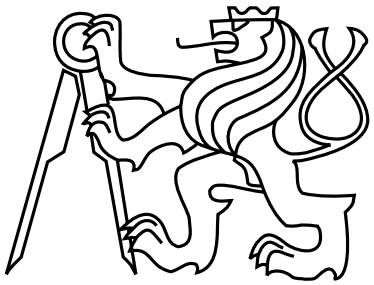
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky



Diplomová práce

Správa a monitoring rozsáhlých sítí

Bc. Lukáš Červenka

Vedoucí práce: Ing. Ján Kučerák

Studijní program: Elektronika a komunikace, Magisterský

Obor: Komunikační sítě a internet

květen 2020

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22. 5. 2020 .....

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Červenka** Jméno: **Lukáš** Osobní číslo: **420412**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Správa a monitoring rozsáhlých sítí

Název diplomové práce anglicky:

Large Networks Monitoring

Pokyny pro vypracování:

Navrhněte a realizujte nástroj, který nad existující virtuální infrastrukturou umožní integraci a centrální dohled nad konfigurací použitých dílčích komponent. Systém musí umožňovat konfiguraci, automatickou kontrolu na základě uživatelem definovaných pravidel, zálohu a obnovu nastavení. Své řešení zaměřte primárně na platformy VMware ESXi, Zabbix a Sophos Firewall. Systém navrhněte v rámci možností modulárně, aby případné přidání podpory nové platformy znamenalo co nejmenší zásah do existujícího systému.

Seznam doporučené literatury:

[1] Mistrovství ve VMware vSphere 5 : kompletní průvodce profesionální virtualizací, Scott Lowe ; překlad Jiří Huf.
[2] Mastering Zabbix Andrea Dalle Vacche and Stefano Lee
[3] Zabbix 1.8 Network Monitoring Rihards Olups
[4] Cloud Computing Security: Foundations and Challenges, John R. Vacca

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ján Kučerák, katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.01.2020** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2021**

Ing. Ján Kučerák podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Abstract

This master thesis focuses on research of network management and monitoring. Its main goal is to analyze environment and develop a software tool, which can centrally monitor and repair configuration of components in existing virtual infrastructure. In this software, user can specify his demands in defined scenarios and then see the consistency of network configuration in one place. The supported components are hypervisors, firewalls, Network Monitoring Softwares (NMS) and documentation systems.

Keywords: NMS, centralized monitoring, REST API, autoconfig tool

Abstrakt

Diplomová práce se zabývá analýzou možností vzdálené správy a monitoringu sítí, a poté návrhem, implementací a testováním nástroje, který umožňuje centrální dohled nad konfigurací komponent existující virtuální infrastruktury. Pomocí něj lze na základě uživatelem definovaných pravidel centrálně provádět kontrolu a opravu nastavení jednotlivých komponent jako jsou hypervizory, firewally nebo dohledové a dokumentační systémy.

Klíčová slova: NMS, centrální dohled, REST API, automatizační nástroj

Poděkování

Rád bych poděkoval vedoucímu své diplomové práce, Ing. Jánu Kučerákovi, za jeho ochotu a trpělivost. Chtěl bych také poděkovat své rodině a přátelům za podporu při studiu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Každá infrastruktura, na které jsou poskytovány služby, vyžaduje pro spolehlivý chod dohled, údržbu a včasnou modernizaci. Aby tyto činnosti mohly být vykonávány efektivně a bezpečně, musí být vzájemně sladěna celá řada systémů - operační systémy na serverech, hypervizory, firewally a jiné síťové prvky, dohledové systémy a také aktuální dokumentace.

Ačkoli se o velkou část údržby a dohledu dnes starají různé automatizované nástroje, nastavení těchto nástrojů provádí často stále „jen“ člověk. V reálném nasazení dochází vlivem lidské nedůslednosti k nekonzistenci nastavení nebo neaktuálnosti dokumentace, což může způsobovat zbytečné výpadky či prodlužovat dobu obnovy v případě výskytu mimořádné události.

Vzhledem k tomu, že většinu zmíněných systémů lze konfigurovat pomocí API či jiné metody vzdáleného přístupu, vznikla myšlenka vytvořit propojení mezi těmito systémy - vrstvu, která by aktuální stav reálné infrastruktury promítala především do konfigurace dohledu a dokumentace.

V úvodních kapitolách jsou popsány způsoby vzdálené správy a dohledu. Další kapitoly se zabývají klíčovou částí zadání práce, tedy návrhem a implementací automatizačního softwaru, a také jeho testováním v reálném prostředí.

Kapitola 2

Správa a dohled sítí

Předpokladem úspěšného provozování síťové infrastruktury je možnost jednotlivé prvky vzdáleně řídit, sledovat jejich stav a mít přehled o způsobu nastavení infrastruktury. Ke každé této sekci existuje celá řada programových nástrojů a systémů, které správci usnadňují její realizaci.

2.1 Struktura rozsáhlých sítí

Komunikace v počítačových sítích se dnes zpravidla drží packetové komunikace a referenčního modelu ISO/OSI[10], který pracuje s představou sedmi vrstev komunikace. Výhodou takového přistupu je možnost elegantního návrhu sítí, kdy na různých úrovních je nutné zpracovávat jen menší množsví informací z packetu - vyšší vrstvy jsou pro přenos transparentní.

Počítačové sítě se dále tradičně dělí dle svého rozsahu a způsobu použití na sítě.

- **Personal Area Network (PAN)**
  Síť s krátkým dosahem sloužící k propojení zařízení jednoho uživatele. Data se mohou přenášet po kabelu nebo bezdrátovou technologií pracující zpravidla ve volném pásmu. Typickými zástupci jsou technologie Bluetooth, USB, ZigBee a podobně.

- **Local Area Network (LAN)**
  Jedná se o síť propojující počítače a podobná zařízení uvnitř omezené oblasti, například školy nebo kanceláře. Běžná technologie pro přístup do sítě pomocí kabelu je dnes Ethernet, definován ve standardech 802.3 od Institute of Electrical and Electronics Engineers (IEEE), s rychlostí mezi 100 Mbps a 10 Gbps. Obvykle se jedná o metalický kabel s kroucenými páry zakončený konektorem RJ-45 s délkou do 100 metrů. V případě přístupu přes bezdrátovou technologii se využívá nejčastěji WiFi, přičemž jednotlivé standardy jsou definovány rovněž IEEE v řadě 802.11. Rychlost a spolehlivost přenosu závisí na řadě parametrů jako verzi WiFi protokolu (aktuálně 802.11n, 802.11ac, 802.11ax), kvalitě komunikačního kanálu mezi přístupovým bodem a uživatelem (přístup se koná ve sdíleném pásmu 2.4 nebo 5 GHz) a také využití média ostatními účastníky (jedná se o halfduplexní komunikaci na sdíleném médiu). Reálně dosahované rychlosti jsou mezi jednotkami a stovkami Mbps[3].

- **Metropolitan Area Network (MAN)**
  Jedná se o mezistupeň mezi LAN a WAN. Metropolitní síť představuje propojení sítí LAN v rámci určité oblasti. Může se jednat například o propojení budov vysokých škol nebo lokální optická či bezdrátová síť. Přenos dat probíhá přes optická vlákna s využitím point-to-point propojů nebo systémů CWDM/DWDM, případně přes profesionální mikrovlnné spoje ve vyšších frekvenčních pásmech [12].

- **Wide Area Network (WAN)**
  Sítě WAN slouží k přenosu dat mezi sítěmi LAN/MAN, obvykle na velké vzdálenosti. Slouží k přístupu k síti Internet, rychlost přenosu mezi dvěma body vždy záleží na kvalitě celé komunikační trasy. Jednotlivé WAN sítě poskytovatelů služeb se propojují v peeringových centrech.

V této práci je nutné kombinovat různé pohledy na spravovanou síť.

Požadavkem na monitoring sítě jedné organizace může být sledování spolehlivosti a kvality provozovaných služeb jako Hypertext Transfer Protocol Secure (HTTPS), Internet Message Access Protocol (IMAP) nebo Domain Name System (DNS) (čtvrtá až sedmá síťová vrstva), využití a stav WAN linek připojení do Internetu (druhá až čtvrtá síťová vrstva) a nebo také kontrola chybovosti na bezdrátových linkách sloužících jako propojení mezi budovami organizace (první a druhá síťová vrstva).

Pokud se budeme dívat na síť poskytovatele internetových nebo cloudových služeb, bude pravděpodobně převažovat potřeba dohledu nad infrastrukturou a dostupností služeb.

Pohled na spravovanou síť musí být tedy komplexní, v rámci zjednodušení si síť rozdělíme na tři následující části, ve kterých můžeme zobecnit principy jejich dohledu a správy.

Síťová infrastruktura

Část sítě, ve které nás zajímá infrastruktura pro přenos dat, tedy první až zhruba čtvrtá vrstva ISO/OSI modelu.

V rámci síťové infrastruktury se zabýváme správou a monitoringem parametrů například následujících bodů.

- **Stav a konfigurace síťových prvků**

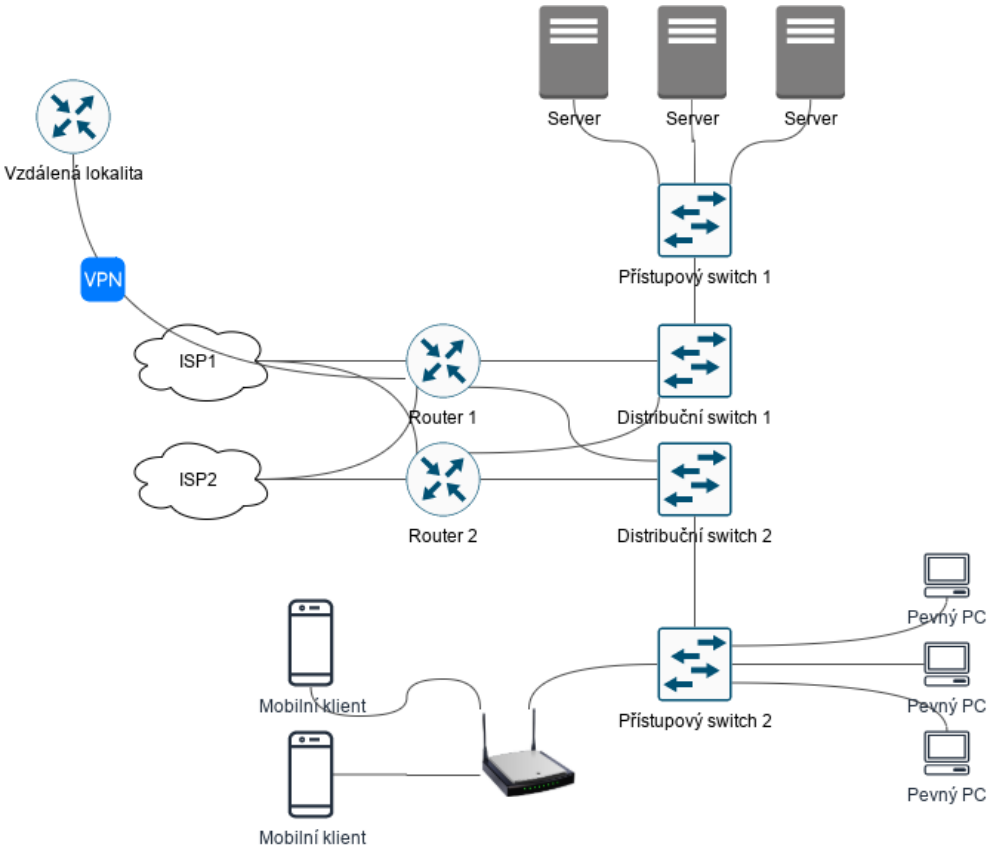
  Pod tento bod spadá fyzický stav síťového prvku, napájení, hardwarová funkčnost, operační systém prvku a jeho konfigurace.

- **Fyzická vrstva**

  Fyzická přítomnost a stav propojovacích linek, útlum optických a metalických tras, linková rychlost, duplex, bitová chybovost, vysílaný a přijímaný výkon na bezdrátovém a optickém propoji, modulace, Signal-to-noise ratio (SNR) a další.

- **Linková vrstva**

  Saturování linek, počty přenesených/zahozených rámců, jejich velikost a chyb. Funkčnost protokolů pro zajištění redundance na L2 jako (R)STP, PVST, MSTP. Můžeme sem zařadit také sledování FDB tabulek a detekci nových zařízení na síti.



Obrázek 2.1: Schéma síťové infrastruktury organizace

- **Síťová vrstva**

  Přidělování IP adres (DHCP, SLAAC, DHCPv6). Směrování packetů na routerech, dynamické protokoly BGP nebo OSPF, stav jejich sousedů a routovacích tabulek. First hop redundance, jako je VRRP.

- **Transportní vrstva**

  Rychlost síťových aplikací často závisí především na TCP protokolu na čtvrté vrstvě. Můžeme provádět kontrolní měření a sledovat vývoj parametrů jako je zpoždění, jitter a přenosová rychlost.

  Z bezpečnostních nebo právních[21] důvodů dochází také k zaznamenávání nebo zkoumání hlaviček packetů uživatelského provozu na síti.

- **Virtuální propojení**

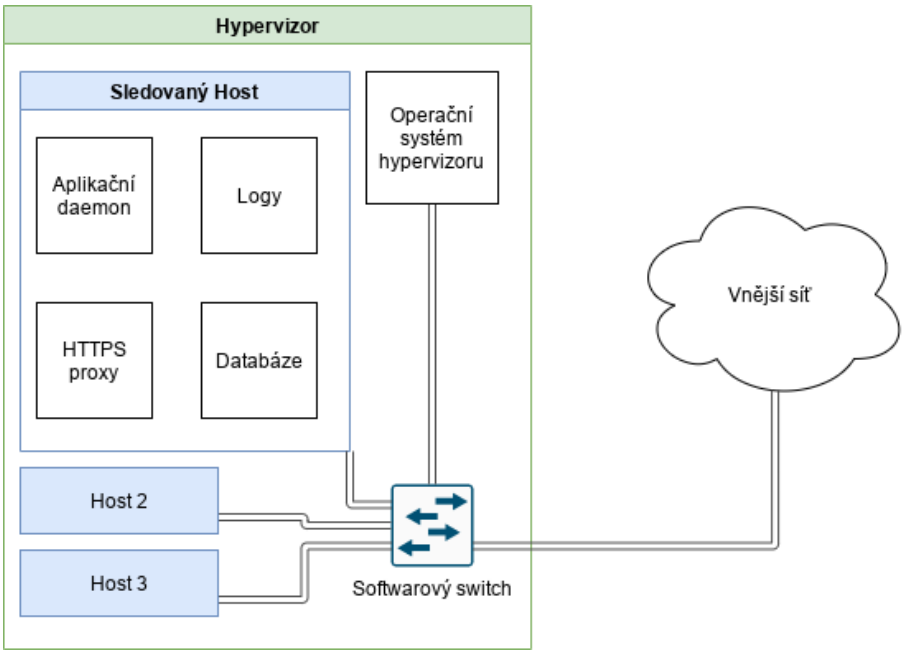
  Do této kategorie patří různé formy tunelů a Virtual Private Network (VPN). Zajímá nás stav navázání spojení, statistiky přenosu, podrobné údaje o připojeném klientovi.

  Pro účely připojení přes pronajaté okruhy či v rámci skutečně rozsáhlých sítí jedné organizace bude žádoucí sledovat parametry Multiprotocol Label Switching (MPLS)

nebo obdobné technologie. Pro monitoring stavu MPLS je možné použít i protokol SNMP[4].

Služby a servery

Při správě a monitoringu služeb se zabýváme především vyššími síťovými vrstvami, stavem samotné aplikační služby, operačního systému a podpůrné infrastruktury. Vzhledem k masivnímu rozšíření virtualizace a kontejnerizace[23] lze tuto sekci demonstrovat následujícím příkladem.



Obrázek 2.2: Schéma služeb na hypervizoru

- **Hypervizor**

  Pojem hypervizor označuje server, který umožňuje spouštět více operačních systémů najednou. Takový jev nazáváme virtualizací, přičemž pro virtuální počítač se používá též označení „host“.

  Pro zjednodušení uvažujme nativní hypervizor, který je provozován přímo na fyzickém hardwaru bez další mezivrstvy. V takovém případě je žádoucí monitorovat stav hardwaru hypervizoru (teplotu, přítomnost napájení na zdrojích, stav diskového pole) i operačního systému hypervizoru (využití paměti, úložiště, procesoru, sítě).

- **Operační systém hosta**

  Jelikož na každém hostovaném virtuálním stroji běží samostatný operační systém, musíme spravovat a monitorovat i jej. Kromě informace o využití zdrojů můžeme sledovat i systémové logy a další údaje.

- **Stav a konfigurace aplikační služby**

  Specifickou částí je správa a sledování konkrétní provozované služby. Z pohledu monitoringu je zásadní udělat přesně cílené testy funkčnosti, které ověří jak síťovou dostupnost např. otevřením TCP spojení, tak správnost aplikací vrácených údajů (HTTPS server vrátil požadovanou stránku s validním certifikátem). V ideálním případě je také možné sledovat detailní statistiky provozu služby, jako jsou třeba počty uživatelů nebo požadavků za daný časový úsek.

Přístupová část sítě

Jedná se o takovou část sítě, která slouží ke zpřístupnění konektivity uživateli. Lze do ní také zahrnout pracovní stanice. Uživatelská zařízení se do sítě připojují pomocí datových kabelů nebo bezdrátové technologie, zásadní je tedy bezpečnost přístupu a ochrana sítě před neoprávněným vniknutím. Na uživatelská zařízení, pokud je má administrátor pod správou, můžeme hromadně distribuovat nastavení, provádět instalace nebo aktualizace softwaru. Ve specifických případech můžeme zaznamenávat i konkrétní činnost uživatele na jeho zařízení.

