



**UNIWERSYTET  
WSB MERITO  
WROCŁAW**

**Wydział Finansów i Zarządzania**

Kierunek: Informatyka

**Tatiana Hołycz**

(numer albumu: 78535)

**Systemy monitoringu IT jako mechanizm ciągłego  
testowania infrastruktury i oprogramowania.**

Projekt kierunkowy

**Prowadzący Projekt kierunkowy:**

**mgr inż. Emil Bakalarz**

Wrocław 2023



## Spis treści

1.	Wstęp.....	4
1.1.	Cel pracy.....	5
1.2.	Rozwiązanie przyjęte w pracy .....	5
2.	Charakterystyka środowiska testowego .....	6
2.1.	Opis infrastruktury fizycznej.....	6
2.2.	Opis infrastruktury logicznej .....	7
2.3.	Konfiguracja środowiska testowego .....	8
3.	Testy osiągalności .....	12
3.1.	Wprowadzenie .....	12
3.2.	Przykłady zastosowań biznesowych.....	12
3.3.	Testy manualne .....	13
3.4.	Testy automatyczne .....	16
4.	Testy sprawnościowe.....	21
4.1.	Wprowadzenie .....	21
4.2.	Przykłady zastosowań biznesowych.....	21
4.3.	Testy manualne .....	22
4.4.	Testy automatyczne .....	30
5.	Podsumowanie i omówienie rezultatów pracy .....	35
	Spis podstawowych pojęć i definicji.....	37
	Spis materiałów źródłowych .....	39
	Spis rysunków.....	40
	Spis listingów.....	41
	Załączniki.....	42

## 1. Wstęp

Choć od stworzenia pierwszego komputera typu PC minęło jedynie 45 lat - co w kontekście historii naszej cywilizacji stanowi jedynie moment - był to niezwykle intensywny czas i niespotykany postęp w rozwoju. Otaczająca nas rzeczywistość zmieniła się na tyle, że niewyobrażalnym stało się dla nas istnienie jakiegokolwiek organizacji, instytucji czy firmy, niezależnie od jej wielkości, która potrafiłaby funkcjonować poprawnie bez narzędzi i infrastruktury IT.

Wspomniana intensywność ostatnich lat w kwestii rozwoju technologii informatycznych, oprogramowania i usług, a także tak powszechny dla społeczeństwa błąd percepcyjny polegający na dostrzeganiu, docenianiu i koncentracji uwagi na przedmiotowej (użytkowej) warstwie szeroko rozumianej technologii IT, mocno wpłynęły na obecnie postrzegany obraz technologii, na jej rozwój i zalety, jednocześnie nie dostrzegając warstwy technologicznej, na której się ona opiera. Sieć Internet oraz usługi jakie dzięki niej można uzyskać wykorzystywane są przez społeczeństwo każdego dnia, zapominając (w najlepszym przypadku nie kładąc priorytetu na jej stan), o infrastrukturze sieci IT, która de facto odpowiada i jest niezbędna do dalszego funkcjonowania wspomnianych usług.

Do niedawna jeszcze, w społeczeństwie funkcjonowało pojęcie kogoś takiego jak „pan od drukarki”. Niezależnie od tego czy mówimy o administratorze sieci IT, czy informatyku utrzymania systemów IT, jego pozycja wewnątrz większości organizacji i firm była niemalże na poziomie „konserwatora” bądź portiera odpowiedzialnego za parking. W latach 2017 - 2019 studia na kierunku Informatyka ukończyło od 13 tys. - 15 tys. studentów rocznie. Szacunki Ministerstwa Edukacji wskazują, iż 85% wspomnianych absolwentów stanowili programiści aplikacji, stron internetowych, urządzeń IoT i aplikacji oraz gier na urządzenia mobilne. Wnioskując z tego technicy systemów IT oraz administratorzy sieci IT stanowili mniej niż 15% wszystkich studentów, mimo iż to od ich pracy, wiedzy i potencjału rozwojowego zależy czy pozostałe 85 % osób będzie miało możliwość pracować i wykorzystywać swoją wiedzę.

Ostatnie dwa lata to nieoczekiwany zwrot tej sytuacji i dynamicznie rosnące znaczenie jak i zapotrzebowanie, na wcześniej wymienianych „panów od drukarki”. Kluczowy wpływ na taką sytuację ma intensyfikacja i wysoka podatność na zagrożenia cybernetyczne, tak dla samej sieci strukturalnej jak i dla samych aplikacji, duża ilość uszkodzonych, utraconych bądź w najgorszym wypadku wykradzionych danych wewnętrznych firm i organizacji, a także ich publicznych wycieków.

Zbrodniczy najazd Rosji na Ukrainę i wybuch wojny w Europie był w tym kontekście ostatecznym i bardzo silnym katalizatorem tego procesu, a sama wojna, w szczególności jej pierwsze miesiące, jak w soczewce pokazały słabość istniejących systemów zabezpieczeń, błędów

i niedoszacowania roli testerów w projektowaniu aplikacji, a także cały wachlarz wiążących się z tym konsekwencji – od przejęcia kontroli nad wieloma systemami monitoringu, poszczególnymi sieciami telewizji kablowej, aż do ryzyka uszkodzenia sieci energetycznej i gazowej oraz utraty kontroli nad grupą wojskowych satelitów orbitalnych należących do Rosji.

### **1.1. Cel pracy**

Genezą powstania tej pracy, jest chęć przedstawienia jak duże znaczenie i wagę dla bezpieczeństwa oraz stabilności funkcjonowania sieci IT stanowi rola testera i bieżące monitorowanie dostępności i stanu infrastruktury. Wykorzystanie zdobytej w procesie edukacji wiedzy, metodologii działań, a także warsztat pracy daje szansę wejścia na szeroki i niezagospodarowany do tej pory obszar działań i zastosowań testowania w kontekście bezpieczeństwa, stabilności i przewagi w obszarze bezpieczeństwa narodowego instytucji państwowych, jak i przewagi rynkowej i konkurencyjnej pojedynczych firm i dużych podmiotów gospodarczych.

### **1.2. Rozwiązanie przyjęte w pracy**

W pracy przedstawione są dwa kluczowe i komplementarne dla siebie zagadnienia, jakimi jest:

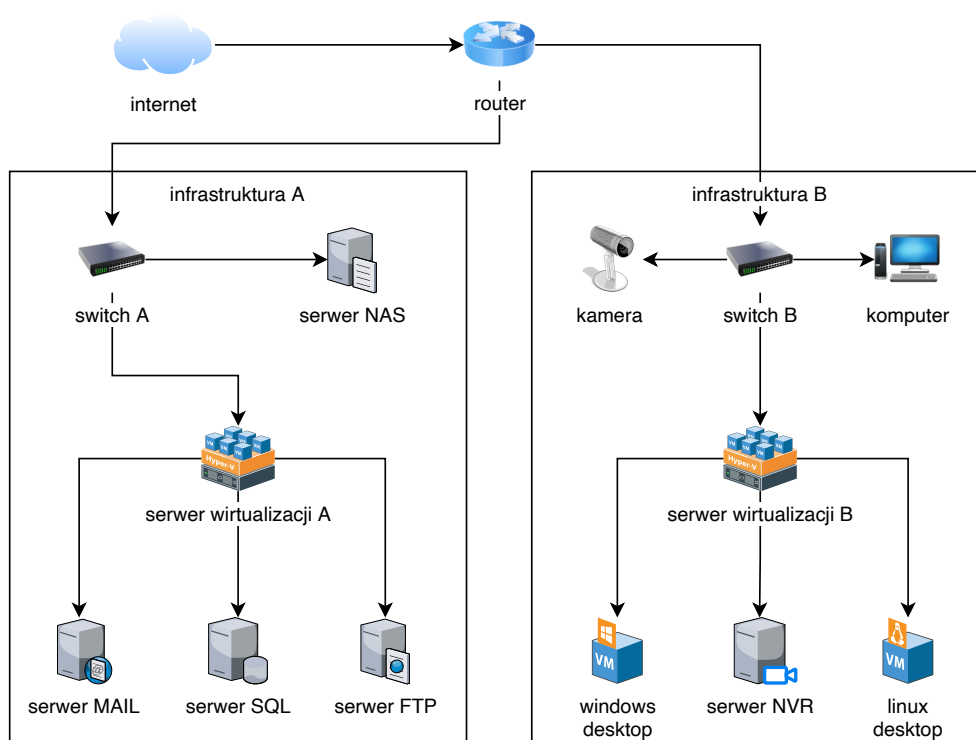
1. Utrzymanie stałej dostępności infrastruktury poprzez system monitoringu sieciowego.
2. Redukcja potencjalnego ryzyka strat, przez stałą analizę kondycji sieci i infrastruktury IT.

Aby to zrealizować, wykorzystane zostały darmowe i bardzo podstawowe, ogólnodostępne narzędzia z dziedziny administracji sieci IT, poprzez realizację i prezentację wyników pojedynczych testów manualnych, następnie ich automatyzację i autonomizację.

## 2. Charakterystyka środowiska testowego

Poniżej przedstawione zostało środowisko testowe wykorzystywane w ramach niniejszej pracy, w podziale na trzy warstwy warstwę sprzętową, logiczną i programową.

### 2.1. Opis infrastruktury fizycznej



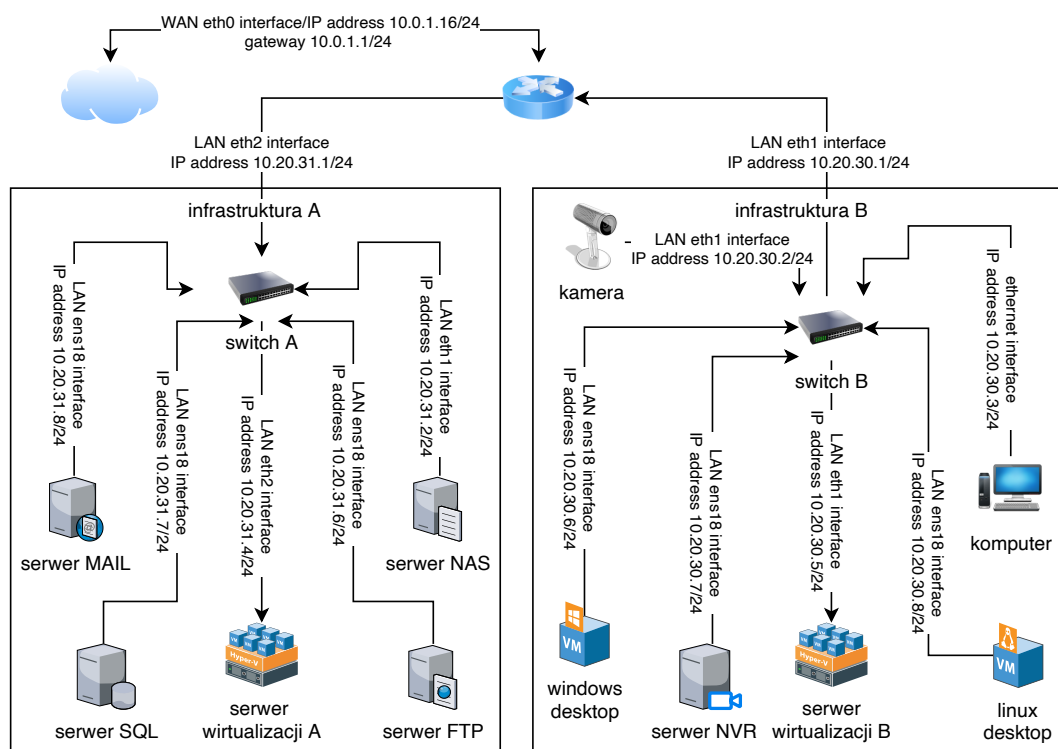
Rys. 1. Schemat fizyczny infrastruktury technicznej

Przygotowana na cele niniejszej pracy infrastruktura, oparta jest na rzeczywistym sprzęcie wyszczególnionym poniżej, a konstrukcja infrastruktury jest faktycznym odzwierciedleniem przedsiębiorstwa bądź organizacji składającej się dwóch oddziałów, połączonej wspólnym *routerem*.  
Parametry techniczne infrastruktury:

1. *router* – Ubiquiti model Edgerouter X - 5 niezależnych interfejsów sieciowych RJ45 o przepustowości 1 Gbps każdy, zarządzany przez terminale telnet, ssh oraz interfejs www
2. switch A – Cisco\Linksys model SG100D-08 – niezarządzany, 8-portowy, o przepustowości każdego z portów 1 Gbps
3. serwer NAS – Synology model DS411 – obsługujący do czterech dysków twardych z interfejsem SATA, 1 interfejsem sieciowym o przepustowości 1 Gbps, dwoma portami USB typu 2.0, portem eSata, procesorze taktowanym 1.6 GHz, pamięci ram o pojemności 512 MB w technologii DDR3

4. serwer wirtualizacji A – Dell PowerEdge 2950 II, 4 x port LAN (RJ45) 1 Gbps, 1 dedykowany interfejs LAN (RJ45) iDrac (integrated Dell Remote Access Controller), 2 x PSU 750W, 2 x Intel(R) Xeon(R) CPU E5345, 4 rdzenie taktowane 2.33GHz, 32 GB pamięci RAM w technologii DDRIII z kontrolą ECC, karta graficzna ATI Radeon HD 4200
5. kamera Hikvision – DS-2CD1641FWD, o maksymalnej rozdzielczości 2688 x 1520, 1 interfejsie LAN 10/100, zasilana przy użyciu protokołu PoE 802.3af
6. komputer stacjonarny – HP Compaq 6005 Pro SFF, 1 x port LAN (RJ45) 1Gbps, procesor AMD Phenom(tm) II X4 B97 taktowany 3.20 GHz, pamięć RAM 8 GB w technologii DDRIII
7. serwer wirtualizacji B - Dell PowerEdge T320, 2 x port LAN (RJ45) 1Gbps, 2 x PSU 350W, 1 x Intel(R) Xeon(TM) E5-2407 v2, 4 rdzenie taktowane 2.40GHz, pamięć ram 16 GB w technologii DDR3 z kontrolą ECC

## 2.2. Opis infrastruktury logicznej



Rys. 2. Schemat logiczny infrastruktury technicznej

Główny router posiada trzy interfejsy sieciowe: eth0, eth1 oraz eth2. Interfejs eth0 pełniący rolę interfejsu WAN połączony jest z Internetem i ma przypisany zdefiniowany w serwerze DHCP statyczny adres IP 10.0.1.16/24 oraz bramę 10.0.1.1/24. Eth1 (adres IP to 10.20.30.1/24)

połączony jest z *infrastrukturą B*, natomiast Eth2 (adres IP to 10.20.31.1/24) połączony jest z *infrastrukturą A*.

