# Отчет по работе SecSense kyarnk

Кярнонен Никита.
В рамках командных соревнований DevSecOps 2025 мною была проделана следующая работа:

1. Первичная защита и анализ хостов
2. Организационные моменты
3. Инфраструктура
4. Deploy
5. WAF
6. Мониторинг
7. Логирование
8. Пентест и уязвимости (на уровне наших приложений и помощи команде)

## Первичная защита и анализ хостов

Я выступал в роли капитана команды, поэтому сразу сделал всем wg конфиги и раскидал. Зашел на сервер, прокинул нужные ключи и начал анализ хостов:

1. Проверка интерфейсов
2. sudoers файлов
3. процессов
4. пользователей
5. порты
6. директории
7. флаги
   После первичной проверки были найдены флаги, уязвимое php, пользователи неизвестные, были мной откнфигурированы ssh и sudoers.
   Нашел что по умолчанию стоит apache2, менял на nginx потом

## Организационные моменты

Далее я создал гитлаб, сделал доску issue board и распределил задачи на первый день (в другие дни я также отслеживал статус кто чем занят и что нам нужно сделать). Каждый

выбрал зону ответственности, были кто и просто на подхвате.

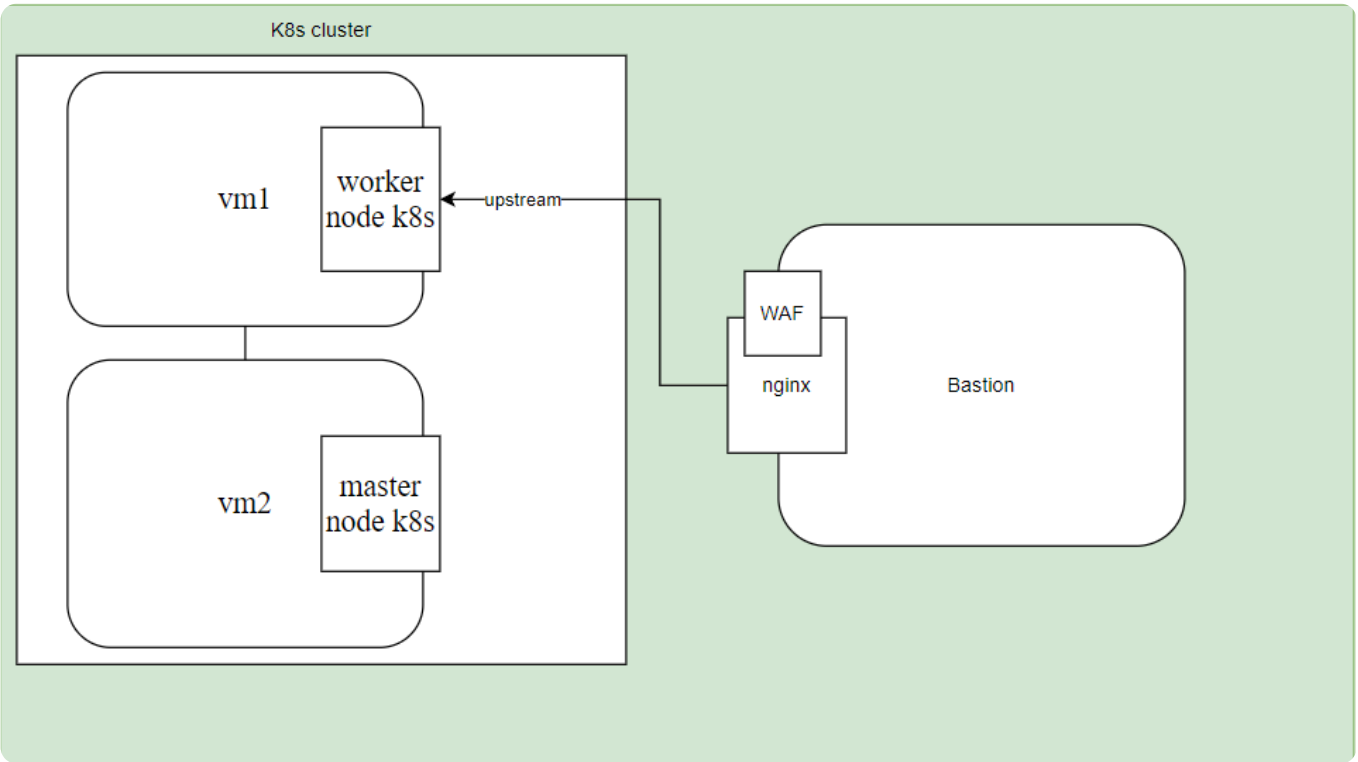
За несколько дней до соревнований я собрал информацию и протестировал локально работу со сканерами уязвимости и предоставил ребятам собственные инструкции/документации