2.2 Monitoring sítí

Monitoring sítě je proces, při kterém jsou pomocí softwarového nástroje shromažďovány a vyhodnocovány provozní údaje z jednotlivých částí a prvků sítě. Úlohou je detekovat a optimálně předcházet nefunkčnosti síťových zařízení či služeb.

2.2.1 Požadavky na monitoring sítí

Rozsah sledovaných dat

V závislosti na typu a rozsahu monitorované sítě je třeba důkladně zvážit, do jaké hloubky chceme monitoring provádět.

Rozsah možností je velmi široký, od pouhé kontroly dostupnosti až po důkladné statistiky, kdy lze reagovat na překročení limitů nebo nečekaný trend. Pokud je síť rozsáhlá, je potřeba získaná data pro zachování přehlednosti agregovat a vizualizovat ve formě map nebo grafů.

Držíme-li historii sledovaných dat, s rostoucím počtem monitorovaných veličin také rostou nároky na výpočetní výkon a úložiště serveru, kde je dohledový systém spuštěn. Nejde jen o velikost úložistě, ale i o rychlost čtení a zápisu, protože vyčítání dat se provádí v pravidelných intervalech pro každé sledované zařízení.

Existující nástroje pro monitoring sítí bývají zaměřené na úzký rozsah funkcionalit, pro komplexní sledování sítě tak může být potřeba nasadit více systému najednou.

Způsob monitoringu

V zásadě rozlišujeme dva způsoby získávání dat o stavu sítě:

- **Pravidelné vyčítání hodnot (polling)**

  V tomto případě na straně dohledového systému dochází k pravidelnému sběru monitorovaných veličin (např. přes SNMP GET) a následnému porovnání s pravidly pro generování upozornění. Část systému, která sběr provádí, se obvykle nazývá poller.

- **Reagování na události, které oznámí samotné sledované zařízení**

  Jedná se o odlišnou metodu, kdy na straně dohledového systému poslouchá daemon[1], který čeká, až od zařízení dostane zprávu o nějaké vzniklé události. Takovou zprávu opět porovná s uživatelem definovanými pravidly a případně vytvoří upozornění. Doručování takových zpráv se provádí přes SNMP Trap nebo směrováním systémového logu přímo na dohledový server (např. pomocí syslogu).

Výhodné je kombinovat oba přístupy, jelikož pravidelné získávání dat nemusí odhalit krátce se vyskytující problém, který vznikne a zase zmizí v periodě mezi běhy polleru. Stejně tak v případě aktivního upozorňování na problémovou situaci prvkem nemusí být upozornění v důsledku většího problému na síti vůbec doručeno (např. při dočasné nekonzistenci routovací tabulky).

Výstup dohledu

Základní vlastnostní každého monitorovacího systému je schopnost upozornit na vzniklý problém.

Uživatel může nastavit limity hodnot, při jejichž překročení dojde k upozornění. Některé nástroje jsou schopny automaticky odhadnout limitní hodnoty podle šablon.

Upozornění probíhá zobrazením zprávy v dohledovén systému, zaslání emailem, SMS či na nějakou Instant Messaging (IM) platformu.

Výstupem dohledu mohou být též grafy historických hodnot nebo kompletní reporty. Pro data, u kterých s postupem času nepotřebujeme držet velkou přesnost ale spíše trend, se s oblibou využívá Round-robin database (RRD). Například provoz na síťovém rozhraní routeru nepotřebujeme znát zcela podrobně rok zpět, je ale žádoucí umět porovnat historické průměry objemu dat proti dnešnímu provozu, abychom odhadli trendy vývoje do budoucna.

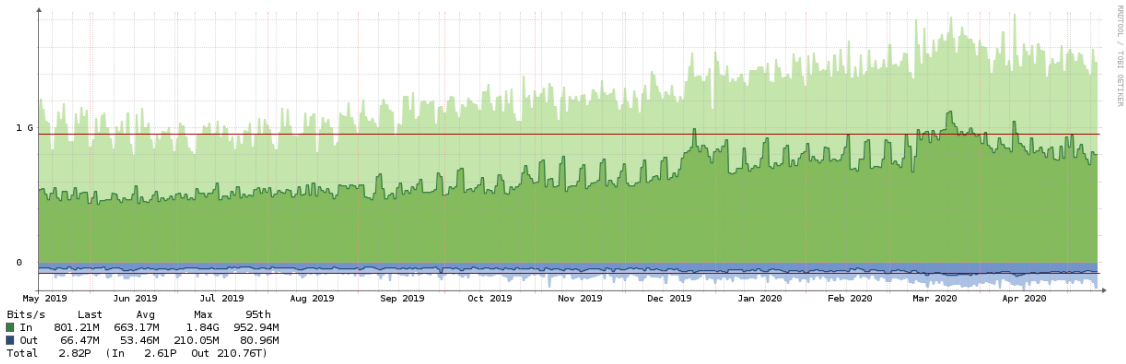
Automatizace

S rostoucí velikostí sítě se zvyšuje důležitost automatizace některých podpůrných procesů potřebných pro její chod. Jeden z takových procesů je i sledování stavu zařízení v síti.

Dohledové systémy často implementují funkci automatického vyhledání nových zařízení na síti. Způsoby provedení jsou různé, jedná se například o skenování IP rozsahu sítě a dotazování pomocí SNMP či některého network discovery protokolu. Jiné nástroje využívají znalost sousedních prvků třeba z tabulky CDP nebo LLDP.

Pokud se využívá více nástrojů pro monitoring jedné sítě, může být výhodné v nich udržovat stejnou databázi sledovaných zařízení nebo data získaná jedním systémem využít v dalším systému.

[1] Daemon je program dlouhodobě spuštěný na pozadí, který často zajišťuje provoz serveru síťové služby.



Obrázek 2.3: Ukázka RRDtool, využití portu routeru během obdoví jednoho roku

2.2.2 Realizace monitoringu

Dohledový systém je provozován zpravidla na serveru poblíž páteře sledované sítě. Důvodem je, že data ze sledování z odlehlejších částí mohou být zatížena chybou některé z tranzitních tras. Také je žádoucí, aby byl dohled dostupný i při výpadku některé z tras. Pro velké sítě je monitoring provozován na více serverech v geograficky oddělených lokalitách.

Protokol SNMP

Simple Network Monitoring Protocol (SNMP) je široce používaný protokol pro monitorování stavu síťových zařízení. Je popsán v řadě dokumentů RFC, aktuální verze především v RFC 3411 až 3418. Pro přenos se používá protokol User Datagram Protocol (UDP) na portech 161 pro získávání dat a 162 pro trapy. Vyskytuje se v několika verzích, v praktickém nasazení lze narazit na verze 1, 2c a 3. Protokol umožňuje čtení i zápis dat.

Struktura dat je stromová, jednotlivá data jsou poskytována jako proměnné různých datových typů. Popis stromu je uveden v souborech Management information base (MIB). Hierarchie dat obsahuje identifikátory OID, pomocí kterých jsou adresovány konkrétní proměnné pro pozdější vyčítaní nebo zápis. Strom lze rozšiřovat o vlastní sekce a hodnoty podle potřeb. K zařízením bývá výrobcem dodáván také MIB soubor popisující implementovaná rozšíření.

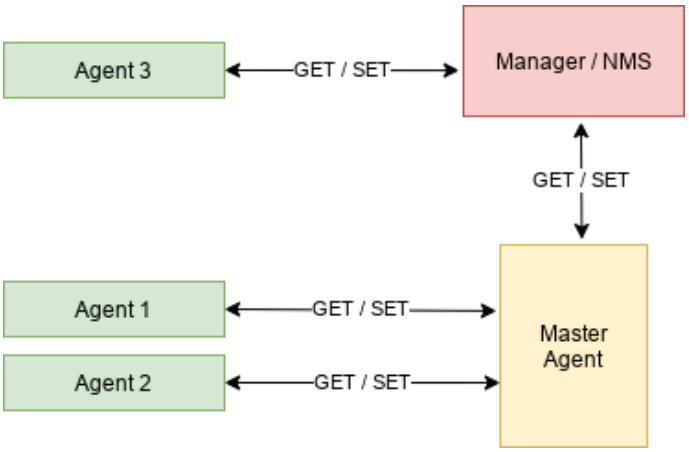
Rozdíly verzí SNMP

První verze SNMPv1 byla publikována již v osmdesátých letech minulého století, má podobný koncept návrhu jakou současná verze a stále se v některých jednodušších zařízeních používá. Z dnešního pohledu zaostává především v oblasti bezpečnosti přenosu, kdy data jsou přenášena nešifrovaně a jedinou ochranou je nutnost při komunikaci specifikovat komunitu (de facto tedy heslo), které se ovšem také přenáší nešifrovaně.

Další komerčně úspěšná je verze SNMPv2c, která přináší vylepšení v oblasti výkonu a také 64 bitové countery, protože stávající 32 bitové mohly například u sledování přenesených dat na 10 Gbps rozhraní přetéct dříve než je vůbec perioda jejich vyčítání NMS. Bezpečnost je bohužel stále ponechána na ověření jména komunity.

Současnou verzí je SNMPv3, která přináší především vyšší bezpečnost přenosu. Nově umožňuje nastavit několik úrovní šifrování, kdy je možné šífrovat jen autentizaci nebo i přenášená data.

Struktura SNMP
Konceptem struktury SNMP je správa síťových prvků pomocí na nich spuštěných SNMP Agentů z místa dohledu, tedy Managera. Agenty lze nadále shlukovat pomocí Master agenta.



Obrázek 2.4: Struktura konceptu SNMP

Typy zpráv SNMP
Základními zprávami v SNMP jsou GET, SET a Trap.

- **GET**
  Pro získávání informací slouží zprávy `GetRequest`, `GetNextRequest` nebo `GetBulkRequest`.
- **SET**
  Pro zápis informací se používá `SetRequest`. Parametrem je jedna nebo více hodnot pro zápis, který je prováděn jako atomická operace.
- **Trap**
  Zpráva, kterou aktivně posílá agent managerovi při detekci nějaké události.

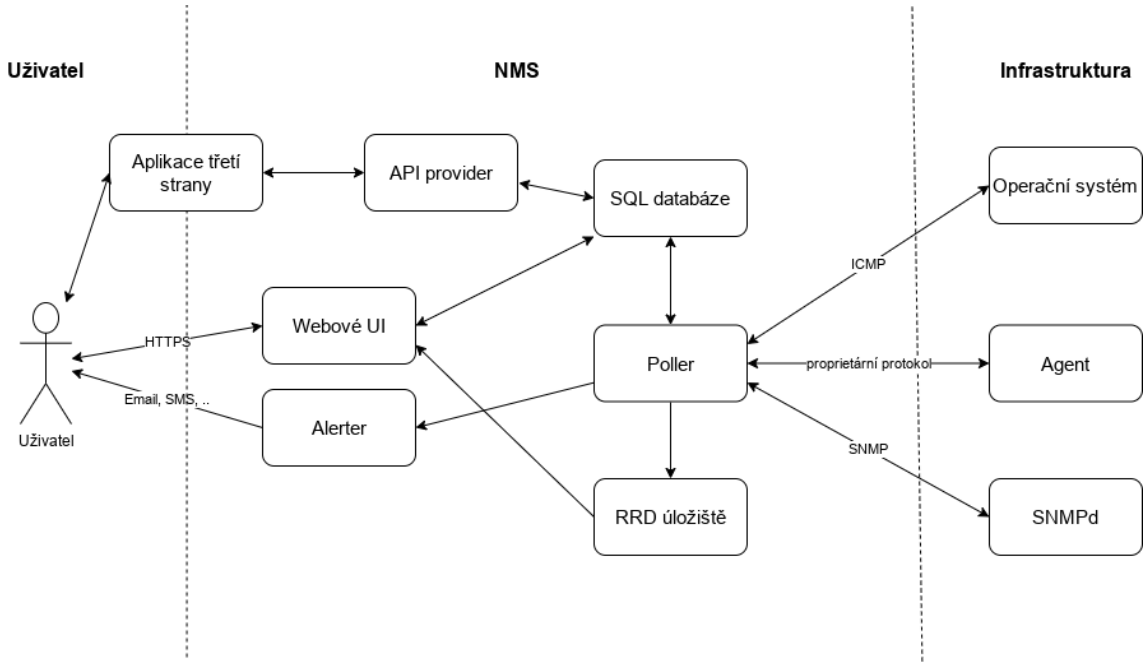
Příklad vyčítání
Vyčítání parametrů radiové části mikrovlnného spoje Ceragon (přijímaný signál, vysílací výkon a SNR) pomocí SNMP z příkazové řádky. Levý údaj značí OID konkrétního parametru, na pravé straně je typ proměnné a její hodnota.

```
$ snmpwalk -v2c -ccommunity 172.23.92.194 1.3.6.1.4.1.2281.10.5.1.1
iso.3.6.1.4.1.2281.10.5.1.1.2.268451969 = INTEGER: -38
iso.3.6.1.4.1.2281.10.5.1.1.3.268451969 = INTEGER: 3
iso.3.6.1.4.1.2281.10.5.1.1.5.268451969 = STRING: "36.21"
```

Struktura dohledového systému

Ačkoli se konkrétní struktura softwaru mezi jednotlivými systémy liší, dají se nalézt společné rysy.

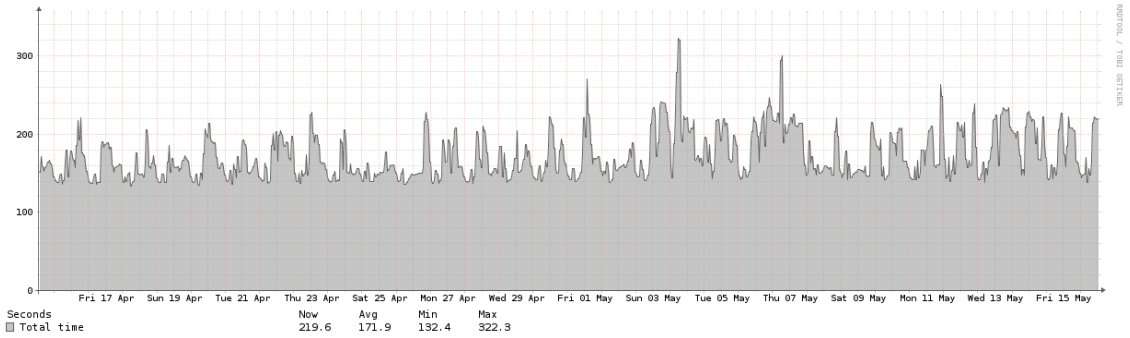
- **Komunikace s uživatelem** je realizována nejčastěji přes webový prohlížeč, tedy protokoly HTTP/HTTPS. V souladu s očekáváními bývá k dispozici tvorba vlastní dashboard s důležitými grafy a údaji nebo responzivní rozhraní pro přístup z mobilního telefonu. Některé NMS vyžadují od uživatele instalaci samostatné aplikace na uživatelský počítač. Objevují se také aplikace pro mobilní telefony.
- **Sběr dat** probíhá na několika úrovních - periodická kontrola dostupnosti přes ICMP echo a vyčítání dat pomocí polleru, sledováním příchozích logů a naslouchánímm zprávám od sledovaných zařízení.
- **Sesbíraná data** o zařízeních se ukládají do databáze. Podle jejich typu a určení jsou to buď SQL databáze jako MariaDB nebo PostgreSQL, RRD soubory, případně key-value databáze jako MongoDB či Elasticsearch.
- **Upozorňování na problémy** je uživateli zasíláno formou emailu, SMS nebo třeba pomocí API do informačního systému. Nalezený problém je znázorněn ve webovém rozhraní, pokročilé systémy umí definovat hierarchii upozornění, aby se zabránilo zahlcení zprávami.



Obrázek 2.5: Architektura obecného NMS

Škálování, optimalizace

S rostoucím počtem sledovaných zařízení roste výpočetní náročnost a přichází tak potřeba optimalizace monitorovacího systému, případně jeho škálování na více serverů. Pokud se dopustí stav, že je dohledový server přetížen, začínají se v grafech objevovat bílá místa nebo se generují falešné poplachy. Častou příčínou takového jevu je, že poller nestíhá doběhnout dříve, než je spuštěna jeho další instance. Řešením by tak mohlo být zvýšení intervalu pollingu, přišli bychom ale o požadovanou přenost měření.



Obrázek 2.6: Statistika času běhu polleru produkčního NMS Observium

V následujících bodech si dovolím shrnout metody optimalizace[15] provozu monitorovacího systému. Většina bodů bude platná pro NMS Observium, které je podrobněji popsáno v další kapitole.

- **Paralelizace polleru**
  Proces vyčítání dat ze zařízení lze obvykle zrychlit tím, že se spustí více instancí polleru najednou, kdy každá zpracovává svou část seznamu zařízení.

  Tento přístup řeší problém s dlouhým pollingem pomalých zařízení, může ale narážet na malou rychlost úložiště nebo databáze.

  V dokumentaci bývá uveden doporučený počet threadů polleru na jedno výpočetní jádro, dohledový systém může také poskytovat statistiky běhu polleru, dle kterých lze vhodné parametry odhadnout.

- **Optimalizace úložiště RRD souborů**

  Ve standardní konfiguraci se získáná data ukládájí na pevný disk do souborů RRD. Každá měřená veličina má svůj vlastní soubor, při pollingu tedy dochází k mnoha čtením a zápisům malých dat z a do velkého počtu souborů.

  Takový způsob zatížení není ideální pro běžné rotační disky, prvním krokem tak může být přesun úložiště na SSD disk. Jako ještě rychlejší úložiště lze použít RAMDisk, tedy simulaci disku čistě v operační paměti. Nevýhodou takového řešení je, že data z RAMDisku jsou při restartu serveru ztracena, musí se tedy provádět jejich pravidelná duplikace na pevný disk, aby se množství ztracených dat minimalizovalo.