Na obu interfejsach włączona jest usługa DHCP przydzielająca adresy IP urządzeń wewnątrz infrastruktury na bazie listy statycznych dzierżaw w usłudze. Interfejsy nie zostały połączone (*bridge*), więc każdy przydziela adresy IP w masce sieciowej odpowiedniej dla danej infrastruktury. Między interfejsami zachowano utworzony dynamicznie routing pakietów pomiędzy obiema infrastrukturami (co pozwala urządzeniom *infrastruktury A* na dostęp do urządzeń *infrastruktury B* i analogicznie w drugą stronę tak samo).

## 2.3. Konfiguracja środowiska testowego

Konfiguracja właściwego środowiska testowego jest kluczowa dla osiągnięcia prawidłowych wyników w testowaniu oprogramowania. W niniejszej pracy infrastruktura jest podzielona na dwie części – *infrastrukturę A* i *infrastrukturę B*.

W *infrastrukturze A* znajduje się:

### 1. serwer NAS

The screenshot displays the Synology DSM 'Panel sterowania' (Control Panel) interface. The left sidebar contains navigation options: 'Szukaj', 'Użytkownik', 'Grupa', 'Domena/LDAP', 'Łączność', 'QuickConnect', 'Dostęp zewnętrzny', 'Sieć', 'DHCP Server', 'Bezpieczeństwo', 'System', and 'Centrum informacji'. The main panel shows the 'Ogólne' (General) tab with the following information:

Podstawowe informacje	
Numer seryjny	C3J1N00883
Nazwa modelu	DS411
Procesor	MARVELL Kirkwood 88F6282
Częstotliwość taktowania procesora	1.6 GHz
Rdzenie procesora	1
Łącznie pamięci fizycznej	512 MB
Wersja oprogramowania DSM	DSM 6.2.4-25556 Update 6
Czas systemu	06.04.2023 02:39:14
Czas operacyjny	1 dni 10 Godz 7 minut(a) 49 sek
Stan temperatury systemu	● Normalny

Czas	
Adres serwera	time.google.com
Strefa czasowa	(GMT+01:00) Amsterdam, Berlin, Rome, Stockholm, Vi...

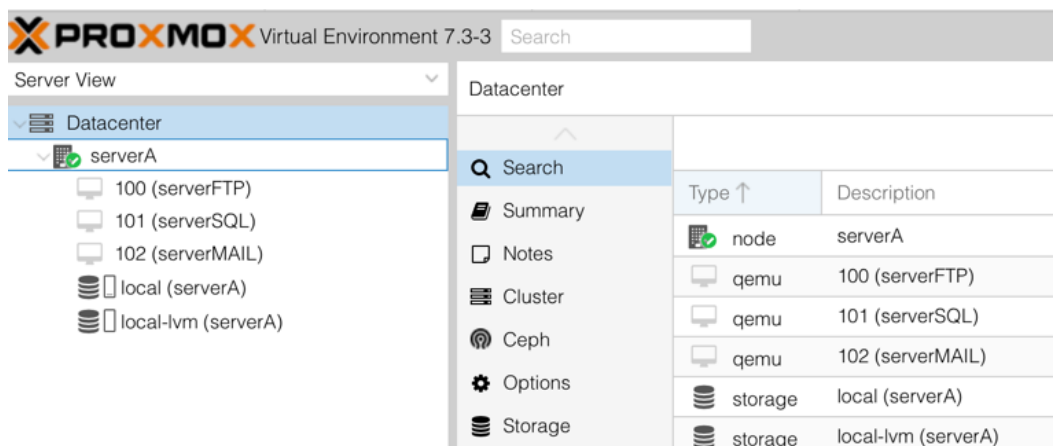
Urządzenia zewnętrzne	
Brak podłączonych urządzeń zewnętrznych.	

Rys. 3. Centrum informacji o serwerze NAS

Całość konfiguracji umieszczona została w pliku konfiguracja NAS.dss w folderze *config* repozytorium GitHub.



2. serwer wirtualizacji Proxmox, na którym znajdują się trzy maszyny wirtualne (serverFTP, serverSQL, serverMAIL)




Rys. 4. Infrastruktura A w Proxmox

Całość konfiguracji umieszczona została w pliku serwer wirtualizacji A.conf w folderze *config* repozytorium GitHub.

Natomiast w *infrastrukturze B* znajdują się:

1. kamera Hikvision

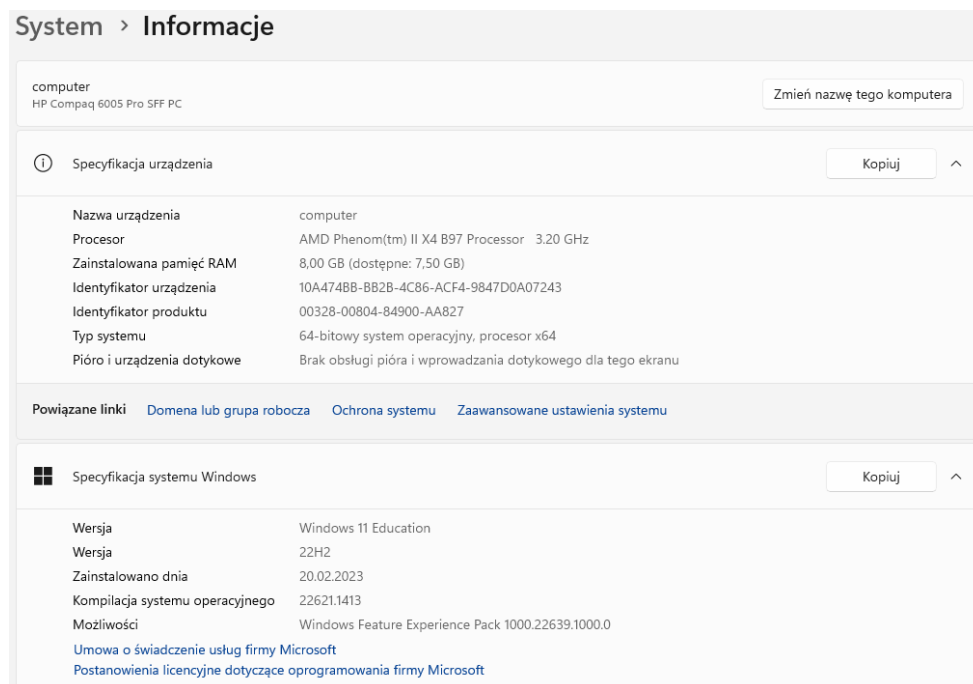
Podstawowe informacje	Ustawienia czasu	Czas letni	RS-232	O
Nazwa urządzenia	camera			
Nr urządzenia	88			
Model	DS-2CD1641FWD-I			
Nr seryjny	DS-2CD1641FWD-I20170325BBWR734894448			
Wersja firmware	V5.5.82 build 190909			
Wersja kodowania	V7.3 build 190128			
Wersja dodatku WWW	V4.0.1 build 180905			
Wersja dodatku	V3.0.6.46			
Liczba kanałów	1			
Liczba dysków	0			
Liczba wejść alarmowych	0			
Liczba wyjść alarmowych	0			
Własność oprogramowan...	B-R-R6-0			

 Zapamiętaj

Rys. 5. Informacje o kamerze Hikvision

Kamera poza aktywacją urządzenia i nadaniem jej poświadczeń logowania nie posiada zmian w zakresie konfiguracji w stosunku do ustawień fabrycznych.

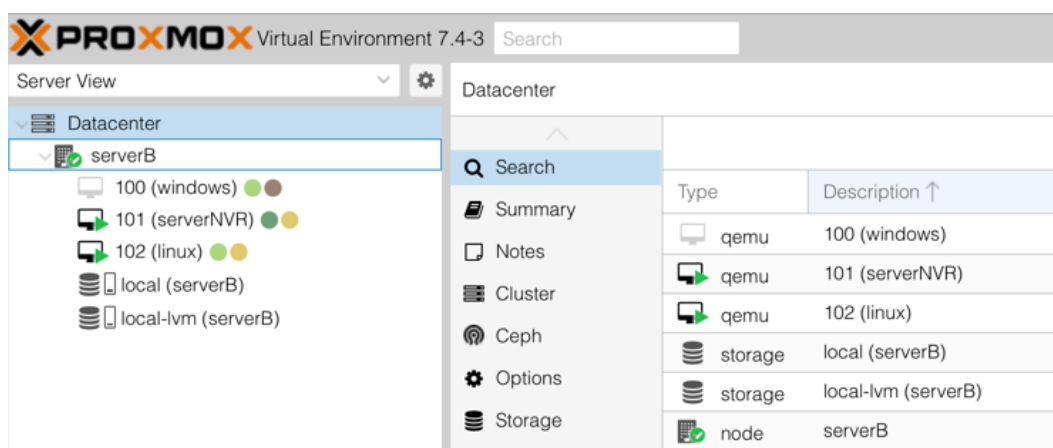
## 2. komputer stacjonarny z systemem Windows



Rys. 6. Informacje o fizycznym komputerze

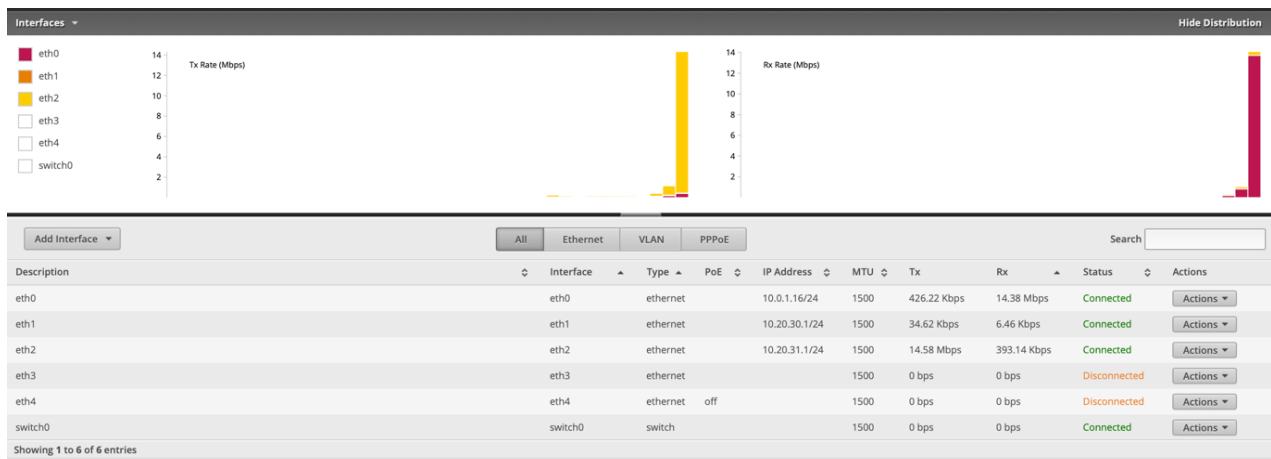
System zainstalowany na bazie licencji edukacyjnej w ramach nadanego przez uczelnię dostępu do systemu Microsoft Azure.

## 3. serwer wirtualizacji Proxmox, którym również uruchomione są trzy maszyny wirtualne (serverNVR, Windows Desktop, Linux Desktop)



Rys. 7. Infrastruktura B w Proxmox

Całość konfiguracji umieszczona została w pliku serwer wirtualizacji B.conf w folderze *config* repozytorium GitHub.