В целом везде где нужно было помочь с частью git управления, работа с docker hub

Подготовка общего отчета и презентации

## Инфраструктура

На мне полностью лежала инфраструктура, поэтому я пошел изучать и разворачивать k8s cluster через kubespray. Представил структуру




И пошел реализовывать. Kubespray представляет собой Ansible развертку

```
[all]
master ansible_host=10.10.17.4 ip=10.10.17.4 etcd_member_name=etcd1
worker ansible_host=10.10.17.3 ip=10.10.17.3

[kube_control_plane]
master ansible_host=10.10.17.4 ip=10.10.17.4

[kube_node]
```

```
worker ansible_host=10.10.17.3 ip=10.10.17.3

[etcd]
master ansible_host=10.10.17.4 ip=10.10.17.4

[k8s_cluster:children]
kube_control_plane
kube_node
```

А вот сама установка

```
Kube
sudo apt-get update -y
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update -y
sudo apt-get install git pip python3.11 -y

sudo -i
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3.11 get-pip.py

RETURN TO USER
exit
git clone git@github.com:kubernetes-sigs/kubespray.git
cd kubespray/
python3.11 -m pip install -r requirements.txt
python3.11 -m pip install ruamel.yaml

Копируем пример инвентаря
cp -rfp inventory/sample inventory/mycluster

Объявить ip
declare -a IPS=(10.10.17.3 10.10.17.4)
создаем inventory

В контрол ноду одна нода
etcd одинаково по хостам


ansible-playbook -i inventory/mycluster/inventory.ini cluster.yml -b -v &

Конфиг копируем на мастер и с него управляем
mkdir ~/.kube
```

```
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
sudo chown $(id -u):$(id -g) ~/.kube/confg
```

## Deploy

Дальше я расписывал уже Deployment, Service, ConfigMap, StorageClass для k8s
Я деплоил как и CRUD приложение, так и приложение из 3тьего задания, так и мониторинг
Дебажил долго 3тье задание...

## WAF

После я устанавливал nginx на бастион сервер и в настройках конфигурации связывал с Service (NodePort) нашего приложения и других сервисов. Все yaml файлы указаны в отчете на команду, тут укажу кратко.
Также есть modsecurity.conf он в командом отчете отображен

```
upstream dedushka {
    server 10.10.17.3:30008;
}

upstream prometheus {
    server 10.10.17.3:30090;
}

upstream grafana {
    server 10.10.17.3:30080;
}


upstream devsec {
    server 10.10.17.3:30007;
}


server {
        listen 80 default_server;
        listen [::]:80 default_server;
        modsecurity on;
        modsecurity_rules_file /etc/nginx/modsecurity.conf;
```

```
        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                proxy_pass http://devsec;
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
                try_files $uri $uri/ =404;
        }

        location /api/ {
        #proxy_pass http://localhost:3500;
                proxy_pass http://devsec;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }


        location /grafana/ {
                modsecurity off;
```

```
                proxy_pass http://grafana;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }


        location /prometheus/ {
                modsecurity off;
                proxy_pass http://prometheus;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;

        }

       location /dedushka/ {
                rewrite ^/dedushka(/.*)$ $1 break;
                proxy_pass http://dedushka/;
                proxy_redirect off;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
        }
```

