releases to the Services to ensure: (a) the functionality of the Services is available to Authorized Users. Service; (b) the functionality of the Services in accordance with the representations and warranties set forth herein, including but not limited to, the Services conforming in all material respects to the specifications, functions, descriptions, standards, and criteria stated (c) the Service Level Standards can be achieved.

Wallarm will use commercially reasonable efforts to make the Services available at least 99.95% during each calendar month. For the purposes of this SLA, “Availability” means Customer’s ability to access and use the Services, and is calculated by subtracting from 100% the number of minutes during the applicable calendar month in which the functionalities of the Services are not Available, but excluding periods of any Wallarm SLA Exclusion (defined below).
SLA in agreement applies to the availability of Service Provider. Availability of Customer Web Applications doesn’t depend on the Availability of the Services provided by Service Provider. Service Provider SLA doesn’t affect SLA of Customer Web Applications.

In case of issues related to the availability of Wallarm Cloud user interface, Customer can request a report for the month when issues happened. This report can be requested via support at support@wallarm.com

In the event Service Provider does not meet a Service Level Standard, Service Provider shall: (a) owe to Customer any applicable Performance Credit; and, (b) use its best efforts to ensure that any unmet Service Level Standard is subsequently met. Notwithstanding the foregoing, Service Provider will use its best efforts to minimize the impact or duration of any outage, interruption, or degradation of Service.