**Rys. 8. Dashboard router**

Całość konfiguracji umieszczona została w pliku `router.tar.gz` w folderze `config` repozytorium GitHub. Konfiguracja sieciowa (adresacja urządzeń) obu infrastruktur realizowana jest przez *router*, zgodnie z w/w specyfikacją logiczną infrastruktury, a środowisko testowe korzysta z dostępu do sieci Internet przez port WAN (interfejs `eth0`) jako *dhcp client*.

### 3. Testy osiągalności

Testy osiągalności są jednym podstawowych i kluczowych elementów w procesie ciągłego testowania infrastruktury i oprogramowania w systemach monitoringu IT. Ich poprawna konfiguracja, wdrożenie a następnie procesowe i szczegółowe podejście do ewentualnych zgłoszeń z nich wynikających, posiada bezpośrednie przełożenie na czas dostępności infrastruktury, przez jej odbiorców, klientów bądź użytkowników. Często stanowi to także element decydujący w odniesieniu do realizacji (bądź naruszenia) warunków umownych w zakresie dostępności infrastruktury – SLA.

#### 3.1. Wprowadzenie

Narzędzie ping funkcjonujące w trzeciej warstwie sieciowej wg. modelu OSI, oparte na protokole ICMPv4 lub ICMPv6, w zależności od wersji wykorzystywanego protokołu jest niewątpliwie najbardziej popularnym i powszechnym narzędziem diagnostycznym na świecie. Znaczna część użytkowników sieci IT, także ta nie będąca ani nie pracująca w sektorze IT miała styczność z tym narzędziem w swoim życiu. Uniwersalność nazw jak i sposobów wykorzystania tego narzędzia niezależnie od systemu operacyjnego bądź środowiska czyni je na swój sposób wyjątkowym. Nazwa narzędzia stanowi onomatopeiczne odzwierciedlenie dźwięku jaki wydaje sonar po otrzymaniu sygnału zwrotnego do radaru (co jednocześnie w sposób idealny obrazuje mechanizm funkcjonowania tego narzędzia).

Dzięki temu narzędziu można szybko i łatwo sprawdzić, czy urządzenia wykorzystywane w infrastrukturze są dostępne w sieci. W przypadku, gdy ping nie jest w stanie nawiązać połączenia z danym urządzeniem, oznacza to, że urządzenie jest niedostępne lub występują problemy z połączeniem.

#### 3.2. Przykłady zastosowań biznesowych

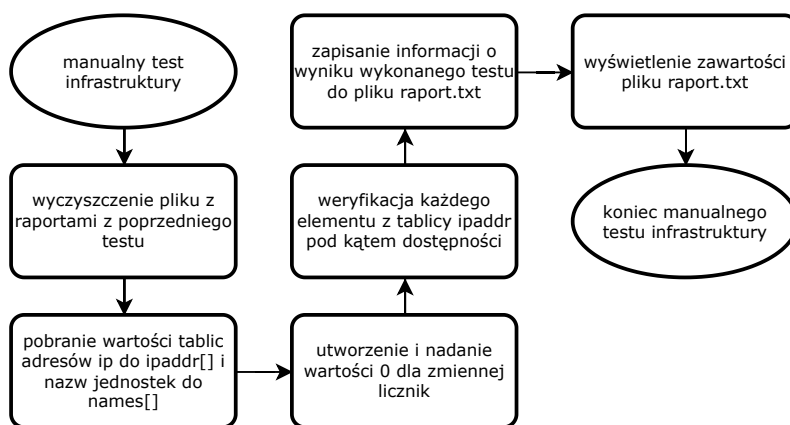
Przykład 1. (wpływ systemu monitoringu dostępności IT na bezpieczeństwo i zdrowie pracowników organizacji):

*Firma logistyczna, transportowa i magazynowa funkcjonująca standardowo w zakresie pracy biurowej i administracyjnej w godzinach 07:00 – 19:00, po momencie zamknięcia pozostaje pod nadzorem pracownika ochrony. W godzinach nocnych (ok. godziny 03:30) dochodzi do wtargnięcia na teren firmy przy jednoczesnym obezwładnieniu i unieruchomieniu wspomnianego pracownika ochrony. Chaotyczność oraz brak wiedzy włamywaczy w odniesieniu do konstrukcji systemów*

monitoringu przemysłowego przekłada się na zniszczenie dwóch kamer wizyjnych wraz z komputerem (w którym agresorzy zakładali, że dokonywany jest zapis nagrań z monitoringu), znajdującym się w pomieszczeniu ochrony przy wjeździe do obiektu. System monitoringu IP analogiczny do przedstawionego poniżej, wskutek braku kontaktu z wymienionymi wyżej urządzeniami zgłasza ich uszkodzenie do centrum monitoringu firmy z zakresu outsourcingu IT. Zgodnie ze standardową procedurą w takiej sytuacji, podejmowana jest próba kontaktu z pracownikiem ochrony i poinformowania go o awarii oraz dokonanie wywiadu wstępnego na temat potencjalnej przyczyny jej powstania. Brak możliwości kontaktu z pracownikiem ochrony przez okres 15 minut i niestandardowy charakter awarii w obiekcie skutkuje podjęciem decyzji przez pracownika dyżurującego w firmie IT o poinformowaniu policji i pogotowia ratunkowego o zaistniałej sytuacji i potencjalnym zagrożeniu. Włamywacze po obezwładnieniu i unieruchomieniu ochroniarza podjęli czasochłonną próbę sforsowania znajdującego się w biurze zarządu sejfu o wysokości 1,2 m oraz wadze 280 kg przy użyciu szlifierki kątowej. Służby po pojawieniu się na miejscu i zorientowaniu się w rzeczywistej sytuacji uwalniają pracownika ochrony, który na szczęście nie odniósł poważniejszych urazów i udają się do siedziby firmy, gdzie dokonują zatrzymania osób odpowiedzialnych za całe zdarzenie.

Całość opisanych powyżej sytuacji jest odzwierciedleniem zdarzeń faktycznych, które dzięki sprawnie funkcjonującym testom dostępności przyczyniły się do zapewnienia bezpieczeństwa pracownikowi ochrony oraz zatrzymania i ukarania osób dokonywujących rozboju na tle majątkowym.

### 3.3. Testy manualne



Rys. 9. Schemat blokowy skryptu wykonania testów manualnych w testach osiągalności

W przypadku testu manualnego, stworzone narzędzie pozwala w sposób natychmiastowy i doraźny na ocenę stanu faktycznego infrastruktury, a także na obserwowanie zmian jakie w niej zachodzą pod wpływem naszym działań. Narzędzie takie może być pomocne nie tyle w samym wykryciu awarii jak w diagnostyce powodu jej powstania bądź sposobu jej rozwiązania. Pozwala ono także na ocenę wpływu poszczególnych zmian konfiguracyjnych na całościową dostępność wszystkich jednostek wchodzących w skład danej infrastruktury bądź jej części. Powyższy schemat blokowy jest graficzną prezentacją kolejności i zależności realizowanych w ramach skryptu procesów.

```
#!/bin/bash

# PL: czyścimy plik z raportami z poprzednich informacji
# EN: clear the report file before start this script from previous results
> raport.txt

# PL: pobieramy wartości adresów IP z pliku infB.txt do zmiennej typu tablica o nazwie ipaddr
# EN: read the IP addresses from the file infB.txt into the variable type tabel
readarray -t ipaddr < infB.txt

# PL: pobieramy wartości nazw urządzeń w sposób analogiczny odpowiadających ich adresom IP z
# EN: read the names of the units from the file infB_names.txt into the variable type tabel named
# "names", in analogy to ip addresss
readarray -t names < infB_names.txt

# PL: ustawiamy zmienną licznik do wartości 0 czyli pierwszego elementu w tablicy names, aby móc
# EN: set the variable "count" to zero, so that the first element of the array "names", because
# we will generate the report with the IP address and the unit name
licznik=0

# PL: Wykorzystujemy pętlę for dla weryfikacji stanu każdego wcześniej zdefiniowanego adresu IP
# EN: We use the for loop to calculate the status of all defined previous IP addresses ping tool
for i in ${ipaddr[@]}; do
    if ping -c1 "$i" >/dev/null 2>&1;
    then
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Jednostka ${names[licznik]} o adresie IP ${i} jest dostępna" >> "raport.txt"
        licznik=$((licznik+1))
    else
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Jednostka ${names[licznik]} o adresie IP ${i} jest niedostępna" >>
"raport.txt"
        licznik=$((licznik+1))
    fi
done

date=$(date +"%Y-%m-%d %T")
echo ""
echo "raport wykonanych testów z daty ${date}"
echo ""
cat raport.txt
```

**Listing 1.** Kod źródłowy skryptu wykonania testów manualnych w testach osiągalności

Powyżej umieszczony autorski kod źródłowy typu *shell* na powłoce typu *bash* bazuje na dwóch plikach pomocniczych: *infB.txt* i *infB\_names.txt*. Pierwszy plik zawiera adresy IP testowanych

urządzeń, a drugi zawiera ich nazwy bądź funkcje (celem ułatwienia i zwiększenia użyteczności oprogramowania).

Przed wykonaniem testu, plik *raport.txt* jest czyszczony, aby uniknąć powielenia wyników z poprzednich testów. Każda linia pliku *raport.txt* zawiera informacje o czasie wykonania testu, nazwie urządzenia i jego adresie IP, a także o tym, czy urządzenie jest dostępne czy też nie. Po zakończeniu testu, skrypt wyświetla całościowe wyniki.

```
[tatiana@linuxdesktop:~/skrypty$ ./check.sh
```

```
raport wykonanych testów z daty 2023-03-31 01:34:12
```

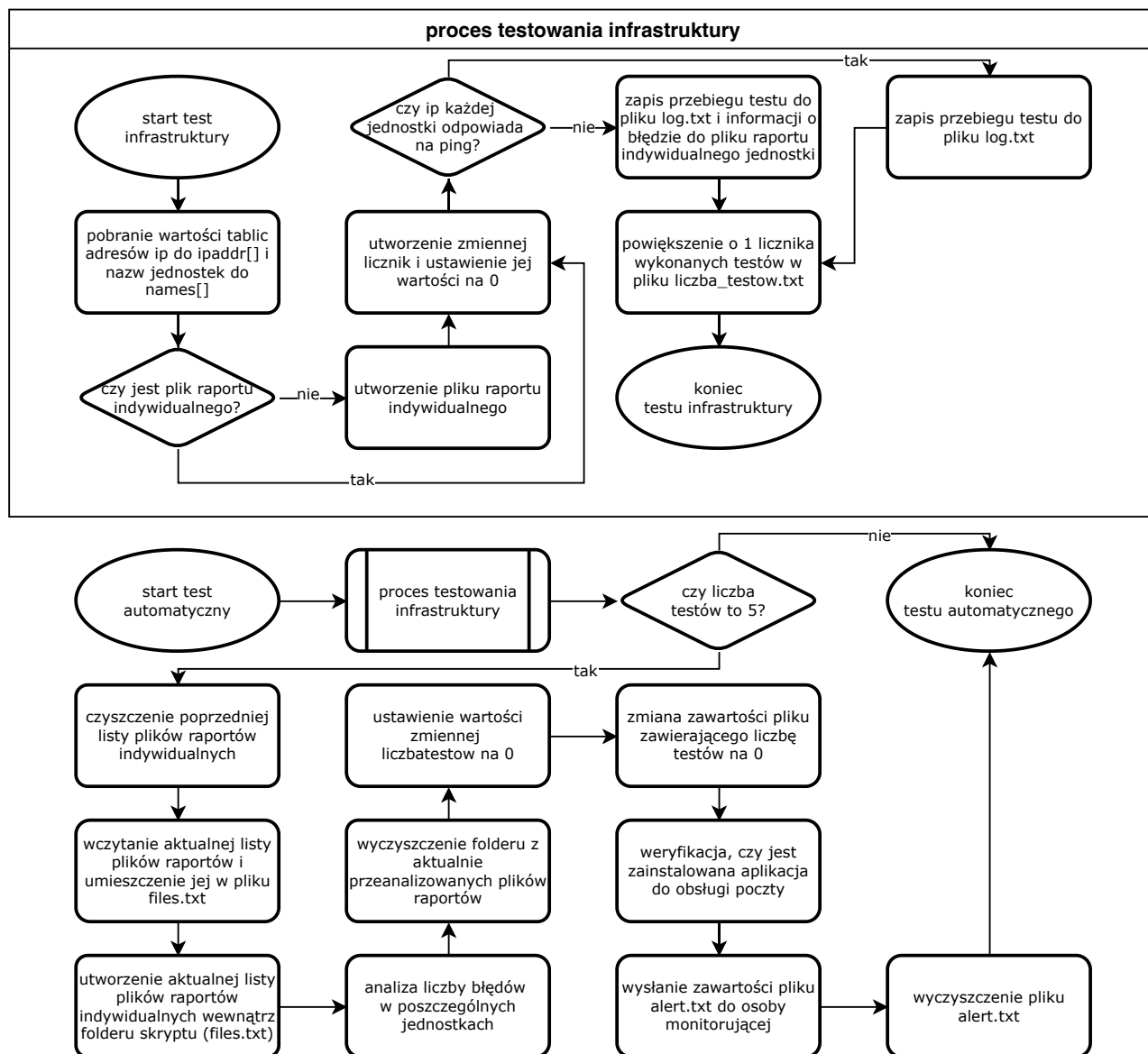
```
2023-03-31 01:34:09 Jednostka serverA-eth1 o adresie IP 10.20.31.4 jest dostępna
2023-03-31 01:34:09 Jednostka router o adresie IP 10.20.30.1 jest dostępna
2023-03-31 01:34:12 Jednostka camera o adresie IP 10.20.30.2 jest niedostępna
2023-03-31 01:34:12 Jednostka computer o adresie IP 10.20.30.3 jest dostępna
2023-03-31 01:34:12 Jednostka server-iDRAC o adresie IP 10.20.30.4 jest dostępna
2023-03-31 01:34:12 Jednostka serverB-eth0 o adresie IP 10.20.30.5 jest dostępna
2023-03-31 01:34:12 Jednostka windows-desktop o adresie IP 10.20.30.6 jest dostępna
2023-03-31 01:34:12 Jednostka server-nvr o adresie IP 10.20.30.7 jest dostępna
2023-03-31 01:34:12 Jednostka linux-desktop o adresie IP 10.20.30.8 jest dostępna
```

**Rys. 10. Prezentacja wyniku testu manualnego w testach osiągalności**

Wyniki testu w celu prezentacji zostały wyświetlone w terminalu po zakończeniu działania skryptu *check.sh* testującego. W raporcie zawarta jest lista wszystkich testowanych jednostek wraz z informacją o ich dostępności w momencie przeprowadzenia testu.