Далее на мне стоял WAF, установил и связал с nginx в конфигурациях. Получилось обезопасить инфраструктуру таким подходом

```
sudo apt install gcc make build-essential autoconf automake libtool libcurl4-openssl

cd /opt && sudo git clone https://github.com/owasp-modsecurity/ModSecurity.git cd Mo
sudo git submodule init sudo git submodule update
sudo ./build.sh
sudo ./configure
sudo make
sudo make install

Тут nginx исходники 1.24.0v
sudo ./configure --with-compat --add-dynamic-module=/opt/ModSecurity-nginx sudo make

sudo cp objs/ngx_http_modsecurity_module.so /etc/nginx/modules-enabled/
sudo cp /opt/ModSecurity/modsecurity.conf-recommended /etc/nginx/modsecurity.conf su
```

```
Это в мейн конф
load_module /etc/nginx/modules-enabled/ngx_http_modsecurity_module.so;

Это в дефолт
modsecurity on;
modsecurity_rules_file /etc/nginx/modsecurity.conf;
```

CodeRuleSet

```
sudo git clone https://github.com/coreruleset/coreruleset.git /etc/nginx/owasp-crs
sudo cp /etc/nginx/owasp-crs/crs-setup.conf{.example,}
sudo nano /etc/nginx/modsecurity.conf
И вписываем
Include owasp-crs/crs-setup.conf
Include owasp-crs/rules/*.conf
```

# Мониторинг

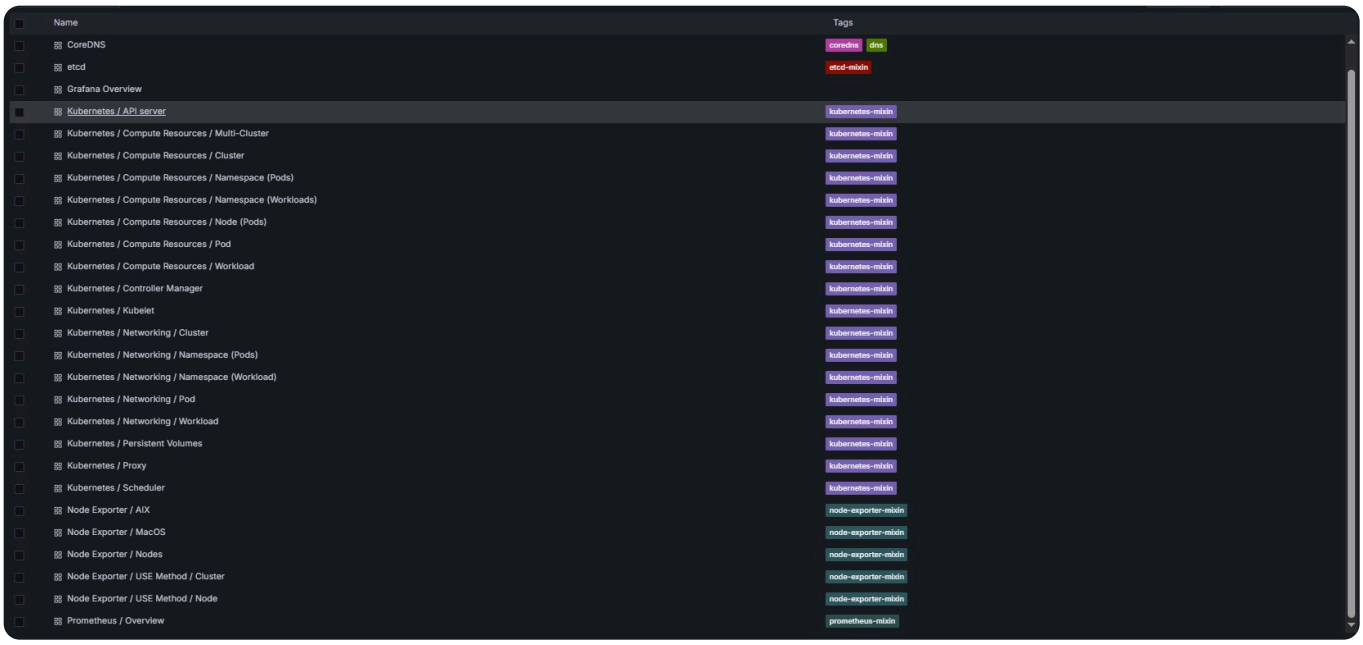
Я взял helm чарт kube-prometheus-stack и задал ему кастомный values для запуска