Další cestou může být použití `rrdcached`. Jedná se součást `rrdtool`, je to daemon, který shromažďuje data k zápisu do RRD souborů a se zápisem čeká až do doby, kdy jich má k dispozici větší množství.

- **Optimalizace SQL databáze**

  Pomalý přístup k databázi může rovněž prodlužovat dobu běhu polleru. Řešením může být snížení latence na SQL server (nejlépe tedy databázi provozovat na stejném stroji jako aplikační vrstvu) a zrychlení I/O operací.

  V případě SQL serveru je možné upravit velikosti různých bufferů, maximální počty spojení, zapnout cache tabulek a samozřejmě zajistit co nejrychlejší provádění diskových operací na nižších úrovních[17].

- **Redukce sledovaných parametrů**

  Jelikož každý sledovaný parametr vyvolává vyslání požadavku, přijetí odpověď, zápis na disk a porovnání získané hodnoty s nastavenými limity, je dobré vypnout sledování takových hodnot, které nejsou u daného zařízení podstatné nebo se na něm vůbec nevyskytují (při použití příliš obecných šablon).

- **Zrychlení SNMP dotazů**
  U SNMP `GetBulkRequest` dotazů lze nastavit parametr `max-repetitions`. Úpravou tohoto parametru lze redukovat počet packetů vyměněných mezi agentem a NMS při hromadném vyčítání. V případě vyšší latence mezi NMS a sledovaným zařízením může být nárůst rychlosti výrazný.

- **Využití proxy nebo distribuovaných pollerů**
  Některé NMS (např. Zabbix) umožňují sběr dat v jednotlivých lokalitách pomocí lokální proxy a pak teprve jejich následný reporting do centrálního dohledu. Je tak možné těžit z nižší latence a vyšší rychlosti linek ve sledované LAN a nemusíme se zabývat zpřístupňováním jednotlivých zařízení, pokud se nachází za firewallem či NAT.

2.3 Správa sítí

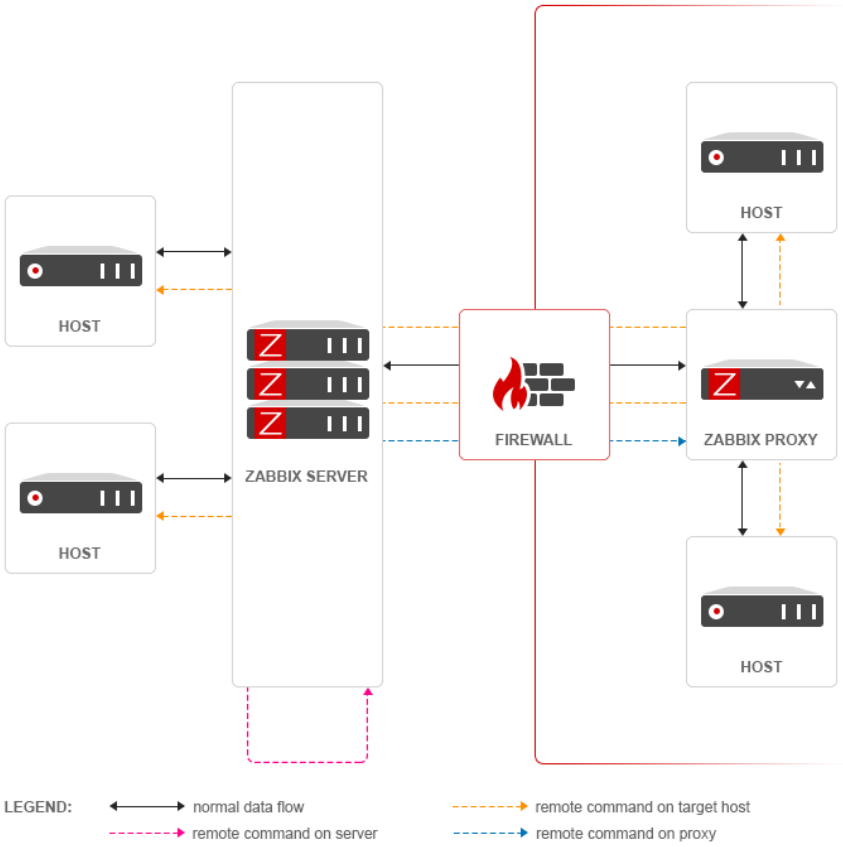
Správa prvků na síti se provádí buď při fyzickém přístupu, např. pomocí konzole, nebo vzdáleně s využitím protokolů vzdálené správy.

2.3.1 Protokoly vzdálené správy

Telnet

Dnes již spíše historický protokol, definovaný především v RFC 854[18], slouží ke správě zařízení přes výhradně textové rozhraní Command Line Interface (CLI). Jedná se o architekturu server - klient, přičemž server poslouchá na portu 23 protokolu TCP.

Vzhledem k tomu, že základní implementace Telnetu neumí šifrovat autentizaci ani přenášená data, je protokol dnes používán jen zcela okrajově, a to u zastaralých nebo velmi jednoduchých zařízení na místní síti.



Obrázek 2.7: Použití Zabbix proxy[29]

Secure Shell (SSH)

Obvyklý způsob vzdálené správy zařízení s operačním systémem GNU/Linux nebo BSD aktuálně představuje protokol SSH ve verzi 2. SSH zajišťuje šifrovaný komunikační kanál mezi klientem a serverem, spojení probíhá přes TCP na portu 22. K autentizaci uživatele slouží jméno a heslo nebo dvojice klíčů.

Použití SSH je velmi široké, kromě poskytnutí CLI přístupu k zařízení je to například přenos souborů přes SCP nebo SSHFS, či vytváření proxy a síťových tunelů.

Popis protokolu je veřejně dostupný v řadě RFC (především RFC 4250[11] až 4256).

Simple Network Management Protocol (SNMP)

Tento protokol je podrobně popsán v kapitole Realizace monitoringu.

Vedle dohledu je možné jej použít i ke správě zařízení, a to pomocí SET požadavků. Při nastavení přes SET by se hodnota z konfiguračního stromu měla vypropagovat do nějaké

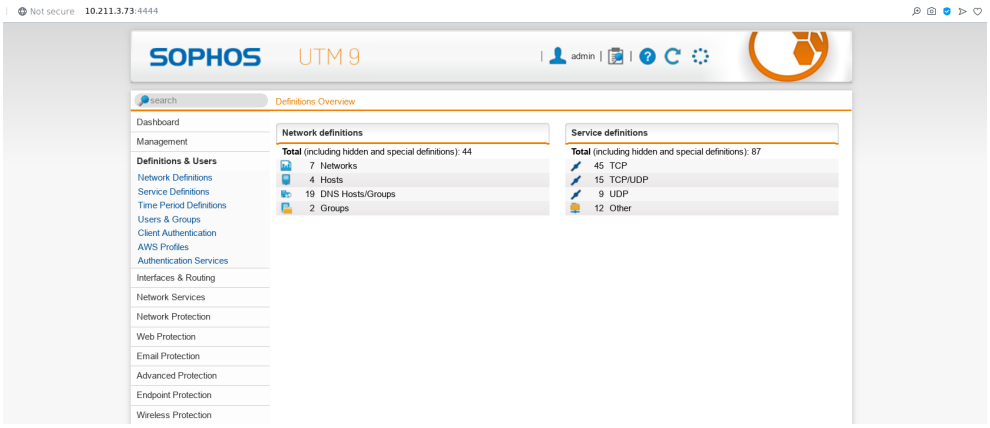
komponenty, která provede změnu v nastavení systému. Velmi špatně by se tak implementovaly komplexnější změny konfigurace, SNMP se tedy pro vzdálenou správu používá spíše okrajově.

HTTP/HTTPS

Velmi obvyklým rozhraním pro správu síťových prvků, především v SOHO oblasti, je webové rozhraní. Uživatel tak přistupuje k zařízení přes standardní webový prohlížeč, na síťovém prvku poté běží HTTP daemon a CGI interpreter. HTTP/1.1 protokol je definován v RFC 2616[6].

Velkou výhodou i nevýhodou HTTP přístupu je jeho variabilita - může znamenat výbornou ergonomii ovládání pro uživatele, ale není ideální pro tvorbu automatizačních nástrojů.

HTTPS je šifrovaná varianta HTTP protokolu.



Obrázek 2.8: Webové rozhraní konfigurace firewallu Sophos

Application Programming Interface (API)

Některé síťové prvky mají možnost konfigurace přes výrobcem popsané API. Může se jednat o zcela proprietární protokol, nebo třeba typicky o REST API, které si vyměňuje data ve formátu JSON a využívá stavových kódů HTTP protokolu[22]. Nad výrobcem definovým API vznikací často knihovny pro oblíbené programovací jazyky, které ještě více usnadňují přístup k zařízení z vyvíjeného programu.

Nevýhodou je, že každý výrobce definuje vlastní protokol komunikace, nelze tedy z jednoho „API klienta“ řídit zařízení více výrobců.

NETCONF

NETCONF protokol je popsán v RFC 6241[5] a je určen ke správě i dohledu zařízení. Komunikace probíhá pomocí metody Remote Procedure Call (RPC) s přenosem dat ve formátu XML přes zabezpečené spojení TLS nebo protokol SSH, obvykle se využívá serveru, který poslouchá na TCP portu 830.

Konfigurace zařízení je přesně popsána v modelu. NETCONF klade velký důraz na konzistenci prováděných změn, definuje tři základní typy konfigurace `running`, `startup` a `candidate`, které jsou uchovány v datastoru ve formátu XML. Konfiguraci lze zamknout, aby se zabránilo neintegritě nastavení, pokud by došlo ke konfiguraci z více míst současně.

2.3.2 Automatizace úkonů

V sítích s větším počtem zařízení stejného či podobného určení je účelné proces jejich nastavení automatizovat. Přístup je v zásadě dvojí - buď definujeme cílový stav a nástroj se postará o jeho realizaci, nebo popíšeme sekvenci kroků, které k cíli vedou, a nástroj se postará o jejich vykonání ve správném pořadí na správných strojích.

Typickými zástupci automatizačních nástrojů jsou Ansible[20] nebo Puppet[19].

Ansible

Ansible je opensource automatizační software, který kombinuje oba výše zmíněné přístupy, tedy ad hoc provádění úloh a software deployment. Pro přístup ke vzdáleným počítačům používá primárně SSH, nikoli agenta. Je tak šetrný k systémovým zdrojům ovládaných strojů a nevyžaduje na nich jeho instalaci.

Uživatel definuje inventář hostů k ovládání. Dále specifikuje tzv. playbooky ve formátu YAML, které přiřazují jednotlivým hostům či jejich skupinám role. Role se skládají z volání modulů pro vykonávání samotných příkazů na vzdáleném stroji.

Software si zakládá na minimalismu a vysoké spolehlivosti. Díky způsobu návrhu modulů by mělo být zaručeno, že opakovaným spouštěním playbooku dojdeme vždy ke stále stejnému výsledku, a bude tak jednoduše možné realizovat zotavení z chyby, která nastala v průběhu nasazování nové konfigurace.

Puppet

Puppet je rovněž automatizační opensource software, proti Ansiblu má ale ryze deklarativní přístup, kdy uživatel specifikuje požadovaný stav v tzv. manifestech pomocí vlastního Puppet jazyka.

Pro konfiguraci ovládaných hostů slouží agent, který pomocí REST API přistupuje k masterovi, na kterém jsou uloženy manifesty. Tyto manifesty zpracuje a přeloží dle svých specifik do konkrétních příkazů, které provede.

2.3.3 Vedení dokumentace

Aby byla možná efektivní správa sítě, vede se k síti dokumentace. Ta by měla být vhodného rozsahu podle určení a rozsáhlosti sítě a zahrnuje například:

- Fyzický popis rozvaděčů, počítačů, kabelů, portů, napájecích větví.
- Databáze majetku s vazbou na IT, přidělení zařízení konkrétním osobám, přehled nad fyzickými adresami prvků na síti.

- Adresní plány pro alokaci IP adres v organizaci.
- Schémata topologie sítě.
- Provozované služby, jejich provázání a specifika konfigurace.
- Správa softwarových licencí.
- Bezpečné uchování přístupových údajů k zařízením a systémům.

Vzhledem k různorodosti údajů budou k jejich uchování sloužit rozdílné systémy. S větším množstvím takových systémů může docházet při změnách v síti k nekonzistenci údajů.

Pro profesionální použití jsou definovány normy specifikující procesy kolem tvorby a uchování dokumentace.

Kapitola 3

Porovnání nástrojů pro monitoring

Pro správu i dohled sítě lze používat celou řadu nástorojů, které se liší svou komplexitou i určením. V této kapitole se pokusím porovnat některé oblíbené a volně dostupné nástroje.

Pro účely porovnání padla volba na čtyři systémy - Zabbix, Icinga, Observium a UNMS. První dva jsem zvolil kvůli jejich popularitě, Observium pro výraznou míru autokonfigurace a UNMS jako zástupce proprietárního systému od dodavatele hardwaru. Všechny uvedené systémy jsem měl zároveň k dispozici na produkčních sítích s vyššími desítkami či stovkami zařízeními.

3.1 Sledované parametry

V rámci porovnání budu u dohledových systémů zkoumat rozsah funkcionality, možnosti rozšíření a integrace.

Rozsah funkcionality

- **Metoda získávání dat**
  Způsob, jakým jsou ze zařízení vyčítána sledovaná data.
- **Rozsah dohledu**
  Posouzení rozsahu dohledu - zda se jedná jen o kontrolu dostupnosti, podporu kreslení grafů vytížení, zjišťování detailů o běžících službách či dokonce schopnost konfigurace sledovaných prvků.
- **Autokonfigurace**
  Míra zahájení a nastavení parametrů dohledu nového zařízení bez účasti uživatele.
- **Způsob upozornění na problémy**
  Rozsah možností, jak umí systém komunikovat s uživatelem ohledně nalezených problémů.
- **Mapy sítě**
  Schopnost kreslit grafické mapy sítě na základě vyčtených údajů. Může se jednat o mapy geografické polohy, znázornění topologie, aktuálních datových toků apod.

- **Škálovatelnost**
  Podpora distribuovaného pollingu, proxy serverů.

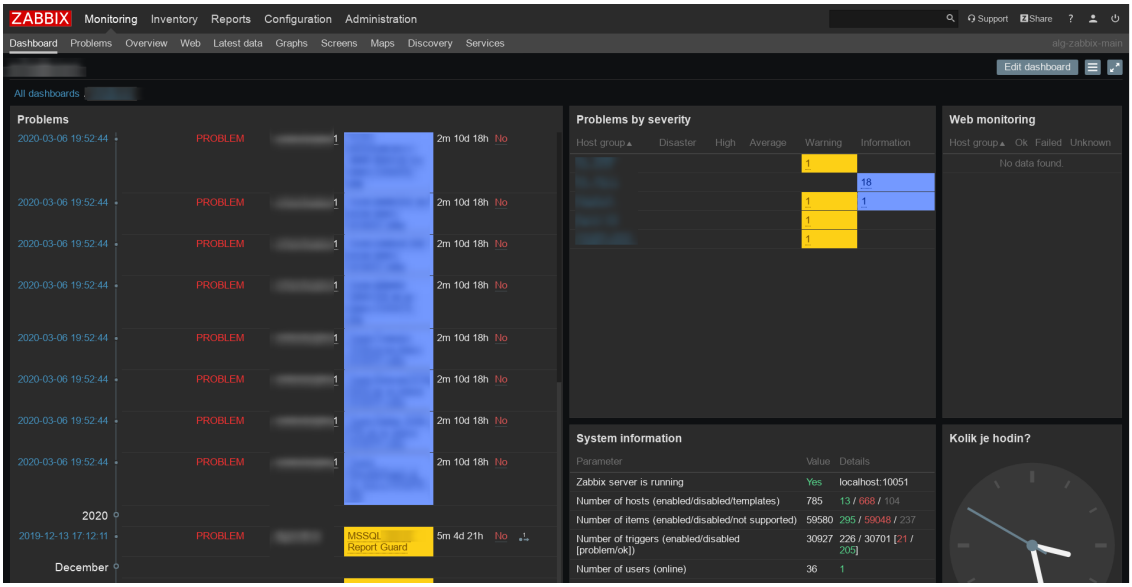
Rozšířitelnost a integrace

- **Možnost šablon/skupin**
  Míra organizace sledovaných zařízení dle pravidel.
- **Vlastní pluginy**
  Možnost doplnit systém o další funkcionalitu.
- **Přístup přes API**
  Popis, zda NMS poskytuje přístup přes API, aby jej bylo možné ovládat z ostatních systémů.

3.2 Popis vybraných nástrojů

3.2.1 Zabbix

Jedná se o opensource software s velkou variabilitou nastavení. Vývoj a podporu zastřešuje Zabbix LLC. Je určený pro sledování širokého spektra služeb i údajů o vytížení systému a sítě.



Obrázek 3.1: Zabbix - uživatelské rozhraní

Metoda získávání dat:
Multiplatformním agentem nebo pomocí SNMP, IPMI či ICMP.

Rozsah dohledu:
Podrobný dohled síťového i systémového vytížení, detaily stavu velkého množství služeb. Podporuje vytváření grafů na základě získaných dat.

Autokonfigurace:
Podpora autodiscovery jak na úrovni metrik ke sledování u jednoho zařízení, tak i automatické nalezení nových zařízení pomocí skenu IP rozsahů.[28]

Způsob upozornění na problémy:
Několik úrovní závažnosti problému, eskalace notifikace v případě delšího trvání problému na další kontakty či komunikační kanály, podpora emailů, XMPP a dalších.

Mapy sítě:
Podpora uživatelem sestavených map, které mohou vizualizovat topologii i sbírané hodnoty.

Škálovatelnost:
Již v návrhu se počítá s architekturou Zabbix serveru a k němu připojených několik Zabbix proxy, které se nachází za firewallem v jednotlivých lokalitách.[29]

Možnost šablon/skupin:
Zabbix podporuje šablony, které může uživatel mapovat na sledovaného hosta. Rovněž lze do systému importovat šablony nové, jsou dostupné ve formátu XML. Uživatel také může shlukovat sledovaná zařízení do jím vytvořených skupin.