Na potrzeby niniejszego testu w sposób świadomy i zamierzony wyłączona została kamera IP wchodząca w skład testowanej infrastruktury co następnie czytelnie wykazane zostało w powyższym raporcie.

### 3.4. Testy automatyczne



Rys. 11. Schemat blokowy skryptu wykonania testów automatycznych w testach osiągalności

Przedstawiony schemat blokowy stanowi graficzną prezentację sposobu funkcjonowania utworzonego kodu, którego celem jest realizacja testów w określonych interwałach czasowych w sposób automatyczny i autonomiczny wraz z okresową analizą ich wyników oraz przesłaniu raportu na temat istniejących uszkodzeń o ile takowe występują.

```

#!/bin/bash

# PL: przejdź do folderu aplikacji celem dostępu do plików roboczych
# EN: Go to the folder where the application is and execute the script
cd /home/tatiana/skrypty

# PL: pobieramy wartości adresów IP z pliku infB.txt do zmiennej typu tablica o nazwie ipaddr
# EN: read the IP addresses from the file infB.txt into the variable type tabel
readarray -t ipaddr < infB.txt
date=$(date +"%Y-%m-%d %T")

```



```

echo "${date} wczytano tablicę adresów IP z pliku infB.txt do zmiennej ipaddr" >> log.txt

# PL: pobieramy wartości nazw urządzeń w sposób analogiczny odpowiadających ich adresom IP z
# EN: read the names of the units from the file infB_names.txt into the variable type tabel named
# EN: "names", in analogy to ip addresss
readarray -t names < infB_names.txt
date=$(date +"%Y-%m-%d %T")
echo "${date} wczytano tablicę nazw urządzeń do zmiennej names" >> log.txt

# PL: weryfikujemy, czy dla każdej maszyny istnieje plik służący do raportowania błędów
# EN: check if there is a file for all units in array "names" named "rap_*" in the directory
for i in ${names[@]}; do
    if [ -f "rap_${i}.txt" ];
    then
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Plik raportu dla jednostki ${i} istnieje" >> log.txt
    else
        touch "rap_${i}.txt"
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Plik raportu dla jednostki ${i} nie istniał i został utworzony" >> log.txt
    fi
done

# PL: ustawienie zmiennej licznik na wartość zero celem analizy tablicy od jej pierwszego
# EN: set the variable named "licznik" to zero for analyse table from first element
licznik=0
date=$(date +"%Y-%m-%d %T")
echo "${date} ustawiono wartość zmiennej licznik" >> log.txt

# PL: weryfikacja dostępności poszczególnych jednostek z listy adresów IP
# EN: checke the availability of all units in the array named "ipaddr"
for i in ${ipaddr[@]}; do
    echo "rozpoczynam analizę jednostki ${i}" >> log.txt
    if ping -c1 "${i}" >> log.txt;
    then
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Jednostka ${names[licznik]} o adresie ip ${i} jest dostępna" >> log.txt
        licznik=${licznik}+1
    else
        date=$(date +"%Y-%m-%d %T")
        echo "${date} Jednostka ${names[licznik]} jest niedostępna" >> "rap_${names[licznik]}.txt"
        echo "Jednostka ${names[licznik]} o adresie ip ${i} jest niedostępna" >> log.txt
        licznik=${licznik}+1
    fi
done

# PL: utworzenie zmiennej liczbatestow i wczytanie jej wartości z pliku liczba_testow.txt
# EN: create the variable named "liczba_testow" and read its value from the file
liczba_testow.txt
declare -i liczbatestow
liczbatestow=$( cat liczbatestow.txt )
liczbatestow=${liczbatestow}+1
> liczbatestow.txt
echo $liczbatestow >> liczbatestow.txt

# PL: weryfikacja czy liczba odbytych testów automatycznych jest większa bądź równa liczbie 5
# EN: check if the number of tests is greater or equal to 5
if [ $liczbatestow -ge 5 ];
then
    #PL: czyszczenie pliku files.txt z poprzednią listą plików raportów indywidualnych
    #EN: cleaning the contents of files.txt from the previous run
    > files.txt
    date=$(date +"%Y-%m-%d %T")
    echo "${date} wyczyszczono zawartość pliku files.txt" >> log.txt

    # PL: wczytujemy aktualną listę plików raportów indywidualnych i umieszczamy ją w pliku
    # EN: read the current list of indydwidual raport files and put them in file files.txt
    files.txt
    find ./ -maxdepth 1 -type f -name "rap_*" -printf '%f\n' >> files.txt
    date=$(date +"%Y-%m-%d %T")
    echo "${date} wczytano listę plików raportów indywidualnych do pliku files.txt" >> log.txt

    # PL: wczytujemy stworzoną listę plików raportów indywidualnych i umieszczamy ją w zmiennej
    # EN: read the created list of idydwidual raport files and put them in files

```

```

readarray -t files < files.txt
date=$(date +%Y-%m-%d %T")
echo "${date} wczytano listę plików raportów indywidualnych do zmiennej files" >> log.txt

# PL: analizujemy liczbę błędów w poszczególnych jednostkach w sieci przez zliczenie liczby
raportów błędnie wykonanych testów ping
# EN: analyze the number of errors, counting lines in each of the individual report
declare -i err
for plik in ${files[@]}; do
    err=$( wc -l < $plik )
    err=${err%% *}
    if [ $err -ge 5 ]; then
        jednostka="${plik%???}"
        jednostka="${jednostka#???}"
        echo "!!! ALARM !!! Jednostka ${jednostka} jest niedostępna" >> alert.txt
    fi
done

# PL: czyścimy folder z aktualnie przeanalizowanych plików raportów indywidualnych
# EN: clean the folder with the files of actual analysed individual report files
rm rap_*.txt

#PL: ustawiamy wartość zmiennej liczbatestow na 0
#EN: set the number of a variable named "liczbatestow" to 0
liczbatestow=0

# PL: zmieniamy zawartość pliku zawierającego liczbę przeprowadzonych testów na 0
# EN: change the contents of the file files.txt to 0
> liczbatestow.txt
echo $liczbatestow >> liczbatestow.txt

#PL: zweryfikuj czy jest zainstalowana aplikacja do obsługi poczty
#EN: check if the application too send a email is already installed in system
if [ -x "$(command -v swaks)" ];
then
    # PL: wysyłamy zawartość pliku alert.txt do osoby monitorującej
    # EN: send the contents of the file alert.txt to the person who work in monitoring
    alert=$(cat alert.txt)
    echo ${alert}
    swaks --body ./alert.txt --to tetiana.w[REDACTED]@gmail.com --from
tetiana.h[REDACTED]@wp.pl --server smtp.wp.pl --port 587 --auth LOGIN --auth-user tetiana.h[REDACTED]@wp.pl
--auth-password [REDACTED]
else
    echo "w systemie nie zainstalowano aplikacji do obsługi poczty swaks, błędne elementy
infrastruktury zostaną wyświetlone na ekranie"
    echo " "
    if [ $(cat alert.txt) = "" ];
    then
        echo "cała sieć funkcjonuje poprawnie"
    else
        echo "$(cat alert.txt)"
    fi

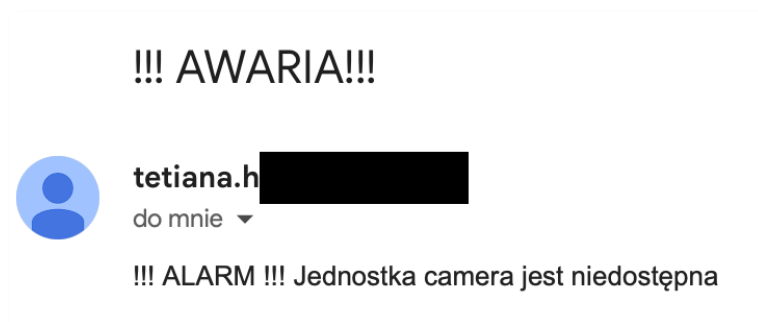
    #PL: czyścimy plik z alertami
    #EN: clean the file alert.txt
    echo ${alert}
    > alert.txt
fi
fi

```

**Listing 2. Kod źródłowy skryptu wykonania testów automatycznych w testach osiągalności**

Skrypt analogicznie jak w przypadku testów manualnych wykorzystuje pliki konfiguracyjne zawierające adresy monitorowanych urządzeń oraz ich nazwy. Skrypt wykorzystuje także pliki robocze „raporty indywidualne” w których dla każdej z testowanych jednostek zapisywane są informacje o nie wykryciu tej jednostki w infrastrukturze. Ostatecznie skrypt dokonuje automatycznej walidacji wyników polegającej na analizie ilości negatywnych testów dla każdego z urządzeń. O ile urządzenie w 5 kolejnych testach wykazało niedostępność informacja taka

przekazywana jest do zbiorczego raportu o awariach w infrastrukturze w pliku *alert.txt*. Mechanizm ten zabezpiecza nas na powstawanie fałszywie pozytywnych raportów o awarii danego urządzenia w wyniku nieprawidłowości pojedynczego testu. Urządzenie raportowane jest jako niedostępne wyłącznie w przypadku 5 negatywnych i następujących po sobie testach. Ostatecznie plik *alert.txt* podlega weryfikacji i o ile zawiera on jakiekolwiek informacje, jego zawartość przesyłana jest drogą mailową, do osoby odpowiedzialnej za monitoring infrastruktury.







**Rys. 12. Prezentacja wyniku testu automatycznego w testach osiągalności**

Podobnie jak w przypadku testów manualnych, w przypadku testów automatycznych uszkodzeniu również podlegało jedno urządzenie sieciowe, to znaczy kamera IP. Powyżej zaprezentowany został rezultat działania testów automatycznych w postaci zrzutu ekranu pojedynczej wiadomości e-mail, będącej alarmem dla potencjalnej osoby monitorującej i odpowiedzialnej za utrzymanie ciągłości funkcjonowania infrastruktury IT.

```
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow  command  
  
* * * * * /home/tatiana/skrypty/check_auto.sh
```

**Rys. 13. Konfiguracja Crona na wykonywanie testów w odstępie czasowym 1min. w testach osiągalności**

Skrypt *check\_auto.sh* zostaje uruchamiany w ramach zadania cron, które jest zaplanowane do uruchamiania co minutę (z gwiazdkami w polach czasowych crontab - \* \* \* \* \*).

	tetiana.h [REDACTED] !!! ALARM !!! Jednostka camera jest niedostępna	00:30 (16 godzin temu)
	tetiana.h [REDACTED] !!! ALARM !!! Jednostka serverA-eth1 jest niedostępna	03:00 (13 godzin temu)
	tetiana.h [REDACTED] !!! ALARM !!! Jednostka camera jest niedostępna !!! ALARM !!! Jednostka serverA-eth1 jest niedostępna	05:30 (11 godzin temu)
	tetiana.h [REDACTED] !!! ALARM !!! Jednostka camera jest niedostępna !!! ALARM !!! Jednostka serverA-eth1 jest niedostępna	08:00 (8 godzin temu)

**Rys. 14. Prezentacja wyniku testu automatycznego ustawionego na wykonywanie co określony czas**

Powyżej zaprezentowano efekty działania skryptu w konfiguracji 1 test na 30 minut - w przypadku konfiguracji Cron wymaga to następująco wyglądającej konfiguracji:

```
*/30 * * * * /home/tatiana/skrypty/check_auto.sh,
```

a co za tym idzie 5 pełnych testów realizowanych było przez dwie i pół godziny, i dokładnie w takich odstępach czasowych wysyłany był raport o awarii. Warto pamiętać, że najkrótszym interwałem czasowym jaki można skonfigurować przy wykorzystaniu usługi Cron to 1 minuta, czyli minimalny okres raportowania i realizacji 5 pełnych testów to 5 minut. W zależności od potrzeb zatem możemy skonfigurować usługę Cron w taki sposób, aby dostarczała nam raporty wiarygodne, a jednocześnie nie powodowała nadmiaru wysyłanych informacji drogą mailową na skrzynkę alertową.