helm upgrade --debug --install --namespace monitoring --values kube-prometheus-stack/values2.yaml --timeout 22s kube-prometheus-stack ./kube-prometheus-stack

Все подробно расписано в отчете на команду, тут укажу пару скринов и values.yaml
Пронаблюдал как на нас нагрузка попадает

```yaml
prometheus:
  prometheusSpec:
    serviceMonitorSelectorNilUsesHelmValues: false
    podMonitorSelectorNilUsesHelmValues: false

  service:
    type: NodePort
    nodePort: 30090  # Пример: порт для доступа к Prometheus через ноду

alertmanager:
  enabled: true
  service:
    type: NodePort
    nodePort: 30093
```

```yaml
grafana:
  enabled: true
  adminPassword: "admin"  # Задай свой пароль

  service:
    type: NodePort
    nodePort: 30080

  ingress:
    enabled: false  # Включи, если у тебя есть Ingress Controller

  # Подгружаем дашборды автоматически
  defaultDashboardsEnabled: true

  grafana.ini:
    server:
      root_url: http://100.66.161.237/grafana
      serve_from_sub_path: true

nodeExporter:
  enabled: true

kube-state-metrics:
  enabled: true
```


Name
Tags
CoreDNS
coredns
dns
etcd
etcd-mixin
Grafana Overview
Kubernetes / API server
Kubernetes / Compute Resources / Multi-Cluster
Kubernetes / Compute Resources / Cluster
Kubernetes / Compute Resources / Namespace (Pods)
Kubernetes / Compute Resources / Namespace (Workloads)
Kubernetes / Compute Resources / Node (Pods)
Kubernetes / Compute Resources / Pod
Kubernetes / Compute Resources / Workload
Kubernetes / Controller Manager
Kubernetes / Kubelet
Kubernetes / Networking / Cluster
Kubernetes / Networking / Namespace (Pods)
Kubernetes / Networking / Namespace (Workload)
Kubernetes / Networking / Pod
Kubernetes / Networking / Workload
Kubernetes / Persistent Volumes
Kubernetes / Proxy
Kubernetes / Scheduler
Node Exporter / AIX
Node Exporter / MacOS
Node Exporter / Nodes
Node Exporter / USE Method / Cluster
Node Exporter / USE Method / Node
Prometheus / Overview
kubernetes-mixin
node-exporter-mixin
prometheus-mixin


datasource
Prometheus
Instance
10.10.17.4:9100
Last 1 hour
UTC
Refresh
30s
14 GiB
12 GiB
10 GiB
8 GiB
6 GiB
4 GiB
2 GiB
0 B
20:30
20:35
20:40
20:45
20:50
20:55
21:00
21:05
21:10
21:15
21:20
memory used
memory buffers
memory cached
memory free
10.4%
Disk
Disk I/O
90 kB/s
80 kB/s
70 kB/s
60 kB/s
50 kB/s
40 kB/s
30 kB/s
20 kB/s
10 kB/s
0 B/s
0.8%
0.7%
0.6%
0.5%
0.4%
0.3%
0.2%
0.1%
0%
21:25
sda read
sda written
sda io time
Disk Space Usage
Mounted on
Size
Available
Used
Used, %
/
33.6 GB
17.9 GB
15.7 GB
46.7%
/run
1.68 GB
1.67 GB
2.72 MB
0.162%
/run/containerd/io.containerd.grpc.v1
67.1 MB
67.1 MB
0 B
0%
/run/containerd/io.containerd.grpc.v1
67.1 MB
67.1 MB
0 B
0%
/run/containerd/io.containerd.grpc.v1
67.1 MB
67.1 MB
0 B
0%
Network
Network Received
60 kb/s
50 kb/s
40 kb/s
30 kb/s
20 kb/s
10 kb/s
0 b/s
caliedfb0c89796
calif3661da7fc4
docker0
ens18
kube-ipvs0
nodelocaldns
vxlan.calico
Network Transmitted
300 kb/s
200 kb/s
100 kb/s
0 b/s