Vlastní pluginy:
Je možné vytvářet vlastní nebo použít pluginy třetí strany. Dokumentace k tvorbě pluginů je volně dostupná.

Přístup přes API:
Zabbix poskytuje API pro zápis i čtení, komunikace probíhá pomocí přenosu zpráv ve formátu JSON přes HTTP protokol.

3.2.2 Icinga

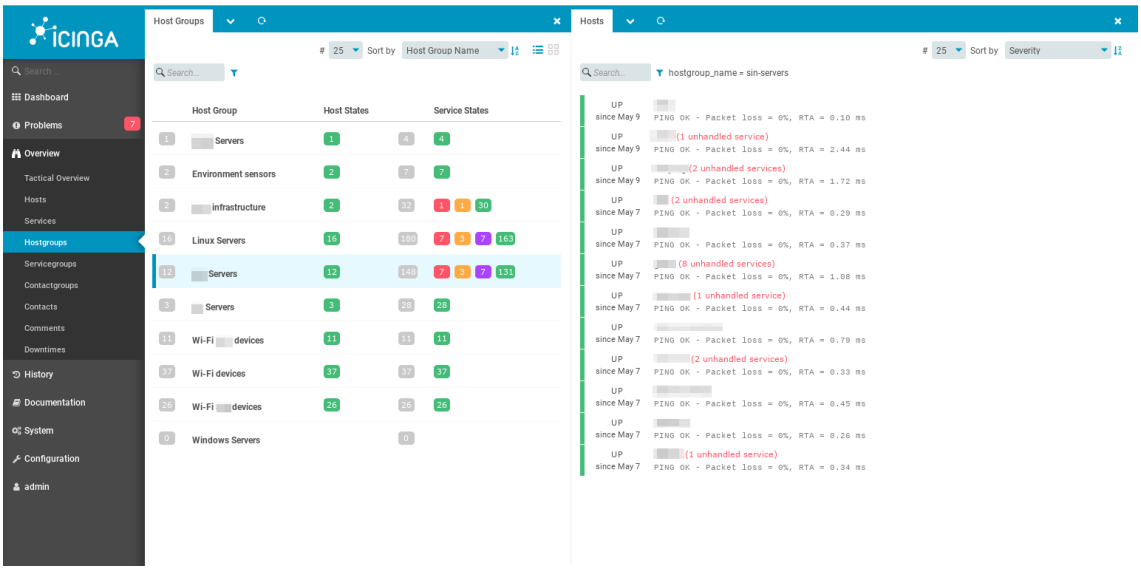
Icinga je rovněž opensource projekt[9], začal jako fork populárního systému Nagios. Díky tomu s ním poskytuje jistou zpětnou kompatibilitu, zejména podobný způsob konfigurace a využití Nagios Remote Procedure Execution (NRPE). Uživateli ale nabízí řadu funkcí navíc, jako třeba moderní webové rozhraní. Konfigurace sledovaných zařízení a služeb probíhá standardně úpravou konfiguračního souboru, je ale možné použít i modul, který tuto funkcionalitu zpřístupní i ve webovém rozhraní.

Metoda získávání dat:
Doporučené je monitorování pomocí agenta, podporováno je ale i SNMP, SSH, WMI a další.

Rozsah dohledu:
Podrobný dohled síťového i systémového vytížení, detaily stavu velkého množství služeb. V základní instalaci nepodporuje tvorbu grafů, lze ale doinstalovat modul pro jejich podporu[7].

Autokonfigurace:
Není podporována, lze naprogramovat skenovací script s voláním API.

Způsob upozornění na problémy:
Několik úrovní závažnosti problému, eskalace notifikace v případě delšího trvání problému na další kontakty či komunikační kanály, podpora emailů, SMS a dalších.



Obrázek 3.2: Icinga2 - Webové uživatelské rozhraní

Mapy sítě:
Ve výchozí instalaci není podporováno, lze doplnit modulem.

Škálovatelnost:
Icinga již v návrhu počítá s distribuovaným monitoringem s topologií stromu. Na vrcholu je master node, na který jsou napojeny satelity nebo koncová zařízení / agenti[8].

Možnost šablon/skupin:
Do skupin je možné shlukovat služby i hosty.

Vlastní pluginy:
Je možné vytvářet vlastní nebo použít pluginy třetí strany. Dokumentace k tvorbě pluginů je volně dostupná. Navíc je Icinga kompatibilní s pluginy původně určenými pro Nagios.

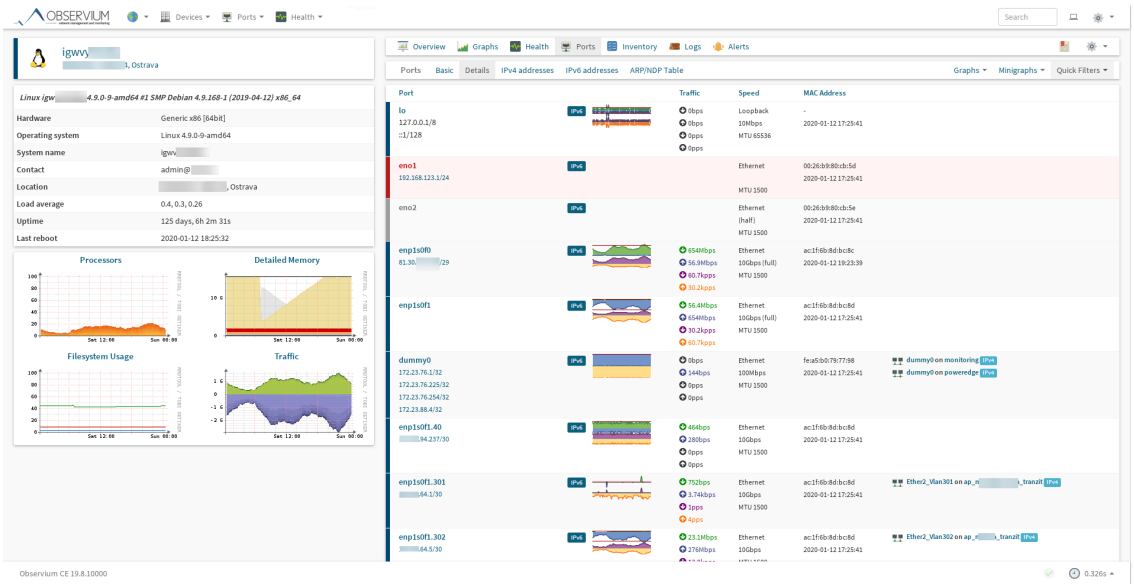
Přístup přes API:
Icinga poskytuje API pro zápis i čtení, komunikace probíhá pomocí přenosu zpráv ve formátu JSON přes HTTP protokol.

3.2.3 Observium

Observium je monitorovací platforma licencovaná pod zjednodušenou formou QPL s nízkými nároky na konfiguraci, která si zakládá na kvalitě uživatelského rozhraní[16]. Nabízí se ve volně dostupné community verzi nebo v placené edici. Tyto verze se liší množstvím funkcionality a poskytovanou podporou.

Metoda získávání dat:
Monitoring probíhá primárně přes protokol SNMP. Možnost dohledu vybraných služeb pomocí agenta.

Rozsah dohledu:
Podrobný dohled síťového i systémového vytížení, sledování ARP/NDP/FDB tabulek pro



Obrázek 3.3: Observium- Webové uživatelské rozhraní

nalezení sousedů. V komunitní verzi nelze sledovat vlastní OID, uživatel je odkázán na existující šablonu pro daný prvek.

Autokonfigurace:
Pravidelně je prováděno vyhledávání podporovaných funkcionalit monitorovaných hostů, které objeví síťová rozhraní, hardwarové senzory a podobně. Rovněž existuje podpora automatického monitoringu zařízení, které byly nalezeny pomocí informací z již monitorovaných hostů (např. pomocí CDP nebo LLDP).

Způsob upozornění na problémy:
Několik úrovní závažnosti problému, podpora emailů, XMPP, platforem pro Instant Messaging (např. Telegram) a dalších.

Mapy sítě:
Ve výchozí instalaci umí zobrazovat geografické mapy se znázorněním stavu prvků v lokalitě. Na základě dat z discovery protokolů podporuje tvorbu map odhadované topologie sítě.

Škálovatelnost:
Částečná podpora pomocí Poller Partitioning, kde podřízené instance polleru posílají získaná data přes síťový přístup k MySQL a RRDCached na centrální server[14].

Možnost šablon/skupin:
Šablony pro dohled prvků jsou dodávány v základní instalaci. Systém automaticky detekuje, kterou šablonu má pro dané zařízení použít. Seskupování se provádí do předvolených skupin dle typu zařízení, pokročilé seskupování na základě uživatelem definovaných pravidel je dostupné jen v placených verzích.

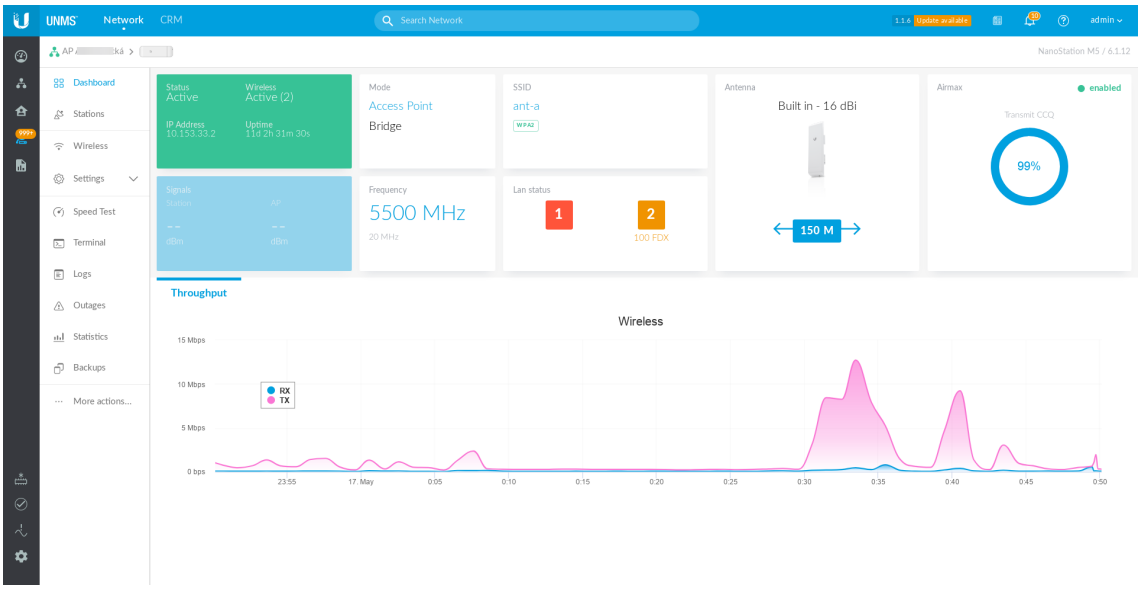
Dle mého názoru jsou dodávané šablony poměrně kvalitní a poskytují výběr těch podstatných údajů pro každou kategorii síťových prvků. Uživatel může v rozsahu relativně restriktivní licence vytvářet či upravovat šablony pomocí editace PHP souborů, nebo požádat o jejich vytvoření autorem programu, jedná-li se o platícího zákazníka.

Vlastní pluginy
Systém nemá rozhraní pro pluginy.

Přístup přes API
Uživatelé placené verze mají k dispozici jednak REST API pro vyčítání a zápis konfigurace a sledovaných dat, a také API pro přístup ke grafům.

3.2.4 UNMS

UNMS je vyvíjeno společností Ubiquiti Networks (UBNT) jako monitorovací a konfigurační nástroj určený primárně pro Internet Service Provider (ISP) firmy používající bezdrátové nebo optické prvky této firmy[27]. Kromě NMS integruje i CRM a umožňuje konfiguraci vybraných síťových prvků. UNMS není uvolněno pod opensource licencí, je ale dostupné zdarma.



Obrázek 3.4: UNMS- Webové uživatelské rozhraní

Metoda získávání dat:
V případě zařízení UBNT probíhá monitoring přes proprietárního agenta, u zařízení jiných vendorů přes SNMP.

Rozsah dohledu:
Pro zařízení UBNT umožňuje podrobný dohled síťového vytížení, detailních parametrů fyzické vrstvy téměř v reálném čase a podrobností konfigurace. U cizích zařízeních je dostupný pouze základní dohled dostupnosti či stavu portů.

Autokonfigurace:
Pro zařízení UBNT umožňuje automatické zjištění přesného modelu a tedy i rozsahu funkcionalit k monitorování, u cizích zařízení je pouze možnost vyčtení seznamu síťových rozhraní a IP adres. Součástí je integrovaný skener sítě, který umí vyhledat zařízení podle discovery protokolu nebo SNMP.

Způsob upozornění na problémy:
Několik úrovní závažnosti problému, podpora notifikace emailem.

Mapy sítě:
Umí kreslit geografické mapy se znázorněním stavu a vytížení prvků a tras v reálném čase, pouze ale pro zařízení od UBNT.

Škálovatelnost:
Není možnost pollingu na více strojích, v budoucnosti je funkcionalita plánována[26].

Možnost šablon/skupin:
Šablony pro UBNT zařízení jsou součastí systému, pro cizí zařízení je předdefinováno několik šablon (router, switch, UPS), uživatel nemůže vytvářet vlastní.

Zařízení jsou uživatelem přiřazována do skupin, které reprezentují fyzické lokality vysílačů.

Vlastní pluginy
NMS systém nemá rozhraní pro pluginy.

Přístup přes API
K dispozici je REST API přenášející zprávy ve formátu JSON, které je podrobně popsáno v dokumentaci.

3.3 Shrnutí, doporučení

Všechny testované nástroje jsou vhodné pro produkční nasazení, jejich primární určení se ale liší. Dovolím si shrnout subjektivní názory vycházející z analýzy a zkušeností s provozem vybraných systémů.

Zabbix je univerzální nástroj pro dohled jak služeb tak i transportní sítě, nabízí široké možnosti nastavení a dostupných šablon. Přestože konfigurace probíhá čistě ve webovém prostředí, není pro nového uživatele příliš intuitivní, zkušený uživatel ale může dosáhnout vysoké míry přizpůsobení specifickým potřebám.

Icinga je výborný nástroj pro sledování sítě, která je orientována více na poskytování služeb než přenos dat. Velmi elegantně je možné systém rozšířit o funkcionalitu pro detailní dohled šitý na míru určité aplikaci, řada uživatelů bude také těžit z existence stovek pluginů pro Nagios[13]. Hůře by se realizoval komfortní dohled nad datovými toky na rozhraních síťových prvků, konfigurace v textových konfiguračních souborech je spíše pro zkušenější uživatele.

Observium nabízí velmi intuitivní uživatelské rozhraní s nutností minimálních zásahů do nastavení způsobu monitorování hosta. Je výborné, že se zařízení sama přiřadí šablona a skupina pro dohled, stinnou stránkou toho ale je horší možnost přizpůsobení specifickým potřebám, problémový dohled exotického prvku, pro který neexistuje šablona, nebo dokonce nemožnost monitorovat zařízení, které nepodporuje SNMP protokol. Systém také není určen primárně pro dohled dostupnosti služeb, sbírá spíše statistická data z jejich provozu. Pro nekomerční nasazení může být také limitující omezenost komunitní verze.

UNMS jsem zvolil jako typického zástupce proprietárních systémů vydávaných výrobcem hardwaru. UNMS přestavuje nástroj pro úzce mířenou cílovou skupinu malých a středně velkých ISP, kteří používají hardware stejného výrobce. Systém šikovně integruje NMS a

CRM, díky čemuž je ihned vidět vazba problému na síti na konkrétní klienty, umožňuje také vzdálenou konfiguraci prvků z centrálního místa webového prohlížeče nebo mobilní aplikace. Bohužel, všechny tyto zajímavé funkce jsou dostupné pouze pro zařízení vyráběné daným výrobcem, protože při jejich realizaci není použit otevřený standard. Uživatel se tak může časem dostat do situace vendor lock-in, kde je postaven proti volbě nakupovat hardware jen jednoho výrobce, který může být cenově nebo parametry horší než konkurence, nebo oželet pokročilé funkce systému.

Kapitola 4

Analýza a návrh automatizačního nástroje

V rámci studia jsem měl možnost nahlédnout do infrastruktury datacentra společnosti Algotech, které je svým rozsahem dostatečně velké na to, aby bylo velmi výhodné řadu procesů optimalizovat a automatizovat. Dostal jsem k dispozici několik hypervizorů a přístup k různým systémům sloužících pro dohled a správu sítě.