## 4. Testy sprawnościowe

Testy sprawnościowe są kolejnym istotnym elementem w procesie ciągłego testowania infrastruktury i oprogramowania w systemach monitoringu IT. Ich celem jest sprawdzenie systemu w określonych warunkach, np. czy temperatura jest poprawna lub czy system jest aktualny. Testy te pozwalają na zidentyfikowanie błędów w systemie.

### 4.1. Wprowadzenie

W kodzie skryptu *bash* zostaną użyte narzędzia SNMP do odczytywania i wyświetlania różnych wartości. Skrypt będzie działał w konsoli, gdzie użytkownik będzie miał do wyboru kilka opcji. Skrypt będzie interaktywny, a użytkownik będzie miał możliwość powrotu do menu głównego po każdym odczycie.

### 4.2. Przykłady zastosowań biznesowych

Przykład 1. (redukcja i minimalizacja ryzyka eskalacji strat przy wykorzystaniu testów sprawnościowych):

*Przedsiębiorstwo z województwa opolskiego, specjalizujące się w termoformowaniu niskonakładowym, posiada złożoną infrastrukturę informatyczną, wspierającą i współpracującą z wieloma urządzeniami do cyfrowej obróbki termicznej i mechanicznej. Kluczowymi elementami dla przedsiębiorstwa w zakresie jego funkcjonowania są projekty, opracowania i specyfikacje indywidualnych form próżniowych do kształtowania polipropylenu. Ich stworzenie wymaga znacznego nakładu pracy i kosztownych badań. Dlatego projekty te, po ukończeniu przechowywane są na dedykowanym urządzeniu NAS z macierzą 1+0. Urządzenie natomiast, umieszczone zostało w przystosowanym na ten cel, zabezpieczonym i odizolowanym pomieszczeniu.*

*O godzinie 22:40 osoba zajmująca się utrzymaniem i administracją IT na rzecz tej firmy, otrzymała informację, o wystąpieniu awarii macierzy RAID. Uszkodzeniu uległ jeden z czterech wykorzystywanych dysków twardych, a komunikat o tym dotarł do administratora już w 7 minut po wystąpieniu awarii, wyłączeniu oraz resynchronizowaniu macierzy. Jeszcze tego samego dnia, zadysponowany został tam serwisant, który korzystając z zakupionego wcześniej i przygotowanego dysku awaryjnego, dokonał jego wymiany i rozpoczął proces resynchronizacji macierzy RAID. Warto zaznaczyć tutaj, iż separacja i izolacja tego pomieszczenia sprawiła, że dźwięk pochodzący z urządzenia NAS świadczący o błędzie macierzy jest praktycznie niedostrzegalny, i niezauważalny dla personelu firmy.*

Firma stosując mechanizm testów sprawnościowych (o logice działania analogicznej do przedstawionej w niniejszej pracy), zniwelowała do minimum czas naprawy i przywrócenia macierzy do pełnej sprawności. Dodatkowo wyeliminowała problem dostrzeżenia i zgłoszenia awarii przez personel firmy. Finalnie znacząco zredukowało to ryzyko eskalacji strat i konsekwencji, jakie wiązałyby się z utratą danych zgromadzonych na macierzy danych.

### 4.3. Testy manualne

Funkcja testowania w obszarze administracji siecią i infrastrukturą IT, ma swoją charakterystykę i właściwości, które sprawiają, że narzędzia służące do testów manualnych zademonstrowane w niniejszej pracy, nie są jedynie manualną formą przeprowadzenia poszczególnych przypadków testowych, a mają swoją odrębną użyteczność i przeznaczenie. Testy manualne w administracji IT są narzędziem informacyjnym, diagnostycznym, wspomagającym tak serwis jak i administrację danej infrastruktury, przez uzyskiwanie bieżącej informacji zwrotnej i możliwość obserwacji afektów naszych działań. Testy automatyczne natomiast są systemami monitorującymi i nadzorującymi poprawność działania infrastruktury w sposób ciągły.

Przedstawione poniżej manualne testy sprawnościowe, pozwalają na personalizację i dostosowanie funkcjonalności pod konkretne potrzeby i zastosowania administracyjne. Dodatkowo pozwalają na działanie scentralizowane na wszystkich obsługujących protokołów SNMP i ujednolicone w zakresie interfejsu.

```
#!/bin/bash
cd /home/tatiana/skrypty/snmp
clear
echo "*****"
echo "* Konsola testów manualnych sprawności sieci *"
echo "*****"

PS3="Wybierz urządzenie, które chcesz sprawdzić za pomocą protokołu snmp:"
select device in "serwer nas" "serwer wirtualizacji A" "koniec"
do
    case $device in
        "serwer nas")
            bash snmp_nas.sh ;;
        "serwer wirtualizacji A")
            bash snmp_serwerA.sh ;;
        "koniec")
            echo "Testy manualne zostały zakończone"
            break ;;
        *)
            echo "wybrałeś nieprawidłową opcję" ;;
    esac
done
```

**Listing 3. Kod źródłowy skryptu wykonania testu manualnego w testach sprawnościowych**

Kod programu stanowi skrypt konsoli *shell* z powłoką *bash*. Pozwala administratorom dokonywać analizy stanu bieżącego określonego typu parametru będącego w obszarze zainteresowania działań tak administracyjnych jak serwisowych. Dodatkowym walorem tych testów jest możliwość realizacji ich w dowolnie wybranym przez nas interwale oraz obserwować zmiany w nim zachodzące pod wpływem naszych działań. Są szczególnie użyteczne w sytuacji, kiedy chcemy w możliwie krótkim czasie dokonać trafnej i punktowej diagnozy uszkodzenia.

```
*****
*
* Konsola testów manualnych sprawności sieci *
* autorstwa Tatiany Hołycz                      *
*
*****

1) serwer nas
2) serwer wirtualizacji A
3) koniec
Wybierz urządzenie, które chcesz sprawdzić za pomocą protokołu snmp:
```

**Rys. 15. Obraz konsoli skryptu snmp.sh po uruchomieniu testów manualnych dla sprawności**

Wskazany powyżej zrzut ekranu jest niezwykle prostą, jednocześnie czytelną prezentacją zalet i potencjału jaki stanowią manualne testy sprawnościowe. Jak widać pozwala on nam na odwołanie się do więcej niż jednej jednostki sieciowej w obrębie jednej aplikacji. Dodatkowo możliwość uruchomienia skryptu w powłoce *shell* w wersji tekstowej, czyni to narzędzie niezwykle łatwo adaptowalnym dla większości powszechnie dostępnych systemów operacyjnych.

Dla uproszczenia w prezentowanym przykładzie wybór poszczególnych opcji stanowi odwołanie do osobnych skryptów, w których uzyskujemy dostęp do analizy parametrów. Dzięki temu rozwiązanie to przyjmuje formę konsoli głównej z sub-skryptami typu *plug-in*, zlokalizowanymi w tym samym folderze.

```
#!/bin/bash
cd /home/tatiana/skrypty/snmp
clear
echo "*****"
echo "* Konsola testów manualnych sprawności sieci *"
echo "* serwer nas                                     *"
echo "*****"
PS3="wybierz czujnik którego wartość chcesz otrzymać:"

select czujnik in "temperatura HDD" "stan chłodzenia" "stan HDD" "aktualizacje" "koniec"
do
    case $czujnik in
        "temperatura HDD")
```

```

HDD1T=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.6.0)
HDD2T=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.6.1)
echo " "
echo "aktualna temperatura dysku twardego nr 1 to: ${HDD1T}"
echo " "
echo "aktualna temperatura dysku twardego nr 2 to: ${HDD2T}"
echo " "
echo "Naciśnij dowolny klawisz aby powrócić do menu czujników"
read -n 1 -s -r key
bash snmp_nas.sh ;;

"stan chłodzenia")
cpufan=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.1.4.2.0)
sysfan=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.1.4.1.0)
if [ $cpufan == "1" ]
then
echo "chłodzenie procesora działa poprawnie"
echo " "
else
echo " "
echo "chłodzenie procesora działa niepoprawnie"
echo " "
fi
if [ $sysfan == "1" ]
then
echo "chłodzenie systemu działa poprawnie"
echo " "
else
echo " "
echo "chłodzenie systemu działa niepoprawnie"
echo " "
fi
read -n 1 -s -r key
bash snmp_nas.sh ;;

"stan HDD")
hdd1s=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.2.1.1.5.0)
hdd2s=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.2.1.1.5.1)
if [ $hdd1s == "1" ]
then
echo "aktualny stan dysku nr 1 jest poprawny"
echo " "
else
echo " "
echo "stan dysku nr 1 wymaga uwagi"
echo " "
fi
if [ $hdd2s == "1" ]
then
echo "aktualny stan dysku nr 2 jest poprawny"
echo " "
else
echo "stan dysku nr 2 wymaga uwagi"
echo " "
fi
read -n 1 -s -r key
bash snmp_nas.sh ;;

"aktualizacje")
sysupp=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.1.5.4.0)
if [ $sysupp == "2" ]
then
echo " "
echo "system operacyjny serwera nas jest aktualny"
echo " "
else
echo " "
echo "system operacyjny serwera nas jest nieaktualny"
echo " "
fi
read -n 1 -s -r key
bash snmp_nas.sh ;;

"koniec")
echo "testy manualne zostały zakończone"
bash snmp.sh ;;

*)

```



```
echo "wybrałeś nieprawidłową opcję" ;;  
esac  
done
```

**Listing 4. Kod źródłowy skryptu do wykonywania testów manualnych serwera NAS w testach sprawnościowych**

Skrypty testów manualnych dla poszczególnych urządzeń pozwalają personalizować treść i parametry, jakie będą wykorzystywane i użyteczne dla administratora. Możliwość ich personalizacji tzn. ich ilość, to czego one będą dotyczyć oraz w jakiej formie będą one przedstawiać uzyskane z agenta SNMP wartości, czyni prezentowane narzędzie wysoce użytecznym i ergonomicznym.

```
* serwer nas *  
*****  
  
1) temperatura HDD  
2) stan chłodzenia  
3) stan HDD  
4) aktualizacje  
5) koniec  
wybierz czujnik którego wartość chcesz otrzymać:
```

**Rys. 16. Obraz konsoli skryptu serwera NAS po jego uruchomieniu**

W prezentowanym przypadku, podczas analizy parametrów serwera NAS, kluczowymi wartościami dla administratora będą:

- temperatury poszczególnych dysków twardych, które mają bezpośredni wpływ na ich żywotność oraz bezawaryjność,
- stan chłodzenia całości urządzenia, określający, czy każdy z obecnych w urządzeniu wentylatorów jest sprawny, i działa z prawidłową prędkością,
- określenie czy wersja systemu jest aktualna, co znacząco ogranicza zagrożenia ze strony ataków zewnętrznych oraz podatności urządzenia na błędy i awarie.

Większość dedykowanych i profesjonalnych rozwiązań sieciowych posiada poza standardowymi parametrami SNMP, także dodatkowe elementy rozszerzające funkcjonalność i zastosowanie protokołu. Najczęściej również publikuje on na swojej stronie dokumentację związaną z dostępnymi „czujnikami” OID i znaczeniu ich wartości oraz publikuje bazę MIB, która pozwala nam się w prosty sposób poruszać po danych wewnątrz agenta SNMP.

```
1) temperatura HDD
2) stan chłodzenia
3) stan HDD
4) aktualizacje
5) koniec
[wybierz czujnik którego wartość chcesz otrzymać:1

aktualna temperatura dysku twardego nr 1 to: 32

aktualna temperatura dysku twardego nr 2 to: 31

Naciśnij dowolny klawisz aby powrócić do menu czujników
```

**Rys. 17. Prezentacja wyniku temperatury HDD serwera NAS**

Prezentowana powyżej ilustracja przedstawia rzeczywistą temperaturę dysków twardych, w momencie wykonywania testów. Dane te oczywiście dostępne są z poziomu konsoli zarządzania WEB danego urządzenia. Warto pamiętać, że administracja całością infrastruktury IT, wymaga zarządzania wieloma, często różnymi i pochodzącymi od różnych producentów urządzeniami sieciowymi. Otrzymanie pożądanych dla sprawności infrastruktury parametrów, wymaga każdorazowo procesu uwierzytelniania oraz wiedzy w którym miejscu danego interfejsu jest on obecny. Wszystko to wymaga znacząco większej ilości czasu i nakładu pracy, co czyni takie działanie znacznie mniej efektywnym w zestawieniu do prezentowanego tutaj narzędzia testowego.