# Логирование

Логирование я задеплоил через loki-stack чарт с дефолтным вэлью, потому что настраивал уже его не вовремя. Просто через helm

А так использовали часто

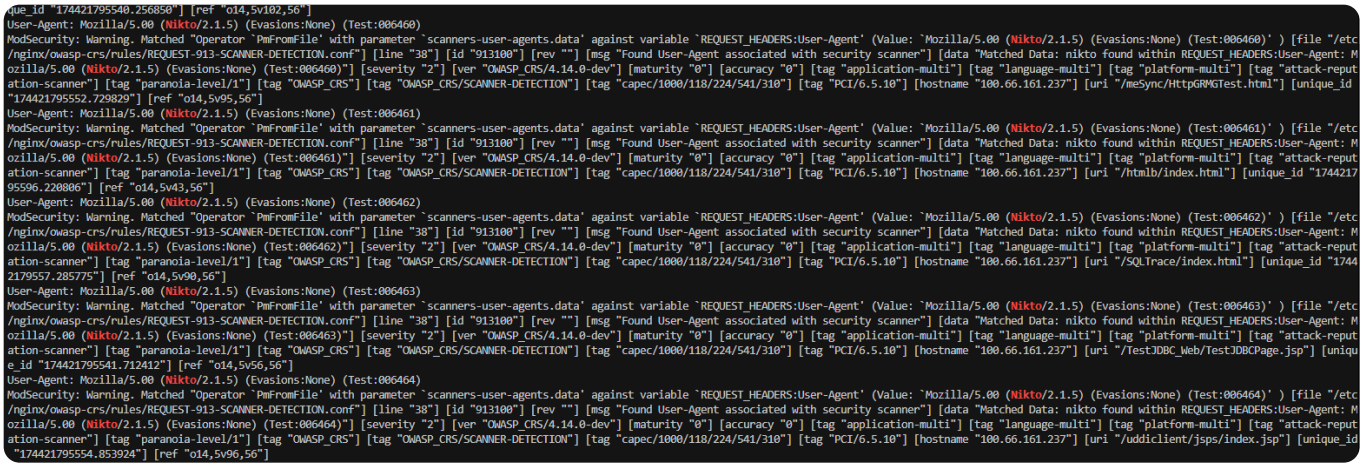
```
sudo tail -n 1000 /var/log/nginx/access.log
sudo tail -f /var/log/modsec_audit.log
htop (не ставили btop, чтобы меньше утилит было)
```

Чтобы получать всю нужную информацию

В sudo tail -n 1000 /var/log/nginx/access.log мы случайно увидели, что нас сканят DASTом Nikto, я проверил нагрузки, все было хорошо, ModSecurity отработал отлично (accesslog nginx затерся)

вот вывод с sudo cat /var/log/modsec_audit.log в момент, когда нас сканировали DAST сканеров утилитой Nikto

ModSecurity успешно блокировал запросы




# Пентест и уязвимости

Как я и сказал, я предоставил документации к сканерам и какие сканеры можно использовать в целом

Я и Дмитрий помогли разработчику приложения просканировать его
Также я отдельно сканировал через Syft Grype наше приложение

В Пентесте я принимал участие исключительно в подсказках, где и что можно посмотреть, что можно применить (касаемо LFI php например). Плюс познакомился с инструментом hydra, который мне показал Дмитрий