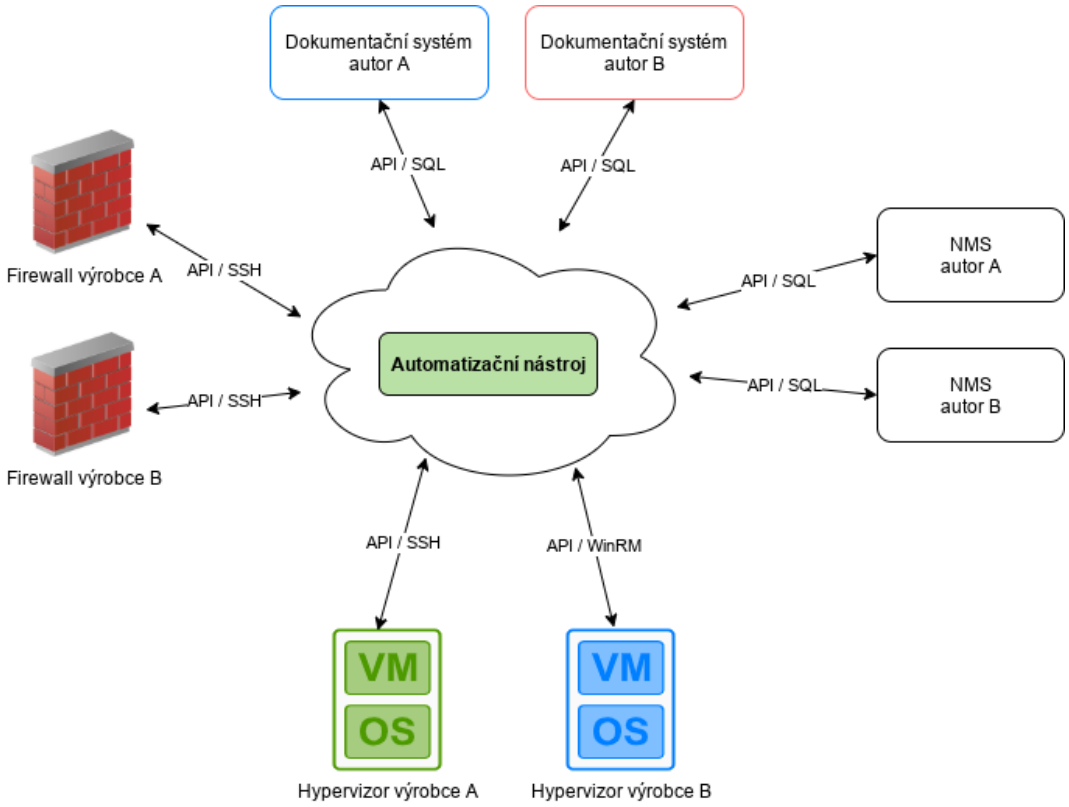
Jako klíčové se ukázalo mít na všech systémech aktuální údaje o skutečném stavu sítě. Jelikož jsou tyto systémy ze své povahy velmi různorodé a jejich vzájemné propojení obecně složité, vznikl nápad návrhu integračního nástroje.

Navrhovaný nástroj musí být schopen komunikovat pomocí různých protokolů vzdálené správy s různými systémy různých výrobců. Toho bylo docíleno modulární architekturou, kdy je možné velmi jednoduše doplnit podporu pro další systém. Každý modul specifikuje množinu podporovaných operací, obvyklá je komunikace přes REST API, které zveřejňuje výrobce daného systému.

Žádoucí je rovněž minimalizovat manuální činnost uživatele/správce, systém se tedy v případě nalezeného problému snaží sám navrhnout vhodné řešení, které uživatel může v rámci systému poupravit a na jedno kliknutí provést ze systému opravu.

4.1 Popis prostředí

Prostředí, se kterým bude nástroj pracovat, je různorodé a může se v průběhu času měnit. Nástroj komunikuje především s hypervizory, firewally, dokumentačními systémy a Network monitoring system (NMS). Komunikace probíhá zpravidla přes API a je čistě záležitostí modulu pro daný systém.



Obrázek 4.1: Popis prostředí

V rámci práce byly zpracovány moduly pro systémy, které jsem měl k dispozici pro testování. Jednalo se o hypervizor VMware ESXi, firewall Sophos UTM, monitorovací systém Zabbix a dokumentační nástroj phpIPAM.

4.2 Požadavky

Součástí procesu návrhu softwaru je přesná specifikace funkčních a nefunkčních požadavků. Funkční požadavky představují samotnou funkcionalitu softwaru, nefunkční pak vlastnosti programu důležité pro jeho provoz a udržitelnost.

4.2.1 Funkční požadavky

Správa uživatelských účtů

Systém umožní zobrazení přehledu, vytváření, úpravu a mazání uživatelských účtů. Rozhraní systému je dostupné až po příhlášení, každý uživatel se do systému přihlašuje svým uživatelským jménem a heslem, které si může změnit. Uživateli lze nastavit emailovou adresu, celé jméno a odběr emailových uporoznění.

Správa inventáře komponent infrastruktury

Uživatel může zobrazovat, vytvářet, upravovat vlastnosti a mazat záznamy v inventáři komponent infrastruktury v kategoriích Hypervizory, Firewally, Dohledové systémy a Dokumentační systémy.

Vlastnostmi záznamu se rozumí parametry připojení k rozhraní vzdálené správy dané komponenty. Seznam požadovaných parametrů poskytne konkrétní modul pro určitou komponentu. Uložené nastavení bude možné jednoduše otestovat mimo proces kontroly nastavení celé infrastruktury.

Definování pravidel kontroly nastavení jednotlivých komponent

Systém umožní prohlížet, upravovat a mazat tzv. „uživatelské scénáře“, ve kterých uživatel specifikuje přesné požadavky pro provedení kontroly nastavení jednotlivých komponent. Bude možné specifikovat množinu prvků, pro kterou se nastavení kontroluje, samotné požadované nastavení a množinu komponent, na které má být nastavení přítomno. Součástí scénáře bude také to, jaká je navrhovaná oprava v případě zjištění nesouladu.

Provedení kontroly nastavení komponent infrastruktury

Kontrola nastavení bude probíhat v pravidelných intervalech dle uživatelem definovaných scénařů. Bude možné ji také spustit ručně ve webovém rozhraní.

Výsledkem kontroly bude seznam nalezených chyb, které bude možné řešit nebo ignorovat. Pokud se uživatel rozhodne chyby ignorovat, nebude na ně již při dalším běhu kontroly upozorňován. V případě řešení problému bude uživateli na základě scénáře a informací, které modul získá z komponenty, navrhnut postup řešení, které bude nástroj schopen sám po kontrole a potvrzením uživatelem provést.

Záloha a obnova nastavení

Systém bude umožňovat zálohu a také obnovu nastavení.

Emailová upozornění

Uživatel bude moct dostávat emailová upozornění o nových nalezených problémech s nastavením komponent. Systém bude využívát již existující mailový server.

Logování událostí

Všechny podstatné úkony provedené uživatelem nebo automaticky systémem budou zaznamenány v logu událostí, aby je bylo možné zpětně dohledat. V logu bude možné filtrovat podle klíčových slov.

4.2.2 Nefunkční požadavky

Rozšířitelnost pro další platformy

Návrh by měl počítat s možností přidání podpory pro další platformy komponent. Měl by také reflektovat, že různé platformy mají různou množinu operací, které jsou schopny provést.

Webové rozhraní

Komunikace s uživatelem bude probíhát pomocí webového rozhraní. Pro všechny operace nad infrastrukturou bude požadováno přihlášení uživatele pomocí jména a hesla. Vzhled rozhraní by měl být přízpůsoben pro různé velikosti obrazovek.

Zabezpečená komunikace

Vzhledem k povaze nástroje je podstatný důraz také na zabezpečení komunikace s jednotlivými komponentami infrastruktury.

Bezpečnost je komplexní problém, v rámci návrhu této aplikace se jedná především o nastavení vhodných přístupových práv pro účty využívající API, šifrování komunikace přes API a omezení dostupnosti managementu komponent pomocí firewallu. Vzhledem k použití webové technologie je již samozřejmá komunikace s uživatelem přes protokol HTTPS a ukládání uživatelských hesel do databáze s využitím metod, které znesnadňují jejich zneužití.

Využití opensource technologií

Nástroj by měl být provozován na operačním systému Linux, je tedy žádoucí, aby využíval obvyklou sadu nástrojů a aplikací na tomto systému dostupnou.

4.3 Programovací jazyk a prostředí

Při volbě programovacího jazyka byly zváženy funkční i nefunkční požadavky na software a byl vybrán jazyk Python, především kvůli dostupnosti knihoven pro práci s API.

Python je vysokoúrovňový interpretovaný programovací jazyk, který se hojně používá v programech a skriptech pro operační systém Linux. Jazyk je vyvíjen jako open source projekt, k dispozici je velké množství knihoven rozšiřujících jeho funkcionalitu.

Vzhledem k požadavku na webové uživatelské rozhraní jsem se rozhodl využít framework Flask. Framework je obecně softwarová platforma, která zjednodušuje vývoj softwaru tím, že poskytuje základní stavební bloky a funkce. Flask je microframework, potřebná funcionalita

jako ORM nebo validace formulářů je poskytována formou rozšíření. Základními komponentami Flasku jsou šablonovací engine Jinja a knihovna Werkzeug, která zajišťuje WSGI rozhraní.

4.4 Architektura softwaru

Software je navržen tak, že každý blok zajišťuje část funkcionality.

- **Databáze**

  Z definovaného modelu tříd je pomocí ORM vygenerována struktura databáze, ve které program uchovává veškerá data. Jako backend slouží MariaDB.

- **Webové rozraní**

  Pro zpřístupnění webového rozhraní uživateli je potřeba HTTP server. Přestože má Flask integrovaný Web Server Gateway Interface (WSGI), je použit propracovanější `gunicorn` server.

  Samotné webové stránky jsou složeny z HTML šablon, CSS kaskádových stylů a pomocných scriptů v jazyce JavaScript. Zpracování šablon je prováděno interně pomocí nástroje Jinja, který je součástí frameworku Flask.

- **Fronta úloh / Task queue**

  Důležitou částí při provádění dlouhých skenování sítě je implementace asynchronního zpracování úloh. Pokud by uživatelem spuštěný sken probíhal synchronně, musel by uživatel čekat na načtení stránky klidně několik minut, během kterých by neměl přehled o průběhu procesu a mohlo by také dojít k timeoutu HTTP requestu.

  K řešení této části je použit `Redis server`, který slouží jako message broker. S ním komunikuje program pomocí knihovny Redis Queue.

- **Poller / Skener**

  Část sloužící ke skenování sítě a kontrole pravidel definovaných v uživatelských scénářích. Je spouštěna asynchronně pomocí Task queue.

- **Solver**

  Tento modul provádí návrh a také řešení nalezeného problému. Ve velké míře využívá podklady z ovladačů pro kokrétních platformu síťových komponent.

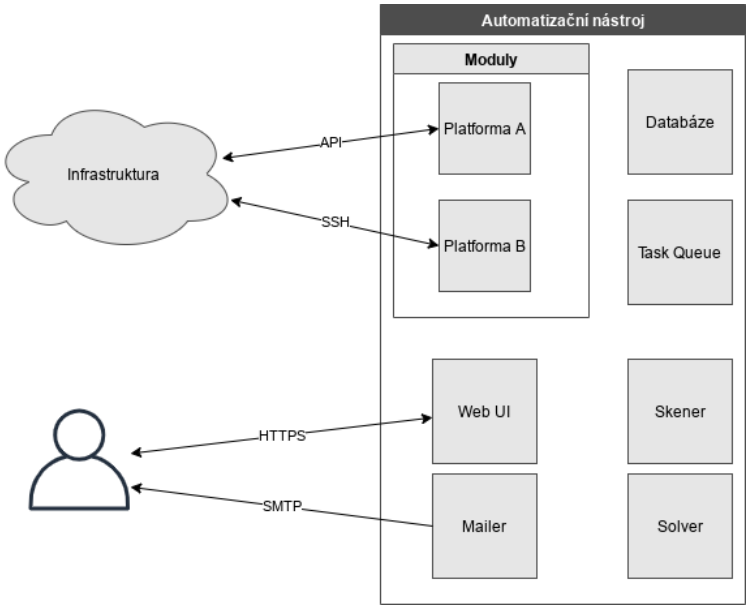
- **Mailer**

  Mailer je jednoduchý modul, který zajišťuje zasílání emailů s upozorněními uživateli.

- **Controllery pro ovládání infrastruktury**

  Pro komunikaci s jednotlivými komponentami infrastruktury se používají controllery. Ten je vždy specifický pro danou platformu, se zbytkem softwaru ale komunikuje přesně definovaným způsobem.

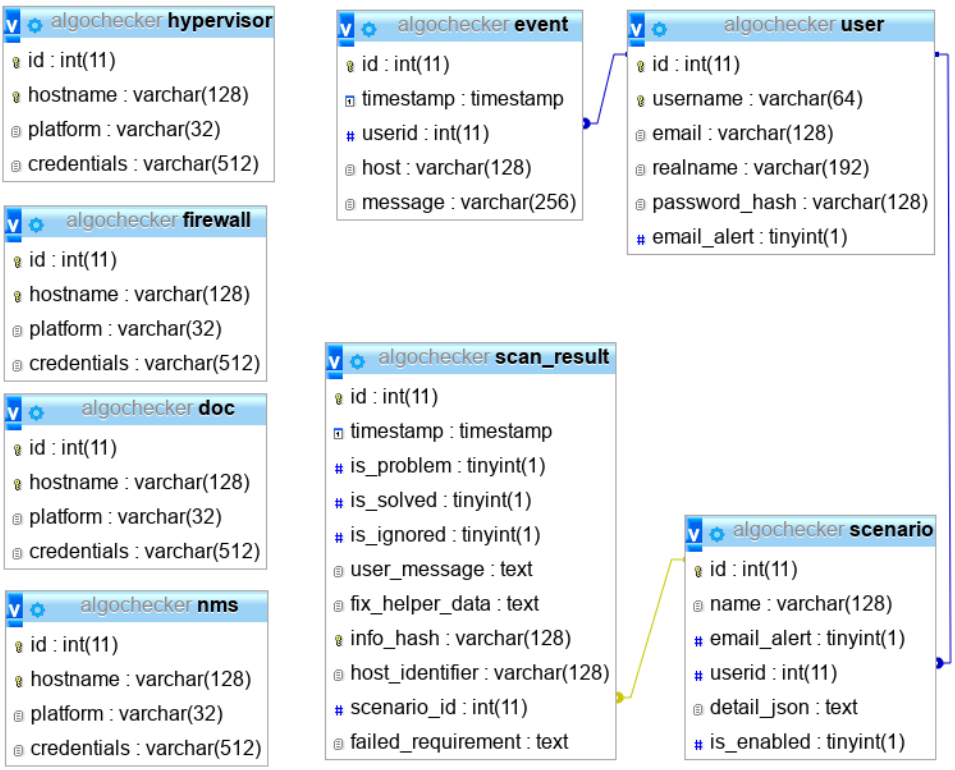
  Jednotlivé controllery jsou načítány pomocí `importlib` a až v případě potřeby.



Obrázek 4.2: Architektura softwaru

4.5 Databázový model

Databázový návrh obsahuje následující tabulky, jejichž názvy jsou odvozeny od jednotlivých tříd.



Obrázek 4.3: Databázový model

- **User**

  Seznam uživatelů systému, jejich kontaktní údaje a heslo.

- **Event**

  Seznam položek logu událostí, je zde vazba na konkrétního uživatele, který akci vyvolal.

- **Scan_result**

  Výsledky provedeného skenu infrastruktury. Každý nesplněný požadavek z uživatelského scénaře pro každého hosta znamená jeden záznam. Stejně tak zcela úspěšný průchod scénářem vytvoří záznam pro každého hosta. Pomocí cizího klíče je záznam vázán na scénář, při kterém došlo k úspěchu nebo chybě.

- **Scenario**

  Zde jsou uloženy uživatelem vytvořené scénáře pro kontrolu sítě.

- **Firewall, Hypervisor, Doc, NMS**

  Položky inventáře, je zde uložené nastavení pro přístup k jednotlivým komponentám infrastruktury.

4.6 Struktura zdrojového kódu

Zdrojový kód softwaru je pro přehlednost a udržitelnost členěn do několika souborů a balíčků / složek.

- `config.py`
  V tomto souboru je přítomno nastavení aplikace, jako jsou údaje pro připojení k databázi, URL, na kterém je aplikace vystavena uživateli, přístupové údaje k mailserveru pro zasílání upozornění a další.

- `app/__init__.py`
  Centrální prvek celé aplikace. Zde bude načítan Flask, jeho potřebné moduly, nastavení a poté inicializován router, modely a další části aplikace.

- `app/email.py`
  Tento soubor obsahuje třídu `MailService`, tedy kód pro zajištění emailové komunikace.

- `app/models.py`
  Zde jsou definovány třídy, které jsou pomocí ORM `SQLAlchemy` mapovány proti databázi. Jedná se o `Event`, `Scenario`, `ScanResult`, `User`, `Doc`, `Nms`, `Firewall` a `Hypervisor`.

- `app/translator.py`
  Zde je definována třída `Translator`, která zajištěje překlad zpráv a konfigurace ve formátu JSON do lidsky čitelné podoby.

- `app/solver.py`
  V tomto souboru se nachází třída `Solver`, která navrhuje opravu skenerem nalazeného nesouladu a poté ji provádí.

- `app/poller.py`
  Obsahem je definice třídy `Poller` a metoda `poll()`. Tato metoda je spouštěna asynchronně pomocí `Redis Queue`.

- `app/forms.py`
  Tento soubor obsahuje definice formulářů s využitím `WTForms`. Každý formulář v uživatelském rozhraní je popsán v samostatné třídě, jejíž proměnné jsou prvky formuláře.

- `app/routes.py`
  Zde se nachází router webové aplikace. Na základě anotací jsou zde definovány URL cesty a k nim náležící metody.

- `controllers/supercontroller.py`
  Obsahem je definice třídy `SuperController`, která zjednodušuje přístup ke controllerům jednotlivých komponent.

- `controllers/nms/nms_controller.py`
  V tomto souboru je definována třída `NmsController`, která zajišťuje abstrakci nad moduly pro komunikaci s dohledovými systémy.
- `controllers/nms/zabbix4.py`
  Modul pro komunikaci s dohledovým systémem Zabbix verze 4.
- `controllers/hypervisor/hypervisor_controller.py`
  V tomto souboru je definována třída `HypervisorController`, která zajišťuje abstrakci nad moduly pro komunikaci s hypervizory.
- `controllers/hypervisor/vmware_esxi.py`
  Modul pro komunikaci s hypervizorem na platformě VMware ESXi.
- `controllers/doc/doc_controller.py`
  V tomto souboru je definována třída `DocController`, která zajišťuje abstrakci nad moduly pro komunikaci s dokumentačními systémy.
- `controllers/doc/phpipam.py`
  Modul pro komunikaci s dokumentačním systémem phpIPAM.
- `controllers/firewall/firewall_controller.py`
  V tomto souboru je definována třída `FirewallController`, která zajišťuje abstrakci nad moduly pro komunikaci s firewally.
- `controllers/firewall/sophos.py`
  Modul pro komunikaci s firewallem Sophos.