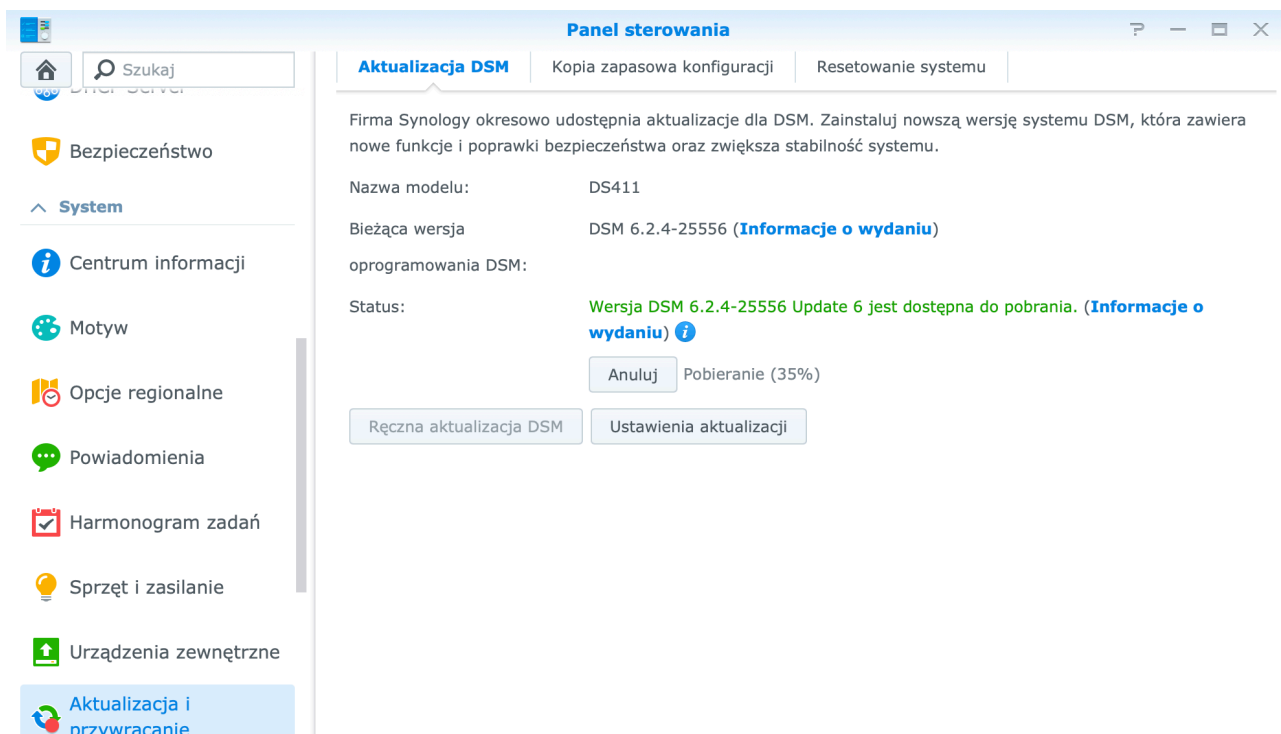
```
1) temperatura HDD
2) stan chłodzenia
3) stan HDD
4) aktualizacje
5) koniec
[wybierz czujnik którego wartość chcesz otrzymać:4

system operacyjny serwera nas jest nieaktualny
```

**Rys. 18. Prezentacja wyniku stanu aktualności serwera NAS**

Podobnie jak w przypadku testów osiągalności symulowanej i wykazanej awarii uległa kamera IP, tak w tym przypadku, system serwera NAS Synology zainstalowany został w wersji, która wymaga

aktualizacji. Na powyższym rysunku można zobaczyć, że w odpowiedzi na zapytanie do agenta SNMP wewnątrz urządzenia, uzyskujemy informację o nieaktualnej wersji systemu operacyjnego, co potwierdza się po weryfikacji na ilustracji widocznej poniżej.



**Rys. 19. Aktualizacja systemu DSM serwera NAS**

Prowadząc okresowy i cykliczny proces rewizji stanu infrastruktury, który pozwala na utrzymanie jej na odpowiednim poziomie bezpieczeństwa i sprawności, otrzymując informację zwrotną o aktualności systemu, dzięki prezentowanemu tutaj programowi, można pominąć manualną weryfikację, i przejść do dalszych urządzeń. Po otrzymaniu jednak informacji, o dostępnych dla danego systemu aktualizacjach, następuje proces ich instalacji i implementacji na urządzeniu, co zapewnia mu stabilność i bezpieczeństwo działania.

```

1) temperatura HDD
2) stan chłodzenia
3) stan HDD
4) aktualizacje
5) koniec
[wybierz czujnik którego wartość chcesz otrzymać:4

system operacyjny serwera nas jest aktualny

```

Rys. 20. Rezultat aktualizacji systemu operacyjnego serwera NAS

Po dokonanej aktualizacji urządzenia, przeprowadzany jest proces weryfikacji stanu danego parametru, aby upewnić się, iż działania przyniosły pożądany efekt, a następnie za pomocą aplikacji można zweryfikować stan kolejnego parametru bądź jak w tym przypadku, wybierając przycisk „koniec”, zamknąć operacje na danym urządzeniu i przejść do menu głównego z listą urządzeń.

```

#!/bin/bash
cd /home/tatiana/skrypty/snmp
clear
echo "*****"
echo "* Konsola testów manualnych sprawności sieci *"
echo "* serwer wirtualizacji proxmox A          *"
echo "*****"
echo " "
PS3="wybierz czujnik którego wartość chcesz otrzymać:"

select czujnik in "macierz RAID" "czas funkcjonowania systemu" "koniec"
do
    case $czujnik in
        "macierz RAID")
            RAIDN=$(snmpget -v 2c -c public -O qv 10.20.30.4 .1.3.6.1.4.1.674.10892.5.5.1.20.140.1.1.1.1)
            RAIDS=$(snmpget -v 2c -c public -O qv 10.20.30.4 .1.3.6.1.4.1.674.10892.5.5.1.20.140.1.1.4.1)
            echo " "
            echo "Ilość macierzy RAID w systemie to: ${RAIDN}"
            echo " "
            if [ $RAIDS == "2" ]
            then
                echo "Macierz RAID działa poprawnie i jest w stanie ONLINE"
            else
                echo "Macierz RAID nie działa poprawnie i należy sprawdzić jej sprawność"
            fi
            read -n 1 -s -r key
            bash snmp_serwerA.sh ;;
        "czas funkcjonowania systemu")
            uptime=$(snmpget -v 2c -c public -O qv 10.20.30.4 .1.3.6.1.4.1.674.10892.5.2.5.0)
            upptimes=$(( uptime % 60 ))
            modm=$(( upptimes % 3600 ))
            upptimem=$(( modm / 60 ))
            modh=$(( upptimes % 86400 ))
            upptimeh=$(( modh / 3600 ))
            modd=$(( upptimes - modh ))
            upptimed=$(( modd / 86400 ))
            echo "Czas aktywności systemu to ${upptimed} dni ${upptimeh} godzin ${upptimem} minut
${upptimes} sekund"
            echo "naciśnij dowolny klawisz aby kontynuować"
    esac
done

```

```

read -n 1 -s -r key
bash snmp_serwerA.sh ;;
"koniec")
echo "testy manualne zostały zakończone"
bash snmp.sh ;;
*)
echo "wybrałeś nieprawidłową opcję" ;;
esac
done

```

**Listing 5. Kod źródłowy skryptu wykonania testu manualnego serwera wirtualizacji A w testach sprawnościowych**

Celem pełnej wizualizacji i prezentacji idei i możliwości manualnych testów sprawnościowych, powyższy kod źródłowy skryptu powłoki *shell*, stworzony został w analogii do kodu analizy serwera NAS, i poddaje bieżącej analizie i weryfikacji, elementy pochodzące z serwera wirtualizacji Proxmox infrastruktury A. (agenta SNMP kontrolera iDrac zintegrowanego z platformą serwera). Warto przez to zauważyć, że stworzone wcześniej rozwiązania, jak i każde z poszczególnych elementów, wraz z rozwojem tego rodzaju aplikacji stają się swoistym „*framework’iem*” (zestawem gotowych narzędzi), przy dalszym rozwoju i rozbudowie w zakresie funkcjonalnym.

```

* serwer wirtualizacji proxmox A                                     *
*****

1) macierz RAID
2) czas funkcjonowania systemu
3) koniec
wybierz czujnik którego wartość chcesz otrzymać:

```

**Rys. 21. Obraz konsoli skryptu serwera A po jego uruchomieniu**

W przypadku tego rodzaju serwerów oczywistym przykładem i jedną z najważniejszych kwestii w zakresie utrzymania sprawności, jest weryfikacja stanu macierzy dyskowych. Dodatkowo skrypt zawiera przykład złożonej konwersji danych otrzymywanych z agenta SNMP jakim jest czas funkcjonowania systemu (*uptime*). Czujnik OID agenta iDrac zwraca wartość typu *integer* równą ilości sekund, które minęły sumarycznie od momentu uruchomienia systemu. W przeciwieństwie do poprzednich sytuacji tutaj, aby podać zwrotnie czytelną wartość w postaci ilości dni, godzin, minut i sekund, niezbędnym było przeprowadzenie serii operacji na zmiennych. Wiedza na temat czasu funkcjonowania systemu od ostatniego uruchomienia, jest szczególnie ważna, jeśli badamy

np. przerwy w zasilaniu, zaburzenia stabilności i wynikające z nich restarty systemu, lub pozostające poza wiedzą i zgodą administratora, celowe działania osób trzecich.

```
1) macierz RAID
2) czas funkcjonowania systemu
3) koniec
[wybierz czujnik którego wartość chcesz otrzymać:1

Ilość macierzy RAID w systemie to: 1

Macierz RAID działa poprawnie i jest w stanie ONLINE
```

**Rys. 22. Wynik testu manualnego macierzy RAID w serwerze wirtualizacji A**

Powyższy rysunek prezentuje wynik, który na potrzeby niniejszej pracy demonstruje syntezę analizy ilościowej (ilość dostępnych macierzy RAID) oraz sprawnościowej (weryfikuje parametr sprawności i dostępności poszczególnych macierzy w systemie).

Należy pamiętać, o wykorzystaniu dostarczonej przez producenta dokumentacji protokołu SNMP, aby w procesie tworzenia testów, jak analizy ich wyników zapewnić jak najwyższy poziom rzetelności i wiarygodności.

Podsumowując całość wyżej wymienionych argumentów, warto zauważyć także iż:

- wyżej zaprezentowany skrypt i narzędzie, jest swoistą listą kontrolną kolejnych czynności do wykonania, i nie wymaga dodatkowego tworzenia takowej, jak w przypadku, kiedy każdy test realizowany byłby manualnie przez klasyczne interfejsy urządzeń,
- narzędzie czyni czynność taką jak bieżąca analizy stanu infrastruktury, prostym do wykonania procesem diagnostycznym, możliwym do scedowania na personel niższego stopnia (wymaga wyłącznie krótkiego okresu wdrożeniowego, aby możliwa była analiza i weryfikacja otrzymywanych danych),
- znaczącym aspektem rozwiązania, którego nie można pominąć jest jego wpływ na wzrost bezpieczeństwa w organizacji, przez możliwość realizacji testów bez udostępniania wskazanym do tego osobom danych poświadczeń każdego pojedynczego urządzenia, jakie podlega analizie.