4.7 Definované třídy

Třídy databázového modelu

Mezi třídy databázového modelu, které jsou pomocí ORM mapován na databázi, patří následující výčet. Většinu instančních proměnných daných tříd není třeba detailně popisovat, protože jejich název je dostatečně vypovídající, zaměřím se tedy jen na méně zjevné proměnné.

- **ScanResult(db.Model)**
  Významné proměnné:
  - `host_identifier` je textová reprezentace názvu hypervizoru, hostname a IP adresy.
  - `failed_requirement` obsahuje slovník, ve kterém je uložen `requirement` ze scénáře, který při kontrole selhal.
  - `info_hash` je SHA256 hash proměnných `host_identifier`, `user_message` a `scenario_id`. Slouží k jednoznačné identifikaci konkrétního problému, aby se při opakovaném skenu problémy neduplikovaly.
  - `fix_helper_data` obsahuje slovník s relevantními údaji o selhání kontroly.

- **Scenario(db.Model)**
  Proměnná `detail_json` obsahuje slovník se strukturovaným popisem scénáře - selektoru, kvantifikátorů apod.
- **Doc(db.Model)**
  Proměnná `credentials` obsahuje dictionary, jehož obsahem jsou parametry pro připojení ke komponentě. Podoba slovníku se pro různé platformy liší a je proto definována v každém modulu.
- **Hypervisor(db.Model)**
- **Nms(db.Model)**
- **Firewall(db.Model)**
- **Event(db.Model)**
- **User(UserMixin)**

V těchto třídách jsou definovány metody pro vytváření, čtení, editaci a mazání instancí z databáze a podobné obslužné metody.

Třídy formulářů WTForms

Následující třídy definují formuláře, které využívají modulu `WTForms`. Tento modul zjednodušuje generování HTML prvků formuláře, validaci a zpracování vstupních dat. Každá třída reprezentuje jeden takový formulář v aplikaci.

- LoginForm(FlaskForm)
- SolveIssueForm(FlaskForm)
- NewDocForm(FlaskForm)
  EditDocForm(FlaskForm)
  DeleteDocForm(FlaskForm)
- NewFirewallForm(FlaskForm)
  EditFirewallForm(FlaskForm)
  DeleteFirewallForm(FlaskForm)
- NewHypervisorForm(FlaskForm)
  EditHypervisorForm(FlaskForm)
  DeleteHypervisorForm(FlaskForm)
- NewNmsForm(FlaskForm)
  EditNmsForm(FlaskForm)
  DeleteNmsForm(FlaskForm)
- NewScenarioForm(FlaskForm)
  EditScenarioForm(FlaskForm)
  DeleteScenarioForm(FlaskForm)

- NewUserForm(FlaskForm)
  EditUserForm(FlaskForm)
  DeleteUserForm(FlaskForm)

Controllery prvků infrastruktury

- SuperController
- DocController
- FirewallController
- HypervisorController
- NmsController

Podpůrné třídy

- MailService
- Config

Poller

Třída poller slouží k provádění skenu sítě a je implementována dle návrhového vzoru singleton[1] kvůli spouštění pomocí `Redis Queue`, kdy bylo potřeba vyloučit vícenásobný běh v jednu chvíli. Uvnitř třídy jsou obsaženy metody `is_running()`, `get_progress()` a `run_poller()`.

Solver

Třída solver obsahuje metody:

- `advise_fix(sr)`
  Parametrem je problémový `ScanResult`, pro který chceme doporučit řešení.
- `solve_issue(sr, details)`
  Parametry jsou `ScanResult`, jehož problém řešíme, a položka details, která obsahuje potřebné údaje k řešení (typicky data vyplněná uživatelem) ve formě slovníku. Problémy jsou řešeny pomocí konkrétního pro danou platformu určeného controlleru zprostředkovaného `SuperControllerem`.

Translator

Třída `Translator` poskytuje metody, které slouží především k převodu JSON datových zpráv do lidsky čitelné podoby. Obsažené metody jsou:

- createHostIdentifier(host),
- getReadableSelector(scenario),
- getReadableRequirement(requirement),
- getReadableInfrastructureType(ifr_type),
- getReadableFixAction(requirement),
- getReadableScenario(scenario),
- getReadableScenarios(scenarios).

4.8 Struktura modulu platformy

Modul nemusí definovat žádné třídy, musí ale obsahovat několik metod a proměnných, ke kterým může jádro programu přistupovat.

Modul by měl implementovat metody:

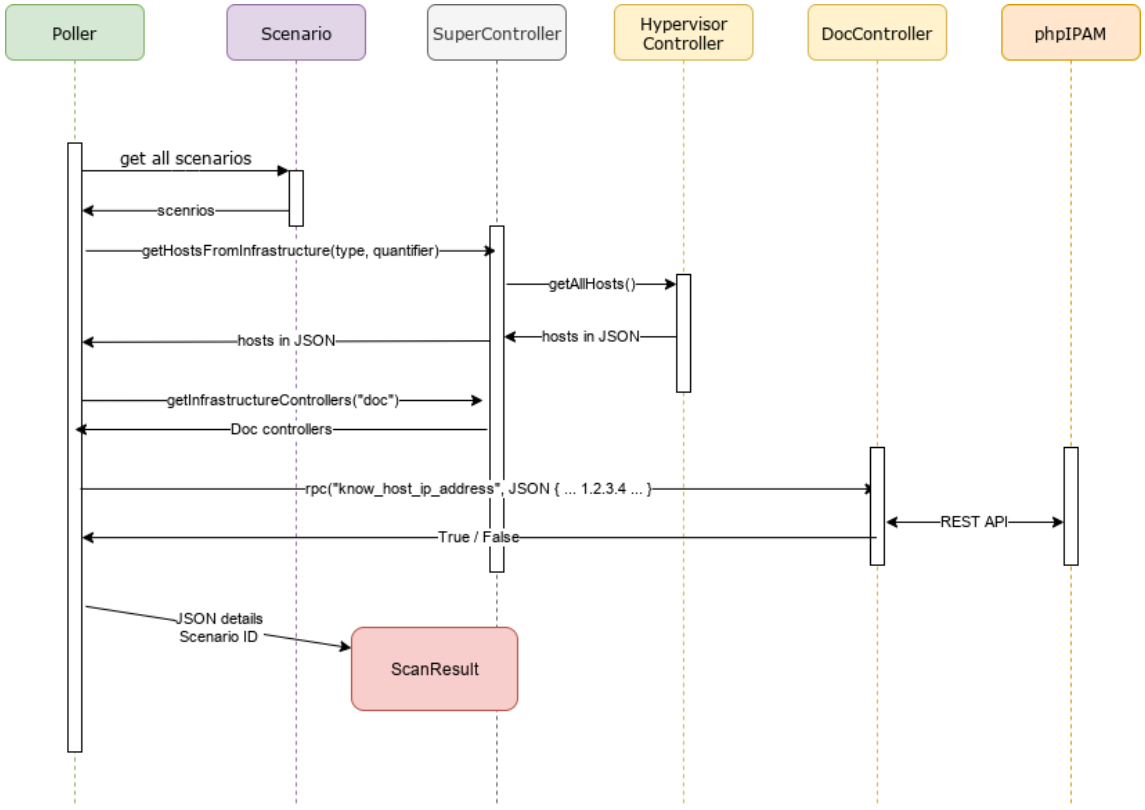
- `connect(credentials)`
  Provede připojení a vrací blíže nespecifikovanou proměnnou `connection`, která se předává dále volaným metodám.
- `rpc(connection, rpcname, details)`
  Slouží ke spouštění příkazů definovaných v akcích nebo opravách. Proměnná details obsahuje JSON s podrobnostmi prováděné akce.
- `getFixDetails(connection, fix_action, helper_data)`
  Tato metoda na základě předloženého názvu opravné akce a dat o problému vytvoří dictionary s navrhnovanou opravou.
- `disconnect(connection)`

Požadované proměnné jsou `CONFIG_SCHEMA`, `AVAILABLE_CHECKS`, `AVAILABLE_ACTIONS`. Jejich strukturu nejlépe ukáže následující část kódu:

```
CONFIG_SCHEMA = """{
    "apiuri" : {
        "type" : "string",
        "title" : "APIURI"
    },
    "user" : {
        "type" : "string",
```

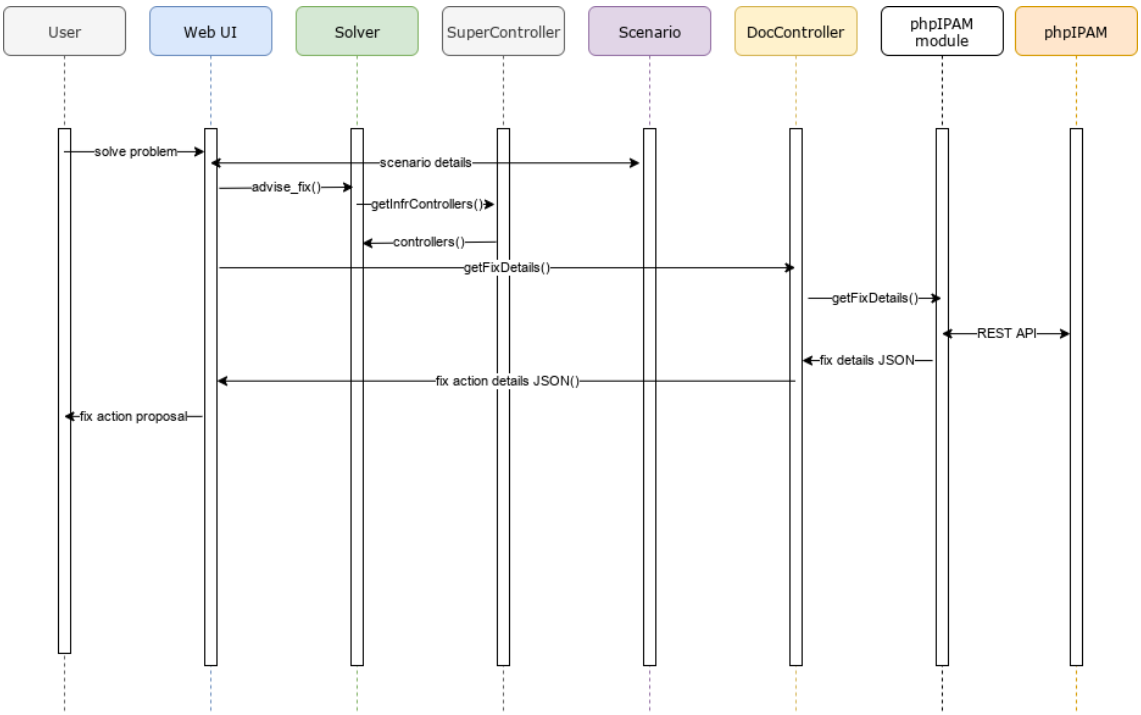
```
        "title" : "Username"
    },
        "password" : {
        "type" : "password",
        "title" : "Password",
    },
}"""
AVAILABLE_CHECKS = {
    "know_host_ip_address" : "host IP address record",
}
AVAILABLE_ACTIONS = {
    "add_host_ip_address" : "add host's IP address record",
}
```

4.9 Diagram procesu skenování sítě



Obrázek 4.4: Diagram procesu skenování sítě

4.10 Diagram procesu provádění opravy



Obrázek 4.5: Diagram návrhu opravy problému

Kapitola 5

Implementace, testování a nasazení

V této kapitole se budeme zabývat podrobnostmi z implementace softwaru, jeho testováním a nasazením.

5.1 Příprava a instalace

Následující návod je určen pro instalaci na linuxové distribuci Debian, pro jiné distribuce bude průběh podobný.

Nástroj je dodáván v komprimovaném archivu, pro jeho instalaci je třeba jej rozbalit a přesunout na správné místo ve filesystému:

```
:~$ tar xfv algochecker.tar.gz
:~$ sudo mv algochecker /opt
:~$ cd /opt/algochecker
```

Následuje instalace prostředí, tedy interpretru pro Python 3 a potřebné moduly.

```
:~$ sudo bash
:~# apt install python3-venv python3-pip gunicorn3
:~# pip3 install -r /opt/algochecker/requirements.txt
```

Následně v souboru `/opt/algochecker/config.py` upravíme parametry pro připojení k mailserveru, kontaktní údaje, adresu serveru a podobně. V souboru se nachází přesný popis konfiguračních voleb.

V dalším kroku je třeba připravit databázi.

```
:~$ sudo bash
:~# apt install mysql-server redis-server
:~# mysql -u root < /opt/algochecker/install.sql
```

Pokud jsme postupovali správně, mělo by být možné aplikaci spustit pomocí příkazu `/opt/algochecker/run.sh`. Následně bychom měli být schopni se ve webovém rozhraní přihlásit pomocí uživatelského jména `admin` s heslem `admin`.

Pro produkční běh doporučuji vytvořit `systemd` službu, která spouštění a vypínání služby obsáhne pomocí standardních systémových nástrojů.

5.2 Technologie webového rozhraní

Uživatelské rozhraní bylo navrženo s ohledem na jednoduchost a multiplatformnost.

Webová aplikace se skládá z HTML šablon, CSS kaskádových stylů, scriptů v jazyce JavaScript a HTTP serveru.

Bootstrap

Aplikace staví na frameworku Bootstrap[2], který výrazně usnadňuje tvorbu responzivních a dobře vypadajících aplikací. Disponuje rozsáhlou dokumentací, ve které jsou popsány jednotlivé grafické komponenty. Přiřazování rolí konkrétních komponent HTML prvku probíhá pomocí parametru `class`.

Velkou předností Bootstrapu je jeho grid systém, kdy obrazovku rozděluje na 12 sloupců, do kterých je možné umísťovat obsah.

JSON Form

JavaScriptový framework `JSON Form`[25] je využit ke generování variabilních formulářů na základě definice ve formátu JSON, kterou systém obdrží od modulu.

DataTables

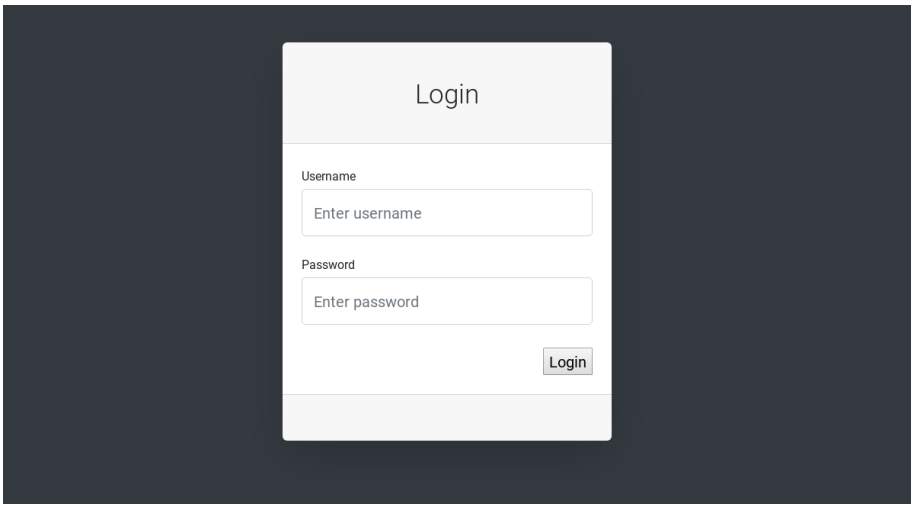
K zobrazení interaktivní tabulky Event logu je použit `JQuery` plugin `DataTables`[24]. Ten umožňuje provádět operace nad tabulkou v přímo v prohlížeči bez nutnosti implementovat operace na backendu.

5.3 Struktura webového rozhraní

Do systému přistupuje uživatel pomocí webového prohlížeče. Pro přihlášené uživatele se na levé straně obrazovky zobrazí vertikální menu s odkazy na následující blíže popsané stránky.

Přihlašovací obrazovka (Login)

Na úvodní stránce je vyžadováno přihlášení, po úspěšném zadání údajů lze pokračovat dále do systému.



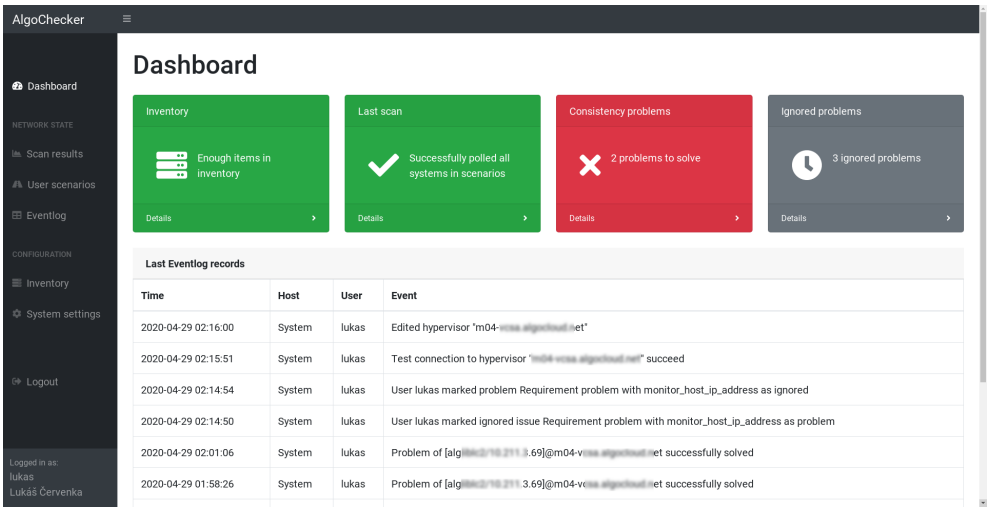
Obrázek 5.1: Web UI - Přihlašovací obrazovka

Dashboard

Na dashboard lze nalézt souhrn aktuálního stavu systému a poslední záznamy eventlogu.

V horní části se nacházejí čtyři boxy, které graficky upozorňují na základní stav systému. První kontroluje obsah inventáře komponent infrastruktury, přičemž hlásí chybu, pokud se v inventáři nenachází dostatek různých typů komponent. Další box zobrazuje stav posledního provedeného skenu sítě, poslední dva boxy poté ukazují počet nevyřešených či ignorovaných problémů.

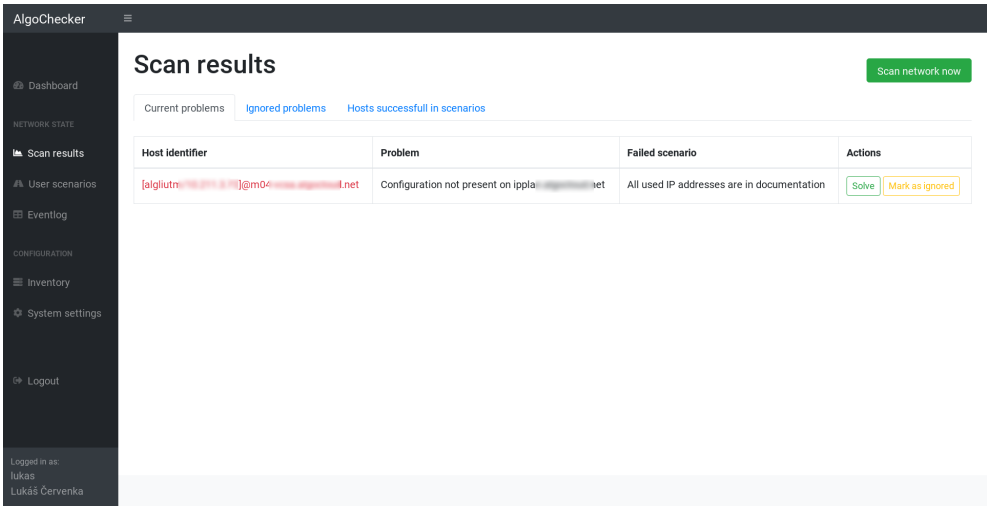
Ve spodní části je pro informaci úryvek logu událostí za poslední den seřazený od těch nejnovějších.



Obrázek 5.2: Web UI - Dashboard

Scan results

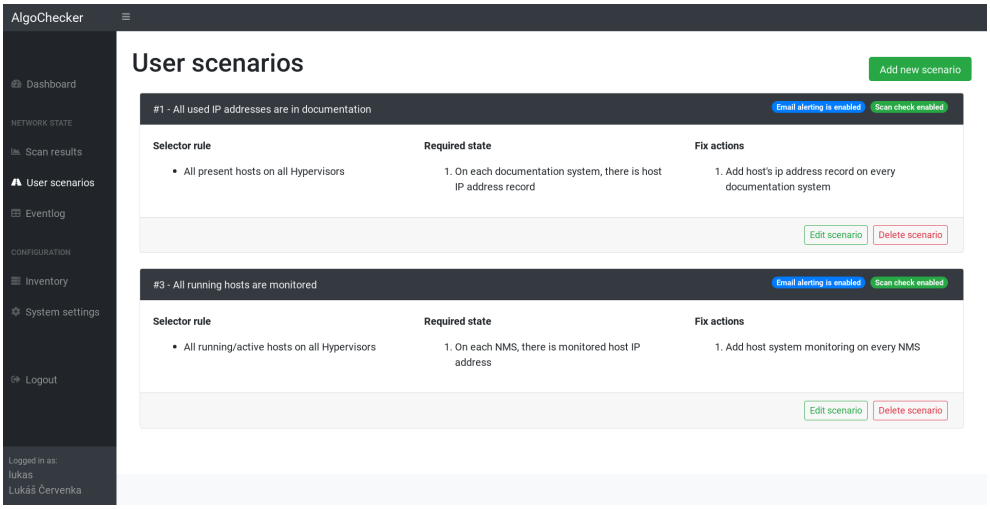
Na této stránce je zobrazen výsledek skenu sítě. Výstupem průchodu scénář může být problémový nebo korektní stav pro každého hosta na procházené platformě. Každý nalezený problém se může uživatel pokusit řešit nebo ignorovat pomocí tlačítek zobrazených u problému v seznamu. V horních záložkách lze přepínat mezi aktuálními a ingorovanými problémy a také informacemi o úspěšných průchodech scénářem. Na této stránce je též možné spustit skenování sítě manuálně.



Obrázek 5.3: Web UI - Výsledky skenování, nalezené problémy

User scenarios

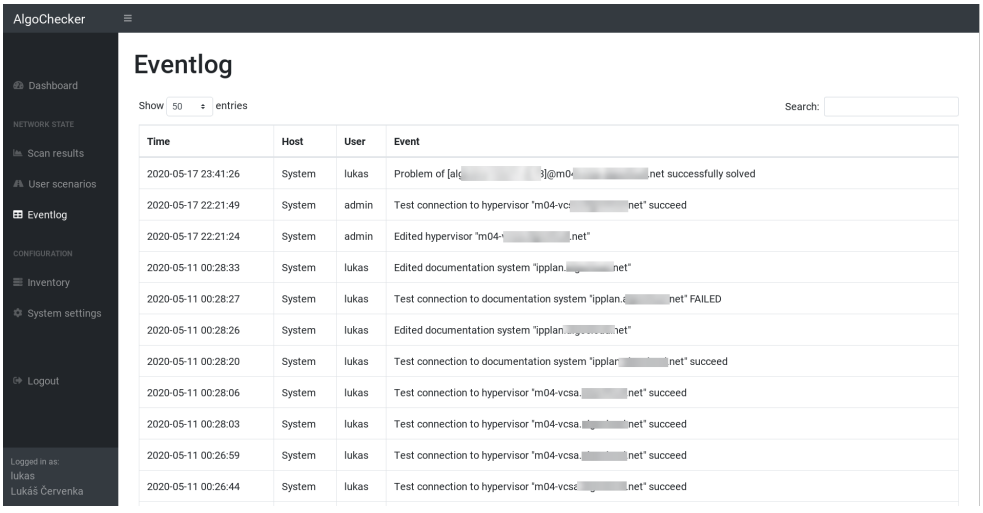
Na této stránce je uveden přehled uživatelem vytvořených scénářů kontroly. Pomocí tlačítek a navazujících stránek je možné scénáře vytvářet, upravovat a mazat.



Obrázek 5.4: Web UI - Uživatelské scénáře kontroly

Eventlog

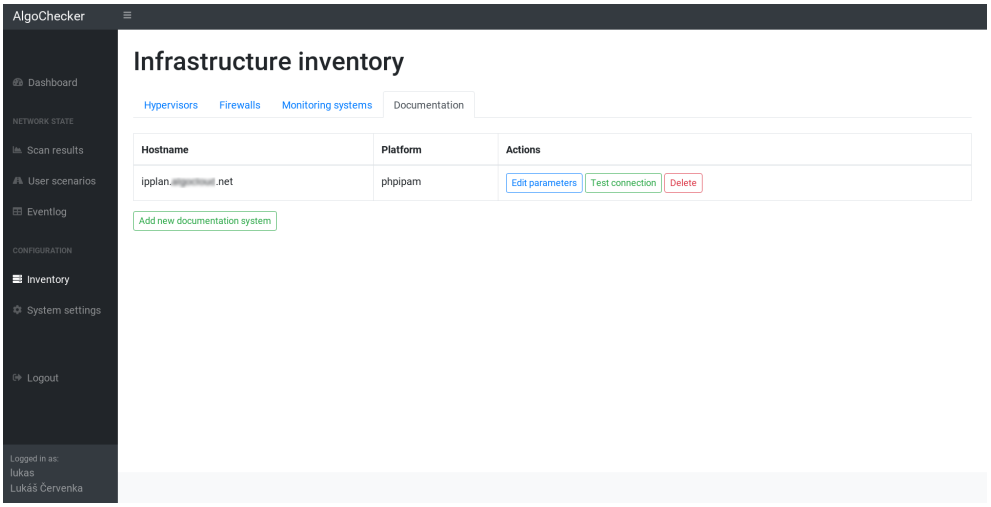
Zde se nachází log událostí systému (eventlog), kam se zaznamenávají podstatné provedené akce. V logu je možno vyhledávat podle klíčových slov.



Obrázek 5.5: Web UI - Eventlog

Inventory

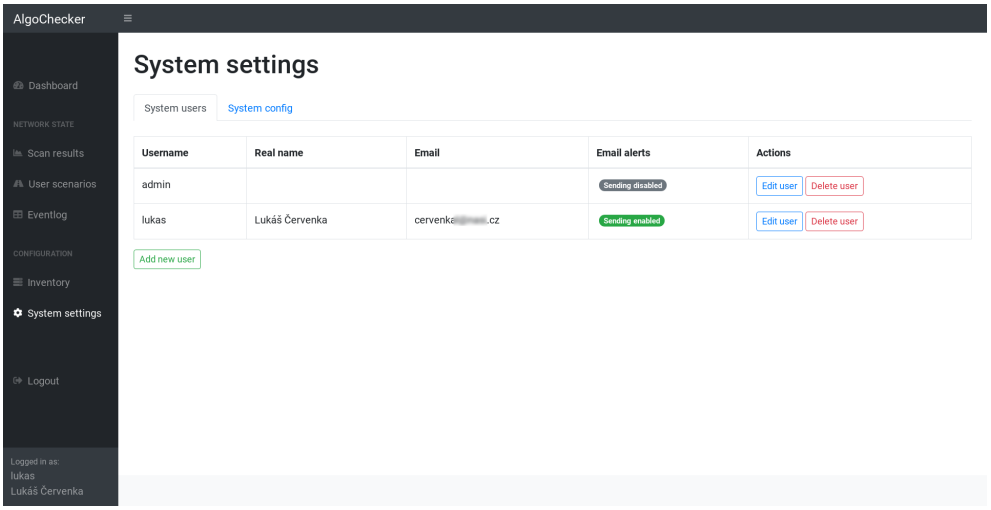
Inventář dává uživateli možnost prohlížet, přidávat, editovat a mazat jednotlivé síťové systémy a komponenty. Je také možné otestovat spojení a ověřit, zda jsou zadány správné přihlašovací údaje a zda je modul pro ovládání dané platformy funkční.



Obrázek 5.6: Web UI - Inventář komponent infrastruktury

System settings

V této sekci naleznene nastavení systému, především tedy správu uživatelských účtů. Je možné uživatele přidávat, upravovat, mazat a měnit jejich hesla.



Obrázek 5.7: Web UI - Správa uživatelských účtů

5.4 Popis a průchod klíčových funkcionalit

5.4.1 Vytváření a správa scénářů kontroly

Scénáře kontroly je možné správovat na výše popsané stránce `User scenarios`. Scénář se skládá ze dvou částí. První je tzv. selektor (`selector`), kterým uživatel určí, které záznamy v reálné síti se vezmou jako pravdivé a správné - tedy jako výchozí bod kontroly. Uživatel kombinuje volby, které nabízí systém. Například tedy:

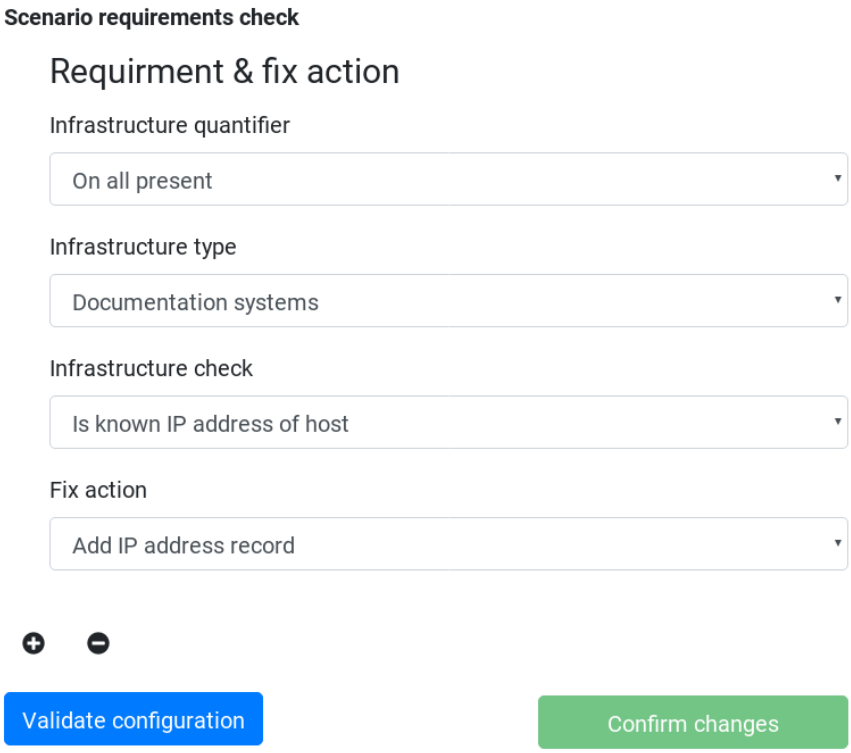
- Pro všechny existující záznamy o hostech ve všech dokumentačních systémech
- Pro všechny aktivní/spuštěné hosty na všech hypervizorech

Obrázek 5.8: Uživatelský scénář - selector

Druhou částí je požadavek (`requirement`), tedy stav, který požadujeme pro každého hosta nalezeného pomocí dříve definovaného selektoru a způsob nápravy, pokud kontrola selže. Takových požadavků může být v jednom scénáři více. Výběr možností konkrétních kontrol (tedy stavů) a akcí k opravě je určen v jednotlivých modulech.

Ukázky zamýšlených stavů:

- Zda je známa IP adresa / hostname hosta
- Zda má host nainstalované guest additions
- Zda má host záznam ve firewallu



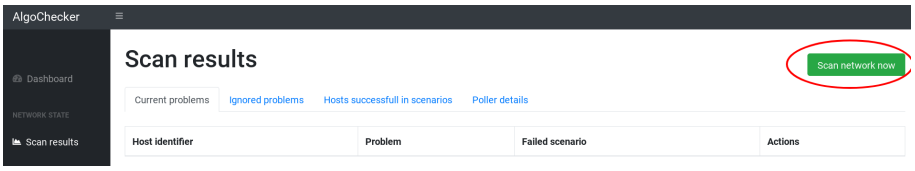
Obrázek 5.9: Uživatelský scénář - requirement

5.4.2 Skenování sítě a řešení nalezeného problému

Sken sítě je možné ručně spustit na stránce `Scan results`. Po spuštění se uživateli zobrazí progress bar, který ukazuje průběh skenu, a následně výsledky. Vzhledem k variabilitě možných kontrol a prováděných akcí popíšu ukázkový průchod na již připravené instalaci.

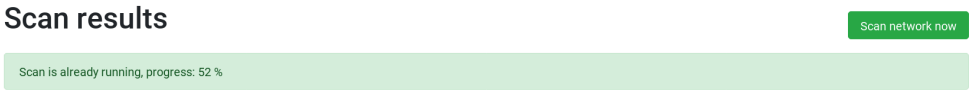
Ukázkový průchod skenu uživatelem

1. Zobrazení stránky `Scan results`, stisknutí `Scan network now`



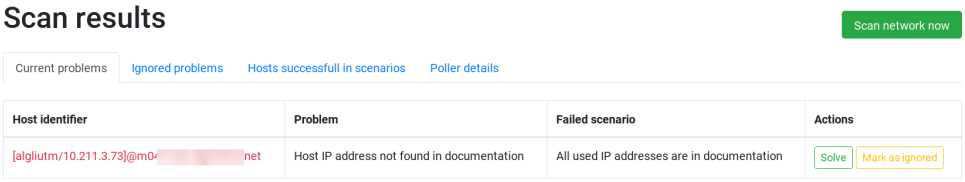
Obrázek 5.10: Ukázka - Stránka Scan results 1

2. Zobrazí se progress bar, na kterém postupně přibývají procenta. Po dosažení 100 % zmizí a nahradí jej stránka s výsledky.



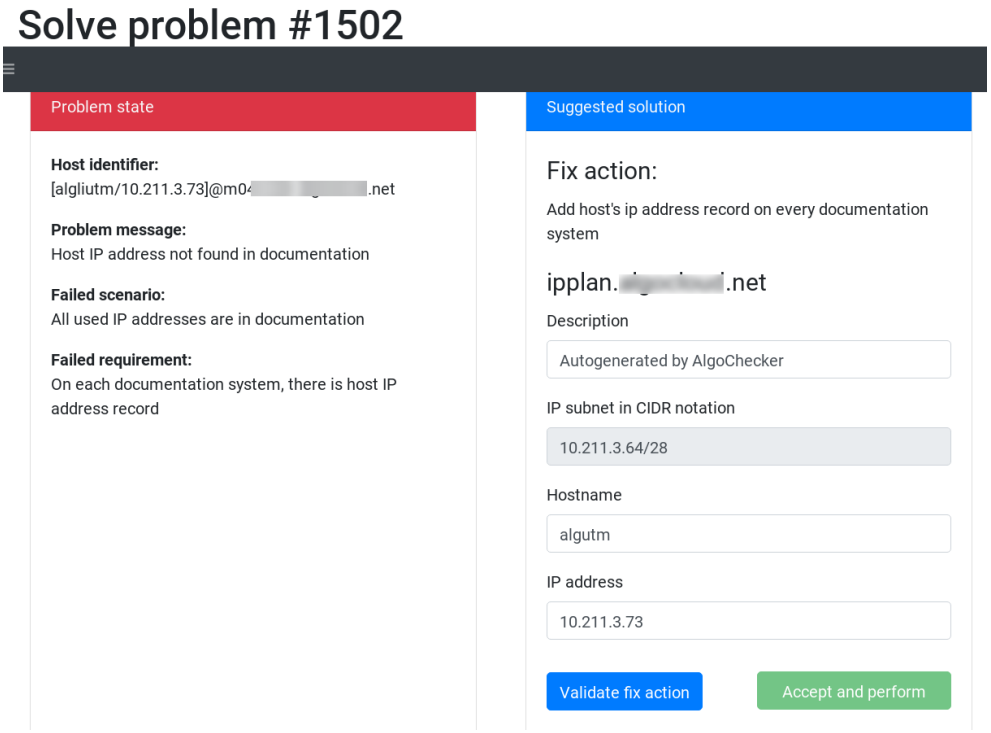
Obrázek 5.11: Ukázka - Progress bar při skenování

3. Na tuto stránku je uživatel přesměrován po skončení skenu. Vidíme, že byl nalezen problém s hostem s IPv4 adresou `10.211.3.73`, která není uvedena v dokumentaci. Klikneme na tlačítko `Solve`.