#### **4.4. Testy automatyczne**

Charakterystycznym elementem testów automatycznych, w kontekście analizy infrastruktury IT jest fakt, iż nie są one jedynie maszynowym powieleniem zestawu działań jak w przypadku testów

oprogramowania, a realizacją oraz analizą wyników testów wraz z ich raportowaniem, na wypadek wystąpienia awarii bądź anomalii. Testy takie informują administratorów odpowiedzialnych za infrastrukturę wyłącznie w sytuacji wystąpienia awarii (za wyjątkiem obowiązkowych raportów zbiorczych, będących jednocześnie informacją o poprawności funkcjonowania systemu). Dodatkowo, przez wykorzystanie protokołu SNMP nie obciążają one znacząco urządzeń wewnątrz danej infrastruktury, co sprawia, że istnieje możliwość ich realizacji wielokrotnie w niewielkich odstępach czasowych (np. raz na 1 minutę). Ma to na tyle duże znaczenie i przełożenie na bezpieczeństwo jak i sprawność całości infrastruktury, ponieważ interwał czasowy w takim przypadku to najdłuższy okres istnienia awarii przed jej wykryciem.

```
#!/bin/bash
cd /home/tatiana/skrypty/snmpauto

#PL: weryfikacja istnienia pliku alert.txt i jeśli nie istnieje utworzenie
#EN: check for existence of alert.txt and if it does not exist create
if [ -f "alert.txt" ];
then
    echo " "
else
    touch alert.txt
fi

# PL: weryfikowanie czy temperatura dysków jest mniejsza od 55 stopni
# EN: Check if the temperature of the disks is lower than 55 degrees
declare -i HDD1T=0
declare -i HDD2T=0
HDD1T=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.6.0)
HDD2T=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.6.1)
if [ $HDD1T -ge "55" ];
then
    echo "Temperatura dysku twardego 1 serwera NAS jest wyższa niż 55°C" >> alert.txt
fi
if [ "$HDD2T" -ge "55" ];
then
    echo "Temperatura dysku twardego 2 serwera NAS jest wyższa niż 55°C" >> alert.txt
fi

# PL: weryfikowanie systemu chłodzenia procesora działa poprawnie
# EN: check that cpu cooling system work properly
cpufan=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.1.4.2.0)
if [ $cpufan != "1" ]; then
    echo "chłodzenie procesora serwera NAS działa niepoprawnie" >> alert.txt
fi

# PL: weryfikowanie systemu chłodzenia urządzenia działa poprawnie
# EN: check that device cooling system work properly
sysfan=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.1.4.1.0)
if [ "$sysfan" != "1" ]; then
    echo "chłodzenie systemu serwera NAS działa niepoprawnie" >> alert.txt
fi

# PL: weryfikowanie stanu dysku twardego 1 serwera NAS
# EN: check that disk 1 work properly
hdd1s=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.5.0)
if [ $hdd1s != "1" ]; then
    echo "stan dysku nr 1 serwera NAS wymaga pilnej uwagi" >> alert.txt
fi

# PL: weryfikowanie stanu dysku twardego 2 serwera NAS
# EN: check that disk 2 work properly
hdd2s=$(snmpget -v 2c -c public -O qv 10.20.31.2 1.3.6.1.4.1.6574.2.1.1.5.1)
if [ $hdd2s != "1" ]; then
    echo "stan dysku nr 2 serwera NAS wymaga pilnej uwagi" >> alert.txt
fi
```



```

fi

# PL: weryfikacja aktualności systemu operacyjnego
# EN: check that system is up to date
sysupp=$(snmpget -v 2c -c public -O qv 10.20.31.2 .1.3.6.1.4.1.6574.1.5.4.0)
if [ $sysupp != "2" ]; then
    echo "system serwera nas jest nieaktualny i wymaga aktualizacji" >> alert.txt
fi

# PL: weryfikacja stanu macierzy RAID w serwerze wirtualizacji
# EN: check that RAID in proxmox server is working properly
RAIDS=$(snmpget -v 2c -c public -O qv 10.20.30.4 .1.3.6.1.4.1.674.10892.5.5.1.20.140.1.1.4.1)
if [ $RAIDS -ne "2" ]; then
    echo "Macierz RAID nie działa poprawnie i należy ją naprawić" >> alert.txt
fi

# PL: weryfikujemy czy raport alertów zawiera jakieś wpisy
# EN: check that alert file contains any data
if [ -s alert.txt ]; then

    #PL: zweryfikuj czy jest zainstalowana aplikacja do obsługi poczty
    #EN: check if the application too send a email is already installed in system
    if [ -x "$(command -v swaks)" ]; then

        # PL: wysłanie raportu do osoby monitorującej IT
        # EN: send alert to IT person
        swaks --body ./alert.txt --to tetiana.w[REDACTED]@gmail.com --from
tetiana.h[REDACTED]@wp.pl --server smtp.wp.pl --port 587 --auth LOGIN --auth-user tetiana.h[REDACTED]@wp.pl
--auth-password [REDACTED]
    else
        echo "w systemie nie zainstalowano aplikacji do obsługi poczty swaks, błędne elementy
infrastruktury zostaną wyświetlone na ekranie"
        echo " "
        echo "$(cat alert.txt)"
    fi

    #PL: czyścimy plik z alertami
    #EN: clean the file alert.txt
    > alert.txt

    #PL: czyścimy i ustawiamy wartość poprawnych testów na 0 w pliku liczbatestow.txt
    #EN: set number of passed tests to 0 and write to file liczbatestow.txt
    declare -i liczbatestow=0
    > liczbatestow.txt
    echo $liczbatestow >> liczbatestow.txt

else
    # PL: powiększenie zmiennej liczby poprawnych testów o 1 i zapisanie jej do pliku
    liczbatestow.txt
    # EN: increase the number of passed tests by 1 and write it to the file liczby.txt
    declare -i liczbatestow
    liczbatestow=$( cat liczbatestow.txt )
    liczbatestow=${liczbatestow}+1
    > liczbatestow.txt
    echo $liczbatestow >> liczbatestow.txt
    if [ $liczbatestow -ge "10" ]; then

        # PL: wysyłamy informacje o 10 ostatnich poprawnych testach infrastruktury
        # EN: send 10 last passed tests like a info to IT person
        swaks --body "Gratulujemy twoja infrastruktura w całości działa poprawnie a system weryfikacji
stale i poprawnie ją weryfikuje" --to tetiana.w[REDACTED]@gmail.com --from tetiana.h[REDACTED]@wp.pl --
server smtp

        #PL: czyścimy i ustawiamy wartość poprawnych testów na 0 w pliku liczbatestow.txt
        #EN: set number of passed tests to 0 and write to file liczbatestow.txt
        declare -i liczbatestow=0
        > liczbatestow.txt
        echo $liczbatestow >> liczbatestow.txt
    fi
fi

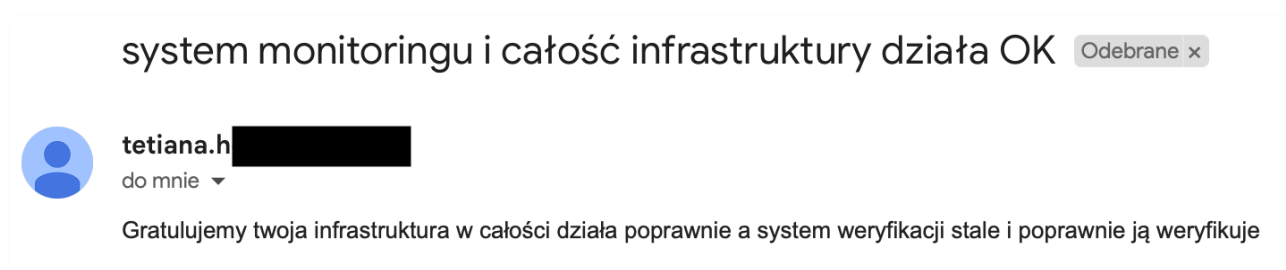
```

Listing 6. Kod źródłowy skryptu testu automatycznego weryfikacji sprawności



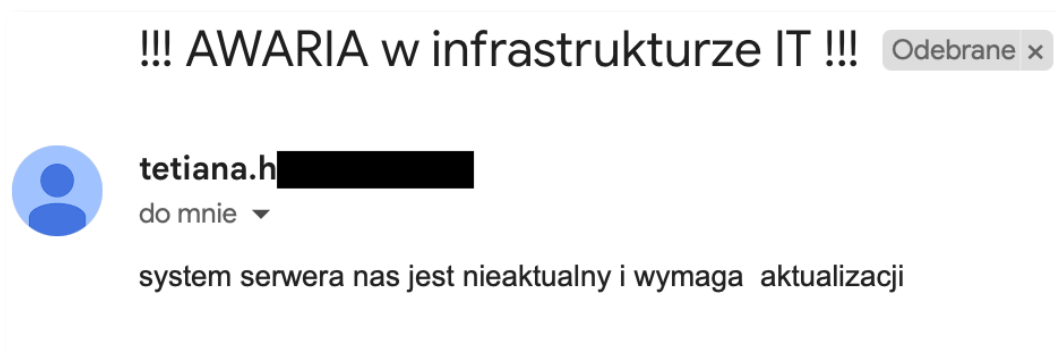
Prezentowany powyżej skrypt, bazuje na mechanizmie wykorzystywanym w przypadku manualnych testów sprawnościowych. Jednostkowe wyniki otrzymywane z czujników OID nie są prezentowane jako informacja zwrotna na ekranie, lecz są poddawane rozróżnieniu, na takie które wskazują na sprawność danego urządzenia, oraz takie które sygnalizują wystąpienie sytuacji wymagających interwencji administracyjnych lub diagnostyczno-naprawczych.

W przypadku testów sprawnościowych, w przeciwieństwie do testów osiągalności, nie przewidujemy możliwości wystąpienia testów fałszywie negatywnych, a wyniki są rozróżniane jednoznacznie jako wskazujące na występowanie lub brak awarii w urządzeniu. Celem tego rodzaju testów jest wyłącznie monitoring sprawności infrastruktury IT, przez co nie wykonujemy tutaj testów o charakterze informacyjnym, których wyniki nie posiadają wpływu na analizę bieżącego stanu sprawności infrastruktury.



**Rys. 23. Pozytywny wynik testu automatycznego testów sprawnościowych**

Elementem niezbędnym w każdym z tego rodzaju automatycznych testów sprawności infrastruktury, informacja poprawnym funkcjonowaniu tego systemu (wiadomość kontrolna). W przypadku przedstawionego powyżej skryptu, taka informacja zostanie przesłana do administratora co 10 kolejno wykonywanych testów.



**Rys. 24. Rezultat negatywnego testu automatycznego dla testu sprawności infrastruktury**

W przeciwieństwie do wiadomości kontrolnych, wysyłanych cyklicznie, wiadomość na temat wystąpienia stanu ostrzegawczego, przekroczenia dopuszczalnych limitów bądź awarii któregoś

z podzespołów bądź procesów wewnątrz urządzenia wysyłana jest natychmiast po zakończeniu całości skryptu. Powyższy rysunek ilustruje przykład wiadomości email, która musi w sposób jednoznaczny określać kluczowe informacje dotyczące identyfikacji uszkodzonego sprzętu (w naszym przypadku jest to serwer nas) oraz w sposób czytelny i klarowny przekazać przyczyny, przez które została ona wysłana do administratora.

## 5. Podsumowanie i omówienie rezultatów pracy

W ramach realizacji projektu udało się przeprowadzić kompleksową ocenę wybranych narzędzi i mechanizmów związanych z bezpieczeństwem systemów informatycznych. Przyjęty w projekcie model diagnostyki infrastruktury IT, pozwolił na efektywne wykrywanie i usuwanie nieprawidłowości oraz zapobieganie awariom.

W rzeczywistości jednak, każda implementacja rozwiązań dotyczących bezpieczeństwa IT, choćby wykorzystująca najprostsze narzędzia i mechanizmy, to zazwyczaj spory nakład czasu pracy, działań administracyjnych i organizacyjnych oraz obciążenie kosztowe. Rozpoczęcie i wykonanie tego procesu możliwe jest wyłącznie w sytuacji, kiedy zasadność całego zestawu działań posiada potencjalny wpływ na zdrowie i życie pracowników organizacji, zabezpieczenie jej otoczenia (kontrahentów, odbiorców, petentów) lub redukcję ryzyka powstania strat finansowych i materialnych. Przedstawione powyżej przykłady zastosowań opracowywanych rozwiązań (*case study*), stanowią zatem integralny i niezbędny warunek każdego z nich.

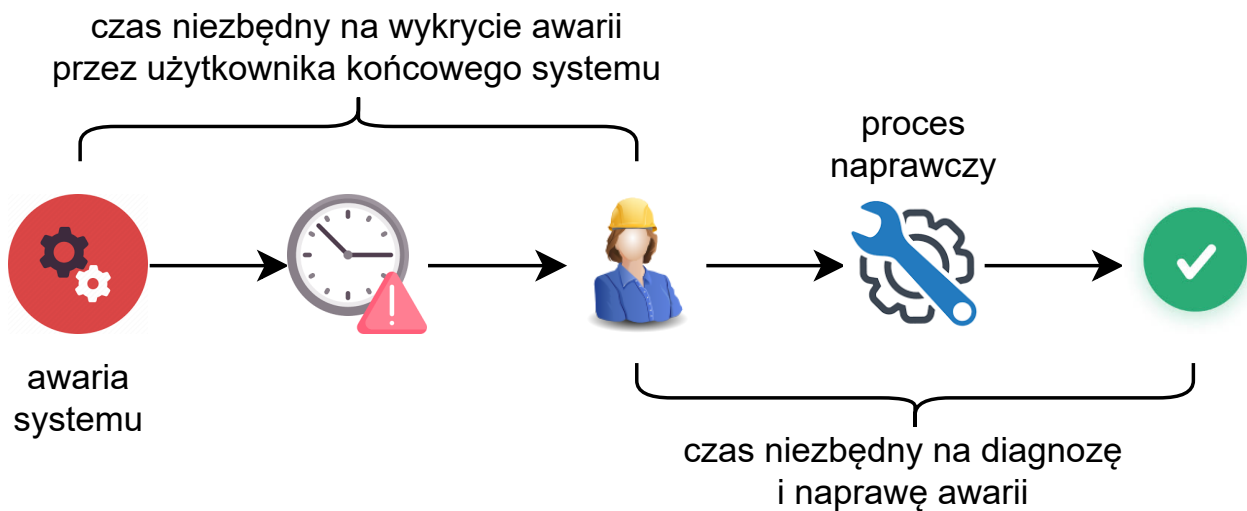
Decyzyjność w zakresie wdrożenia opracowanych tutaj narzędzi, powinna być oparta na typowej dla tego typu rozwiązań analizie oceny ryzyka, to znaczy:

$$\text{koszt wdrożenia} < \text{prawdopodobności jego wystąpienia} \times \text{koszt jego następstwa}$$
  
z uwzględnieniem oczywistych w takich przypadkach czynników dodatkowych jak ścieżki krytyczne, elementy etyczne bądź wizerunkowe przedsiębiorstwa bądź instytucji.