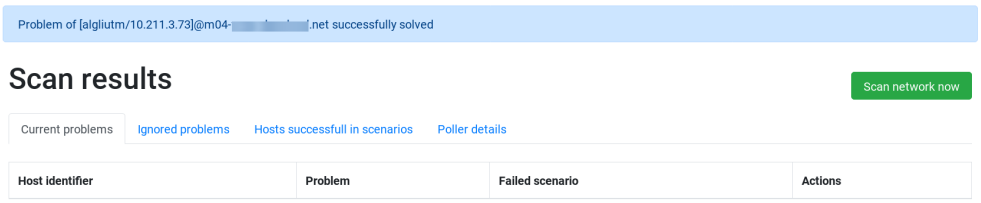
Obrázek 5.12: Ukázka - Stránka Scan results 2

4. Dostali jsme se na stránku pro řešení problému. Systém zde informuje uživatele o detailech nalezeného problému a předkládá návrh jeho řešení, který může uživatel upravit a poté nechat provést. Navrhované údaje jsou správné, volíme stisk tlačítka `Validate fix action` a po úspěšné validaci `Accept and perform`.



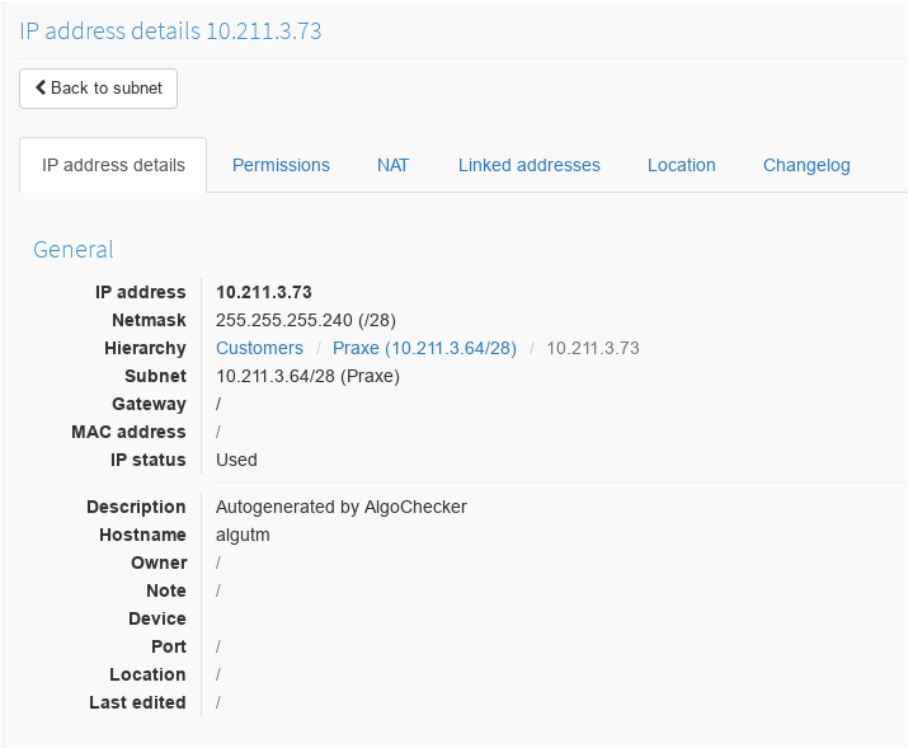
Obrázek 5.13: Ukázka - Návrh řešení problému

5. Uživateli se objevuje zpět stránka `Scan results` s hláškou, že problém byl úspěšně vyřešen. Pro jistotu spouštíme ještě jednou skenování sítě, problém se ale již znova nezobrazuje.



Obrázek 5.14: Ukázka - Stránka Scan Results 3

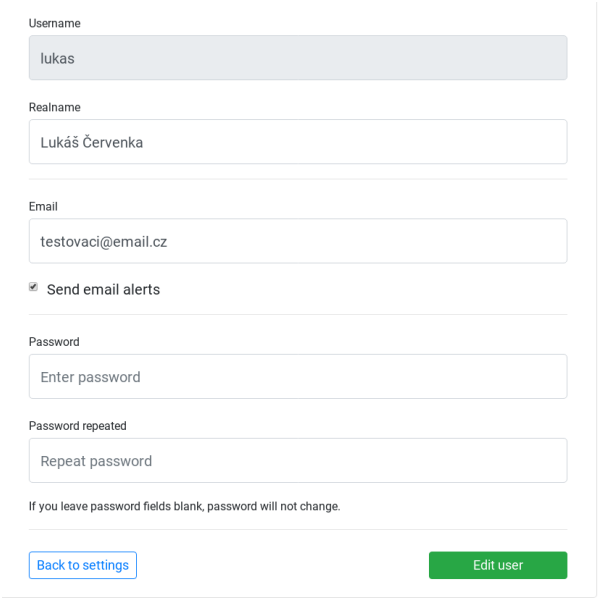
6. V rámci kontroly navštívíme webové rozhraní dokumentačního systému, kde vyhledáme uvedenou IP adresu a zkontrolujeme její uložené parametry.



Obrázek 5.15: Ukázka - Dokumentační systém phpIPAM

5.4.3 Správa uživatelských účtů

Uživatel může spravovat účty pro přístup do systému v nastavení. Není povoleno odstranit uživatele `admin` ani vlastní uživatelský účet.



Obrázek 5.16: Editace uživatelského účtu

5.4.4 Záloha a obnova nastavení

Vzhledem k očekávané nízké četnosti použití této funkcionality není implementováno její grafické rozhraní. Zálohu lze provést pomocí nástroje `mysqldump`, konkrétně:

```
mysqldump -u MySQL_USER -p MySQL_DB > backup.sql
```

Obnovu nastavení poté provedeme obdobně importem SQL souboru do databáze pomocí příkazu:

```
mysql -u MySQL_USER -p MySQL_DB < backup.sql
```

5.5 Testování softwaru

Jelikož se software skládá z programového jádra a modulů pro jednotlivé platformy, je vhodné testovat obě části. Následovalo testování softwaru jako celku na reálné infrastruktuře.

5.5.1 Testování jádra

Backend systému byl testován s důrazem na práci s daty v databázi, tedy na metody spojené se správou uživatelských účtů, inventáře a scénáře. K tomu byla použita knihovna `pytest`, která umožňuje spouštění testovacích metod a kontrolovat očekávané výstupy pro vložené vstupy.

Webový frontend jsem testoval na prohlížečích Opera a Firefox pomocí průchodu předem definovaných kroků na zařízeních s různým operačním systémem a rozlišením obrazovky, kdy jsem sledoval správnost rozmístění prvků a chybovou konzoli ve vývojářských nástrojích prohlížeče.

5.5.2 Testování na reálné infrastruktuře

Celkové chování systému bylo možné otestovat až na reálné infrastruktuře. V tuto chvíli je systém nainstalován v testovacím prostředí, odkud má přístup ke spuštěným instancím jednotlivých síťových komponent. Zde byly rovněž testovány všechny `RPC` metody modulů pro různé stavy komponent a vstupy, aby byla otestována správná reakce UI na data zaslaná moduly.

5.5.3 Testování modulů

Pro implementované moduly byla testována každá podstatná funkcionalita využívající vzdálené volání přes API síťové komponenty. Vzhledem k rozsahu testů je uvedena jedna ukázka. Přihlašovací údaje jsou anonymizovány. Byly testovány metody v modulech:

- Modul pro hypervizor VMware ESXi
- Modul pro dokumentační systém phpIPAM
- Modul pro NMS Zabbix
- Modul pro firewall Sophos

V rámci každého modulu byly testovány jeho `RPC` metody za použití testovacích instalací síťových komponent, aby API volání probíhalo po skutečné síti. Byl k tomu použit vlastní script, který na definované vstupy očekával konkrétní výstupy.

Ukázka jednoho testu v rámci modulu pro Zabbix

Otestujeme vyhledávání v databázi monitorovaných hostů podle IP adresy agenta, tedy RPC metodu `monitor_host_ip_address` implementovanou pro konkrétní platformu. Za vstup použijeme jednoho z hostů na hypervizoru. Jelikož je daný host Zabbixem skutečně monitorován a my poskytneme validní přihlašovácí údaje k API, očekáváme úspěch, tedy vrácení interního `host_id` hosta ze Zabbixu.

Vstup testu:

```
NMS_HOSTNAME = "Migration test Zabbix"
NMS_PLATFORM = "zabbix4"
NMS_CREDENTIALS = '''{
    "apiuri": "https://10.2.3.4/zabbix/",
    "user": "Admin",
    "password": "nbusr123"
}'''

nc = NmsController(NMS_HOSTNAME, NMS_PLATFORM, NMS_CREDENTIALS)
host = {
    'hostname': 'algliblc2',
    'name': 'algliblc2',
    'ip_address': '10.211.3.69',
    ,..
}
print(nc.rpc("monitor_host_ip_address", host))
```

Výstup:

```
$: ./test.py
11122
```

Závěr testu:
Test proběhl úspěšně, na předložený vstup jsme obdrželi předpokládaný výstup.

Kapitola 6

Závěr

Předkládaná diplomová práce se nejprve zabývá sumarizací možností monitoringu rozsáhlých sítí a též jejich správou. Ze zkušeností z produkčního nasazení je porovnáno několik nástrojů pro monitoring.

Hlavní náplní je ale návrh, implementace a testování nástroje pro centralizovanou kontrolu integrity nastavení jednotlivých komponent virtualní infrastruktury.

Výstupem je softwarový nástroj implementovaný v jazyce Python, který přes webové rozhraní umožňuje uživateli vytvářet scénáře kontrol a oprav nastavení systémů a prvků sítě.

Tento nástroj byl také testován na reálné infrastruktuře společnosti Algotech a podporuje komunikaci s hypervizory VMware ESXi, firewally Sophos, NMS Zabbix a dokumentačním systémem phpIPAM. Dle požadavků ze zadání je také poměrně snadné doplnit funkcionalitu softwaru o podporu dalších platforem.

Literatura

[1] Algoritmy.net. *Singleton* [online]. Algoritmy.net. [cit. 5. 5. 2020]. Dostupné z: <https://www.algoritmy.net/article/1326/Singleton>.

[2] Bootstrap. *Bootstrap* [online]. Bootstrap. [cit. 2. 5. 2020]. Dostupné z: <https://getbootstrap.com>.

[3] Cisco. *802.11ac: The Fifth Generation of Wi-Fi* [online]. Cisco. [cit. 1. 5. 2020]. Dostupné z: <https://www.cisco.com/c/dam/en/us/products/collateral/wireless/aironet-3600-series/white-paper-c11-713103.pdf>.

[4] Cisco. *IOS Documentation: MPLS VPN—SNMP Notifications* [online]. Cisco. [cit. 17. 5. 2020]. Dostupné z: <https://www.cisco.com/en/US/docs/ios/mpls/configuration/guide/mp_vpn_snmp_notify_external_docbase_0900e4b1805dff18_4container_external_docbase_0900e4b182004c8d.html>.

[5] ENNS, R. et al. Network Configuration Protocol (NETCONF). RFC 6241 (Proposed Standard), June 2011. Dostupné z: <http://www.ietf.org/rfc/rfc6241.txt>.

[6] FIELDING, R. et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Dostupné z: <http://www.ietf.org/rfc/rfc2616.txt>. Updated by RFCs 2817, 5785, 6266, 6585.

[7] Icinga. *Icinga 2 Addons and Integrations* [online]. Icinga. [cit. 16. 5. 2020]. Dostupné z: <https://icinga.com/docs/icinga2/latest/doc/13-addons/>.

[8] Icinga. *Distributed Monitoring with Master, Satellites and Agents* [online]. Icinga. [cit. 16. 5. 2020]. Dostupné z: <https://icinga.com/docs/icinga2/latest/doc/06-distributed-monitoring/>.

[9] Icinga. *Icinga web page* [online]. Icinga. [cit. 16. 5. 2020]. Dostupné z: <https://icinga.com>.

[10] LAMMLE, Todd. *CCNA: výukový průvodce přípravou na zkoušku 640-802.* Brno : Computer Press, 2010. ISBN: 978-80-251-2359-1.

[11] LEHTINEN, S. – LONVICK, C. The Secure Shell (SSH) Protocol Assigned Numbers. RFC 4250 (Proposed Standard), January 2006. Dostupné z: <http://www.ietf.org/rfc/rfc4250.txt>.

[12] MANNING, Trevor. *Microwave Radio Transmission Design Guide*. United States : Artech House Publishers, 2009. ISBN: 978-1-59693-456-6.

[13] Nagios. *Nagios: Plugins* [online]. Nagios. [cit. 15. 5. 2020]. Dostupné z: <https://exchange.nagios.org/directory/Plugins>.

[14] Observium. *Observium: Poller Partitioning* [online]. Observium. [cit. 15. 5. 2020]. Dostupné z: <https://docs.observium.org/partitioning/>.

[15] Observium. *Observium: Tuning* [online]. Observium. [cit. 15. 5. 2020]. Dostupné z: <https://docs.observium.org/tuning/>.

[16] Observium. *Observium web page* [online]. Observium. [cit. 15. 5. 2020]. Dostupné z: <https://observium.org/>.

[17] Oracle. *MySQL Performance Tuning - Oracle* [online]. Oracle. [cit. 20. 5. 2020]. Dostupné z: <https://www.oracle.com/technetwork/community/developer-day/mysql-performance-tuning-403029.pdf>.

[18] POSTEL, J. – REYNOLDS, J. Telnet Protocol Specification. RFC 854 (INTERNET STANDARD), May 1983. Dostupné z: <http://www.ietf.org/rfc/rfc854.txt>. Updated by RFC 5198.

[19] Puppet. *Puppet web page* [online]. Puppet. [cit. 1. 5. 2020]. Dostupné z: <https://puppet.com>.

[20] Red Hat. *Ansible web page* [online]. Red Hat. [cit. 1. 5. 2020]. Dostupné z: <https://www.ansible.com>.

[21] Sbírka zákonů. *127/2005 Sb. Zákon o elektronických komunikacích* [online]. Sbírka zákonů. [cit. 1. 5. 2020]. Dostupné z: <https://www.zakonyprolidi.cz/cs/2005-127>.

[22] Sophos. *Docs* [online]. Sophos. [cit. 10. 5. 2020]. Dostupné z: <https://developer.sophos.com/intro>.

[23] Spiceworks. *The 2020 State of Virtualization Technology* [online]. Spiceworks. [cit. 10. 5. 2020]. Dostupné z: <https://www.spiceworks.com/marketing/reports/state-of-virtualization/>.

[24] SpryMedia, Ltd. *datatables.net* [online]. SpryMedia, Ltd. [cit. 5. 5. 2020]. Dostupné z: <https://datatables.net>.

[25] tchapi. *GitHub JSON Form* [online]. tchapi. [cit. 2. 5. 2020]. Dostupné z: <https://github.com/jsonform/jsonform>.

[26] Ubiquiti Networks. *UBNT Forum: Scaling UNMS - Problems* [online]. Ubiquiti Networks. [cit. 15. 5. 2020]. Dostupné z: <https://community.ui.com/questions/Scaling-UNMS-Problems-/aef6b4f6-c415-4ed9-bfe6-b51d74e02e4c>.

[27] Ubiquiti Networks. *UNMS web page* [online]. Ubiquiti Networks. [cit. 15. 5. 2020]. Dostupné z: <https://unms.com/>.

[28] Zabbix. *Zabbix Documentation 5.0* [online]. Zabbix. [cit. 16. 5. 2020]. Dostupné z: <https://www.zabbix.com/documentation/current/manual/discovery/low_level_discovery>.

[29] Zabbix. *Zabbix: Distributed Monitoring* [online]. Zabbix. [cit. 16. 5. 2020]. Dostupné z: <https://www.zabbix.com/distributed_monitoring>.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CLI Command Line Interface

DNS Domain Name System

HTTPS Hypertext Transfer Protocol Secure

IEEE Institute of Electrical and Electronics Engineers

IM Instant Messaging

IMAP Internet Message Access Protocol

ISP Internet Service Provider

LAN Local Area Network

MAN Metropolitan Area Network

MIB Management information base

MPLS Multiprotocol Label Switching

NMS Network monitoring system

NRPE Nagios Remote Procedure Execution

PAN Personal Area Network

RPC Remote Procedure Call

RRD Round-robin database

SNMP Simple Network Management Protocol

SNMP Simple Network Monitoring Protocol

SNR Signal-to-noise ratio

SSH Secure Shell

UBNT Ubiquiti Networks

UDP User Datagram Protocol

VPN Virtual Private Network

WAN Wide Area Network

WSGI Web Server Gateway Interface

Příloha B

Obsah přiloženého CD

`./app_source/`	zdrojové kódy vyvíjené aplikace
`./dp_text_source/`	zdrojové kódy textu práce
`./cervelu5_dp.pdf`	text práce