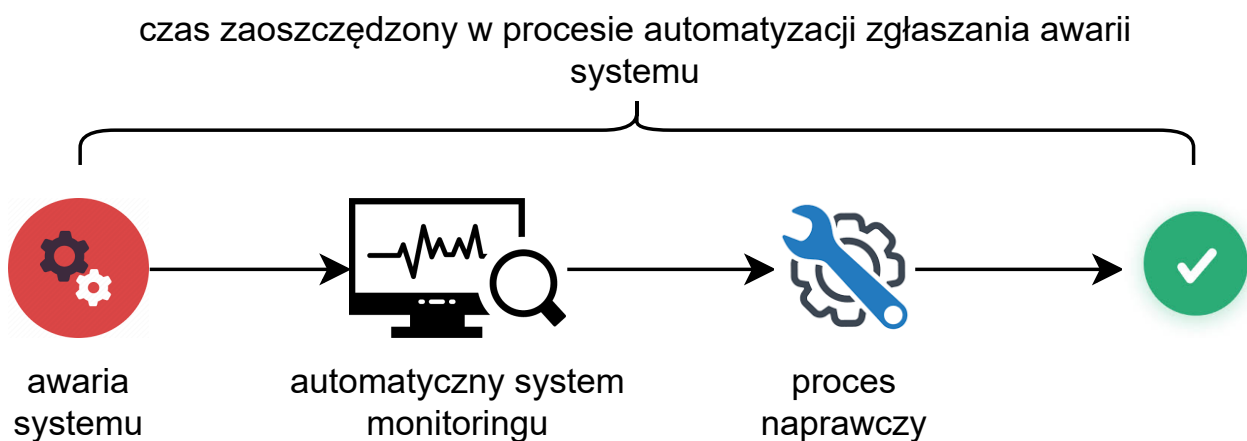
Wyniki każdego z poszczególnych zaprezentowanych powyżej testów jednoznacznie wskazują na fakt, iż stanowią one redukcję czasu reakcji niezbędnego do rozpoczęcia procesu diagnostyczno-naprawczego do minimum, i mogą być elementem decydującym w zakresie długości trwania awarii bądź obszaru jej oddziaływania na urządzenia lub dane klienta, odbiorcy lub użytkownika systemu. Dodatkowo poprawne wykorzystanie w/w narzędzi pozwoli na wydłużenie żywotności poszczególnych elementów infrastruktury, tym samym redukcję potencjalnych kosztów utrzymania infrastruktury IT.

Standardowy sposób naprawy infrastruktury IT ma charakter **reaktywny** i polega na działaniach będących wynikiem zgłoszenia uszkodzenia lub niestabilnego działania zarówno całej jak i części infrastruktury IT i usług w ramach niej funkcjonujących przez odbiorcę bądź użytkownika końcowego. Negatywne aspekty tego rodzaju rozwiązań to oczywiście opóźnienie procesu diagnozy i naprawy o czas w jakim działa ona nieprawidłowo a momentem zauważenia przez odbiorcę, bądź klienta tej nieprawidłowości. Dodatkowo w czasie tym występuje ryzyko dodatkowych komplikacji wynikających z uszkodzeń powiązanych z pierwszym incydentem.

W prezentowanym w tej pracy modelu, w którym infrastruktura jest stale monitorowana i testowana przez systemy autonomiczne, które samodzielnie potrafią zgłaszać informacje na temat błędów i nieprawidłowości w niej występujących, mamy do czynienia z działaniami o charakterze **aktywnym**, gdzie proces diagnostyki i naprawy rozpoczyna się natychmiast po powstaniu błędu bądź awarii infrastruktury.



Rys. 25. Standardowy proces zgłaszania awarii



Rys. 26. Proces zgłaszania awarii po wdrożeniu rozwiązań zaproponowanych w pracy

## Spis podstawowych pojęć i definicji

**ICMP** (*Internet Control Message Protocol*) - protokół warstwy sieciowej, używany do przesyłania informacji kontrolnych i diagnostycznych między urządzeniami, takimi jak *routery* czy komputery. Często używany z narzędziem *ping* do diagnozowania problemów z połączeniem<sup>1</sup>.

**PING** (*Packet Internet Groper*) - narzędzie służące do rozwiązywania problemów warstwy internetowej. Wykorzystuje ono komunikaty ICMP Echo i odpowiedź echa do weryfikacji połączenia między dwoma hostami IP. Po wysłaniu pakietu, host docelowy ICMP odpowiada na żądanie echa pakietem odpowiedzi echa<sup>2</sup>.

**FTP** (*File Transfer Protocol*) – protokół transferu plików, umożliwiający przesyłanie plików między zdalnym i lokalnym komputerem<sup>3</sup>.

**DHCP** (*Dynamic Host Configuration Protocol*) – protokół używany do dynamicznego przydzielania adresów IP klientom z centralnego serwera<sup>4</sup>.

**Telnet** (*Network Terminal Protocol*) – protokół umożliwiający nawiązanie połączenia z urządzeniem zdalnym i rozpoczęcie sesji terminalowej. Za pomocą tego protokołu można zarządzać urządzeniami lub usługami sieciowymi<sup>5</sup>.

**SNMP** (*Simple Network Management Protocol*) – protokół służący do zarządzania siecią. Jego głównym zadaniem jest zbieranie, analiza i raportowanie danych związanych z działaniem różnych komponentów lub usług w sieci w celu ułatwienia zarządzania nią. Protokół począwszy od wersji 2c lub 3 umożliwia także zarządzanie podstawowymi funkcjonalnościami tych urządzeń<sup>6</sup>.

**baza MIB** (*Management Information Base*) – zbiór zmiennych opisujących obiekty, umożliwiając ich zarządzanie w systemie. Każdy obiekt posiada unikalny identyfikator, który pozwala na odczytanie jego wartości lub w niektórych przypadkach, na jego zmianę. Standardowe

---

<sup>1</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 42.

<sup>2</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 119.

<sup>3</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 60.

<sup>4</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 44.

<sup>5</sup> cf. Hunt C., TCP/IP Administracja sieci [TCP/IP Network Administration], Warszawa: Oficyna Wydawnicza READ ME, 1996, Cichocki M., Hermann J., Patryka M. (tłum.), s. 50-52.

<sup>6</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 60.

identyfikatory są wymagane, aby zapobiec problemom wynikającym z różnych nazw obiektów używanych przez różnych producentów<sup>7</sup>.

**OID** (*Object Identifier*) – unikalny identyfikator obiektu w strukturze drzewa OID. Jest to liczba naturalna reprezentowana w formacie tekstowym i składa się z serii cyfr oddzielonych kropkami. Identyfikator pojedynczego „czujnika” zawierający informacje na temat stanu tego czujnika<sup>8</sup>.

**NAS** (*Network Attached Storage*) – urządzenie służące do przechowywania danych, które jest podłączone do sieci komputerowej i umożliwia dostęp do danych z różnych urządzeń poprzez protokoły sieciowe, takie jak SMB, NFS czy FTP.

**SLA** (*Service Level Agreement*) – umowa zawierana między dostawcą usług a klientem, która definiuje wymagany poziom dostępności usługi w określonym zakresie czasowym.

**Cron** – usługa występująca w systemach operacyjnych rodziny Unix, umożliwiająca uruchamianie określonych zadań, skryptów lub programów w ustalonych interwałach czasowych, poprzez definiowanie harmonogramu zadań.

**Testy osiągalności** (*reachability tests*) – pozwalają na redukcję do niezbędnego minimum, czasu pomiędzy wystąpieniem awarii a momentem rozpoczęcia jej diagnozy i naprawy.

**Testy sprawnościowe** (*performance tests*) – pozwalają na wykrycie nieprawidłowości w zakresie funkcjonowania sprzętu i usług oraz umożliwiają jak najszybszy proces diagnostyczny i naprawczy w celu przywrócenia usługi do pełnej sprawności bądź zapobieżenie eskalacji awarii<sup>9</sup>.

---

<sup>7</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 500-501.

<sup>8</sup> cf. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., TCP/IP. Biblia [TCP/IP Bible], Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.), s. 500.

<sup>9</sup> cf. International Software Testing Qualifications Board. (s.d.). ISTQB Glossary [Słownik ISTQB]. [https://glossary.istqb.org/pl\\_PL/search](https://glossary.istqb.org/pl_PL/search), dostęp: 17.04.2023.

## Spis materiałów źródłowych

1. Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M., *TCP/IP. Biblia [TCP/IP Bible]*, Gliwice: Wydawnictwo Helion, 2002, Jarczyk A. (tłum.)
2. International Software Testing Qualifications Board. (s.d.). *ISTQB Glossary [Słownik ISTQB]*.  
[https://glossary.istqb.org/pl\\_PL/search](https://glossary.istqb.org/pl_PL/search)
3. Hunt C., *TCP/IP Administracja sieci [TCP/IP Network Administration]*, Warszawa: Oficyna Wydawnicza READ ME, 1996, Cichocki M., Hermann J., Patryka M. (tłum.)

## Spis rysunków

Rys. 1. Schemat fizyczny infrastruktury technicznej.....	6
Rys. 2. Schemat logiczny infrastruktury technicznej.....	7
Rys. 3. Centrum informacji o serwerze NAS .....	8
Rys. 4. Infrastruktura A w Proxmox .....	9
Rys. 5. Informacje o kamerze Hikvision .....	9
Rys. 6. Informacje o fizycznym komputerze .....	10
Rys. 7. Infrastruktura B w Proxmox .....	10
Rys. 8. Dashboard router .....	11
Rys. 9. Schemat blokowy skryptu wykonania testów manualnych w testach osiągalności.....	13
Rys. 10. Prezentacja wyniku testu manualnego w testach osiągalności.....	15
Rys. 11. Schemat blokowy skryptu wykonania testów automatycznych w testach osiągalności.....	16
Rys. 12. Prezentacja wyniku testu automatycznego w testach osiągalności.....	19
Rys. 13. Konfiguracja Crona na wykonywanie testów w odstępie czasowym 1min. w testach osiągalności .....	19
Rys. 14. Prezentacja wyniku testu automatycznego ustawionego na wykonywanie co określony czas .....	20
Rys. 15. Obraz konsoli skryptu snmp.sh po uruchomieniu testów manualnych dla sprawności.....	23
Rys. 16. Obraz konsoli skryptu serwera NAS po jego uruchomieniu .....	25
Rys. 17. Prezentacja wyniku temperatury HDD serwera NAS.....	26
Rys. 18. Prezentacja wyniku stanu aktualności serwera NAS .....	26
Rys. 19. Aktualizacja systemu DSM serwera NAS .....	27
Rys. 20. Rezultat aktualizacji systemu operacyjnego serwera NAS .....	28
Rys. 21. Obraz konsoli skryptu serwera A po jego uruchomieniu.....	29
Rys. 22. Wynik testu manualnego macierzy RAID w serwerze wirtualizacji A .....	30
Rys. 23. Pozytywny wynik testu automatycznego testów sprawnościowych.....	33
Rys. 24. Rezultat negatywnego testu automatycznego dla testu sprawności infrastruktury.....	33
Rys. 25. Standardowy proces zgłaszania awarii.....	36
Rys. 26. Proces zgłaszania awarii po wdrożeniu rozwiązań zaproponowanych w pracy.....	36



## Spis listingów

Listing 1. Kod źródłowy skryptu wykonania testów manualnych w testach osiągalności .....	14
Listing 2. Kod źródłowy skryptu wykonania testów automatycznych w testach osiągalności.....	18
Listing 3. Kod źródłowy skryptu wykonania testu manualnego w testach sprawnościowych.....	22
Listing 4. Kod źródłowy skryptu do wykonywania testów manualnych serwera NAS w testach sprawnościowych .....	25
Listing 5. Kod źródłowy skryptu wykonania testu manualnego serwera wirtualizacji A w testach sprawnościowych....	29
Listing 6. Kod źródłowy skryptu testu automatycznego weryfikacji sprawności .....	32

## **Załączniki**

[https://github.com/tholycz/projekt\\_kierunkowy](https://github.com/tholycz/projekt_kierunkowy)